Computer-Aided Analysis of Fetal Cardiac Ultrasound Videos

Christopher Philip Bridge

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2017

Acknowledgements

Personal

Firstly I would like to thank my family – Jo, Martin and Sarah – for all their support and encouragement over the last three and a half years. This DPhil would also have been far harder without the support of Chenzi Xu, whose belief in me has helped more than she realises.

I am also grateful to all the other members of the BioMedIA lab at Oxford for their friendship and camaraderie, which has made the lab a great place to work. Especially, and in no particular order, Rui "Lord" Jia, Haiyue Yu, Davis Vigneault, Ruobing Huang, Ali Maraci and Ana Namburete. I wish them all the best for the future.

Finally, I would like to thank my supervisor Alison for her support and advice throughout this process. Also Christos Ioannou's guidance on the clinical aspects of the project has been invaluable, and I am very grateful for his work collecting data.

Institutional

I am grateful to the Engineering and Physical Sciences Research Council (EPSRC) for giving me the oppurtunity to study for this DPhil by funding my studies through a Doctoral Training Award, and to the Institute of Engineering and Technology (IET) for funding my travel to the International Symposium on Biomedical Imaging (ISBI) 2015 through a travel award.

Chris Bridge
Oxford, April 2017

Abstract

This thesis addresses the task of developing automatic algorithms for analysing the two-dimensional ultrasound video footage obtained from fetal heart screening scans. These scans are typically performed in the second trimester of pregnancy to check for congenital heart anomalies and require significant training and anatomical knowledge to perform.

The aim is to develop a tool that runs at high frame rates with no user initialisation and infers the visibility, position, orientation, view classification, and cardiac phase of the heart, and additionally the locations of cardiac structures of interest (such as valves and vessels) in a manner that is robust to the various sources of variation that occur in real-world ultrasound scanning. This is the first work to attempt such a detailed automated analysis of these videos.

The problem is posed as a Bayesian filtering problem, which provides a principled framework for aggregating uncertain measurements across a number of frames whilst exploiting the constraints imposed by anatomical feasibility. The resulting inference problem is solved approximately with a particle filter, whose state space is partitioned to reduce the problems associated with filtering in high-dimensional spaces.

Rotation-invariant features are captured from the videos in an efficient way in order to tackle the problem of unknown orientation. These are used within random forest learning models, including a novel formulation to predict circular-valued variables.

The algorithm is validated on an annotated clinical dataset, and the results are compared to estimates of inter- and intra-observer variation, which are significant in both cases due to the inherent ambiguity in the imagery. The results suggest that the algorithm's output approaches these benchmarks in several respects, and fall slightly behind in others.

The work presented here is an important first step towards developing automated clinical tools for the detection of congenital heart disease.

Contents

List of Figures

List of Abbreviations

1D, 2D, 3D . . *One-*, *two-* or *three-dimensional*, etc.

3V . . . . . The *three vessels view* of the fetal heart showing the pulmonary artery, aorta and superior vena cava.

4C . . . . . The *four chamber view* of the fetal heart.

ANN . . . . . *Artificial neural network*, a machine learning algorithm.

Ao . . . . . *Aorta*.

CHD . . . . . *Congenital heart disease*.

CNN . . . . . *Convolutional neural network*, a machine learning algorithm.

CPU . . . . . A computer's *central processing unit*.

dAo . . . . . *Descending aorta*.

DT . . . . . *Dynamic texture*, a video processing model.

FFT . . . . . *Fast Fourier transform*, a signal processing algorithm.

GPU . . . . . A computer's *graphics processing unit*.

HMM . . . . . *Hidden Markov model*, a statistical model.

HOG . . . . . *Histogram of oriented gradient*, an image representation.

IFFT . . . . . *Inverse fast Fourier transform*.

ISUOG . . . . The *International Society of Ultrasound in Obstetrics and Gynecology*.

KDT . . . . . *Kernel dynamic texture*, a video processing model.

LA . . . . . *Left atrium*.

LDS . . . . . *Linear dynamical system*, a system model.

LV . . . . . *Left ventricle*.

LVOT . . . . . The *left ventricular outflow tract* view of the fetal heart.

PCA . . . . . *Principal component analysis*, a statistical algotihm.

PDF . . . . . *Probability density function* of a distribution over a continuous random variable.

PMF . . . . . *Probability mass function* of a distribution over a discrete random variable.

PA . . . . . *Pulmonary artery.*

RA . . . . . *Right atrium.*

RAM . . . . . A computer's *random access memory.*

R-CNN . . . . *Region-convolutional neural network*, a machine learning algorithm.

RIF . . . . . *Rotation-invariant feature*, an image patch representation.

RNN . . . . . *Recurrent neural network*, a machine learning algorithm.

RV . . . . . *Right ventricle.*

RVOT . . . . . The *right ventricular outflow tract* view of the fetal heart.

SIFT . . . . . *Scale invariant feature transform*, an image patch representation.

SIR . . . . . The *sequential importance resampling* algorithm.

SIS . . . . . The *sequential importance sampling* algorithm.

SURF . . . . . *Speeded up robust features*, an image patch representation.

SVC . . . . . *Superior vena cava.*

SVD . . . . . The *singular value decomposition* of a matrix.

SVM . . . . . *Support vector machine*, a machine learning algorithm.

US . . . . . *Ultrasound.*

List of Mathematical Notation

$\mathbf{i}$	The imaginary unit, $\mathbf{i} = \sqrt{-1}$.
e	The basis of natural logarithms, $e \approx 2.718$.
$\mathbb{N}$	The set of natural numbers (strictly positive integers), $\{1, 2, 3, \ldots, \infty\}$.
$\mathbb{N}_0$	The set of non-negative integers $\{0, 1, 2, \ldots, \infty\}$.
$\mathbb{Z}$	The set of integers $\{-\infty, \ldots, -1, 0, 1, \ldots, \infty\}$.
$\mathbb{Z}_{n,m}$	The set of integers between n and m inclusive, $\{n, n+1, \ldots, m\}$, where $n \in \mathbb{Z}$, $m \in \mathbb{Z}$, $n \leq m$.
$\mathbb{R}$	The set of real numbers.
$\mathbb{R}^+$	The set of strictly positive real numbers.
$\mathbb{R}_0^+$	The set of non-negative real numbers.
$\mathbb{R}^d$	The set of real-valued d-dimensional vectors, $d \in \mathbb{N}$.
$\mathbb{C}$	The set of complex numbers.
$\mathbf{v}$	Lower-case boldface represents a vector.
$\mathbf{M}$	Upper-case boldface represents a matrix.
$\mathbf{I}$	An identity matrix.
$\mathbf{v}^T$	The transpose of the vector (or matrix) $\mathbf{v}$.
$\mathcal{S}$	Calligraphic typeface represents a set (or a transform or distribution for the specific cases below).
$\lvert\mathcal{S}\rvert$	The cardinality of (number of elements in) the set $\mathcal{S}$.
$\overline{x}$	The complex conjugate of a complex number x.
$\mathrm{Re}(x)$	The real part of a complex number x.
$\mathrm{Im}(x)$	The imaginary part of a complex number x.
$\angle x$	The complex argument of a complex number x.
$\delta(\cdot)$	The Dirac delta distribution.
$\Gamma(\cdot)$	The gamma function.
$J_n(\cdot)$	The Bessel function of the first kind of order $n \in \mathbb{C}$.

$I_n(\cdot)$ . . . . . The modified (or hyperbolic) Bessel function of the first kind of order $n \in \mathbb{C}$.

$H_n(\cdot)$ . . . . . The Struve function of order $n \in \mathbb{C}$.

${}_pF_q\left(\begin{matrix}a_1,\ldots,a_p\\ b_1,\ldots,b_q\end{matrix}\middle|\,\cdot\right)$ The generalised hypergeometric function.

$\mathcal{F}_d[\cdot]$ . . . . . The d-dimensional Fourier transform of a function.

$\mathcal{H}_n[\cdot]$ . . . . . The n^{th} order Hankel transform of a function.

$\mathcal{N}_D(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ . . . . The PDF of an D-dimensional Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

$\mathcal{D}(\cdot \mid \mathbf{v})$ . . . . . A discrete probability mass function over the integers $\mathbb{Z}_{1,n}$ where $\mathbf{v} \in \mathbb{R}^n$ is a vector (with sum 1) containing the probabilities of each integer.

$\mathcal{V}(\cdot \mid \mu, \kappa)$ . . . . . The PDF of a von Mises distribution with circular angle μ and concentration parameter κ.

$\mathcal{W}(\cdot \mid \mu, \sigma)$ . . . . . The PDF of a wrapped normal distribution with circular angle μ and variance σ.

$y \sim p(x)$ . . . . . The value y is sampled from the valid probability density or probability mass function $p(x)$.

$x_{n:m}$ . . . . . Colons in a subscript denote all the values of x with subscripts in the range n to m inclusive.

1 Introduction and Preliminaries

Contents

This thesis investigates automated image analysis algorithms to assist with ultrasound screenings for congenital heart disease (CHD). This chapter describes CHD and the associated screening procedures currently used in clinical practice before outlining the contributions of this thesis.

1.1 Anatomy and Function of the Fetal Heart

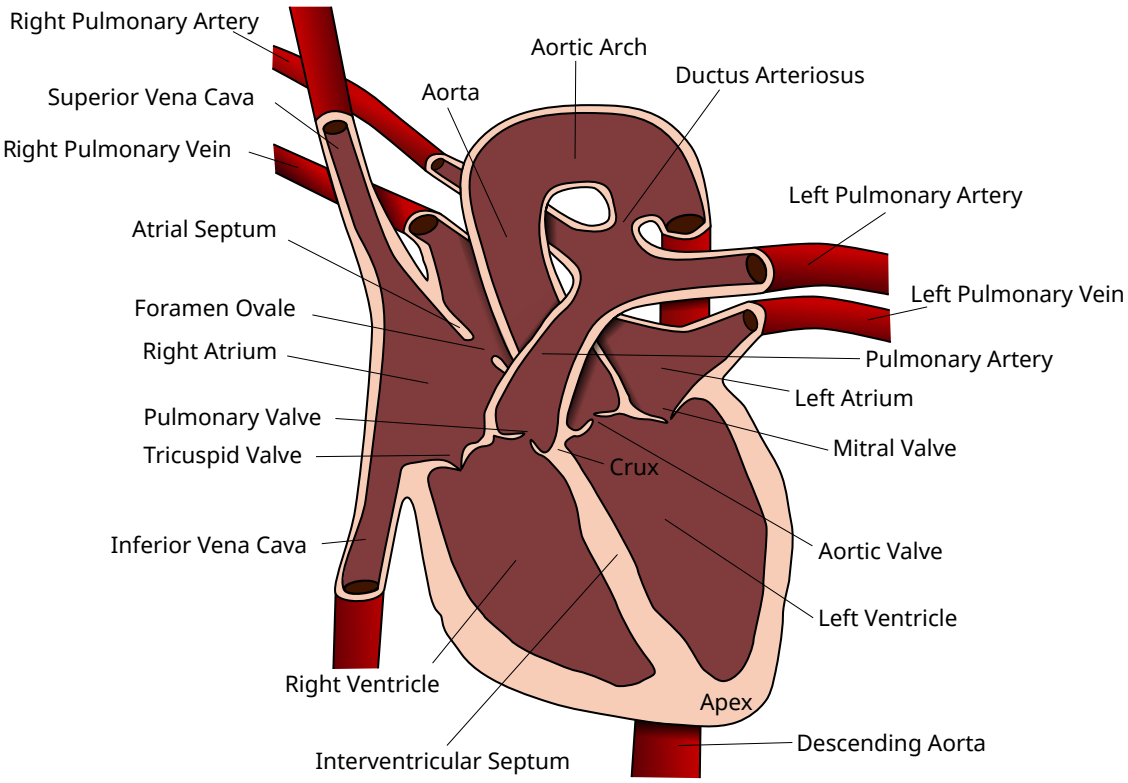
The anatomy of the fetal heart and the surrounding vasculature at gestational ages of approximately 18 weeks and beyond is relevant here, as this is the time when screening for CHD typically takes place. A basic overview of the relevant

anatomy is presented here for the reader who is unfamiliar with this material. For a more detailed discussion, the reader should consider the texts by Yagel, Silverman and Gembruch [1] (for a thorough discussion) and Archer and Manning [2] (for a more concise introduction).

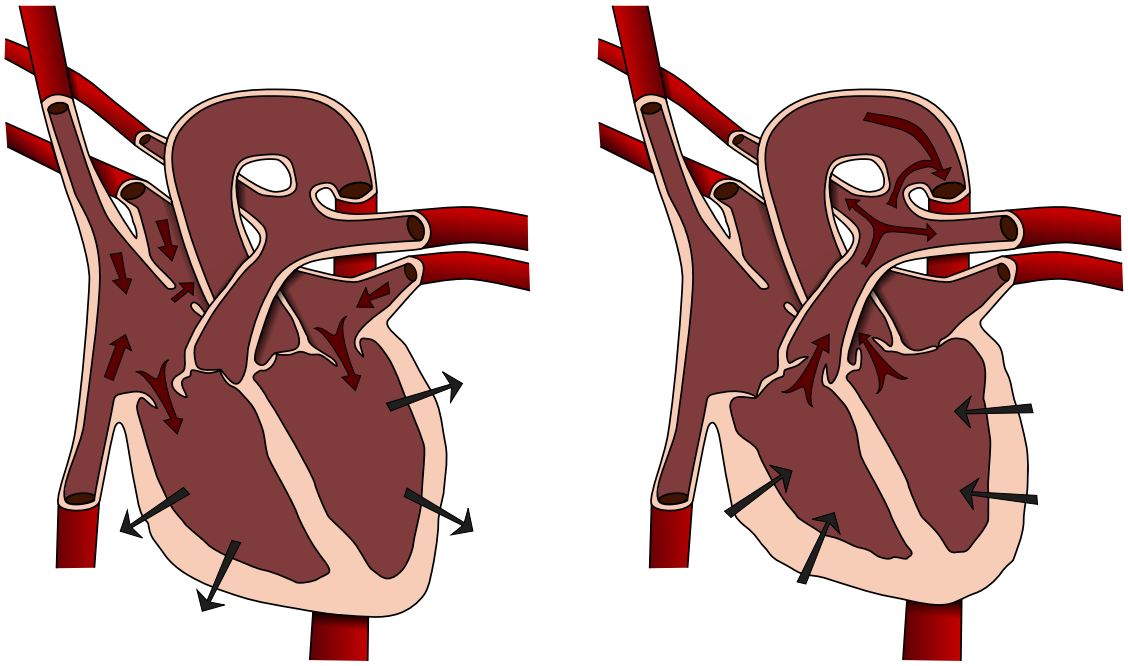
Figure 1.1 shows the anatomy of the fetal heart during this period. By this stage of development, the fetal heart has developed the four chambers of the adult heart (the *left ventricle*, *right ventricle*, *left atrium* and *right atrium*) and has begun to circulate blood through the fetal cardiovascular system.

Broadly, the cardiac cycle can be considered to have two stages, known as *ventricular diastole* and *ventricular systole*. Hereafter, these shall be referred as simply *diastole* and *systole*. During diastole (Figure 1.1b), the two *atrioventricular valves* (the *mitral* valve on left side and the *tricuspid* valve on the right) open and blood moves from the atria to fill the ventricles, which expand. Blood is drawn into the left atrium from the lungs via the *pulmonary veins* and into the right atrium from the body via the *inferior* and *superior venae cavae*. Towards the end of diastole the atria contract to force blood into the ventricles. On the left side, blood moves through the open mitral valve from the left atrium into the left ventricle, whereas on the right side blood moves through the open tricuspid valve from the right atrium into the right ventricle.

During systole (Figure 1.1c), the atrioventricular valves close and the ventricles contract, forcing blood out of the heart through the two *semi-lunar valves* (the *aortic valve* and the *pulmonary valve*). On the left side, blood leaves the left ventricle through the aortic valve into the *aorta*. The ascending aorta exits the heart moving in a *cephalad* direction (towards the head) and arches over at the *aortic arch* to head in a *caudad* direction (towards the 'tail') to the rest of the body. Beyond the aortic arch it is referred to as the *descending aorta*. On the right side, blood is ejected from the right ventricle through the pulmonary valve and enters the *pulmonary artery*, which bifurcates into the *left and right pulmonary arteries* that lead to the left and right lungs.



(a) Anatomy of the fetal heart.



(b) Diastole. **(c)** Systole.

Figure 1.1: Anatomy of the fetal heart and fetal cardiac cycle.

The heart wall consists of the *pericardium*, a fluid-filled sac containing the heart, the *epicardium*, a layer of connective tissue on the exterior of the heart wall, the *myocardium*, the muscular middle layer, and the *endocardium*, which lines the chambers. The *interventricular septum* is the wall separating the left and right ventricles. Similarly, the *atrial septum* separates the two atria.

In cardiovascular systems of adults and children, oxygen exchange occurs in the lungs and the left side of the heart circulates oxygenated blood around the body while the right side of the heart circulates deoxygenated blood to the lungs. By contrast, the fetus is supplied with oxygen from the mother via the placenta and umbilical vein and therefore high blood supply to the fetal lungs is not necessary and the left and right sides of the heart do not need to be separated. Accordingly, there is a gap in the atrial septum connecting the two atria called the *foramen ovale*, which allows blood to pass from the right atrium to the left atrium, thereby bypassing pulmonary circulation. Furthermore, the *ductus arteriosus* (arterial duct) connects the pulmonary artery to the descending aorta and allows most of the blood ejected from the right ventricle to bypass the lungs. In the absence of abnormalities, both the foramen ovale and ductus arteriosus close shortly after birth in order to prevent oxygenated and deoxygenated blood mixing.

The fetal heart rate varies with gestational age, but is generally far higher than adult heart rates and in the range 120 to 160 beats per minute.

1.2 Congenital Heart Disease (CHD)

Congenital heart disease (CHD) is a term used to refer to abnormalities of the heart that are present at birth. This includes a wide variety of defects that affect different areas of the anatomy, and these often co-occur and interact in complex ways. Table 1.1 contains a non-exhaustive list of such abnormalities.

It is estimated that approximately 8 in 1000 live births are affected by some form of CHD [2], and the incidence in the antenatal population is higher due to miscarried and terminated fetuses with CHD. It is therefore one of the most common types of birth defect. It is also a leading cause of infant death, accounting for an

estimated 42% of infant deaths according to the World Health Organisation [3]. Reducing infant mortality is widely recognised as an important global challenge, as demonstrated by its inclusion in the UN Millennium Development Goals [4].

The prognosis and treatment of CHD varies with the nature of the defect. Mild defects may require no treatment and improve naturally over time after birth. However, many defects require surgery or medication shortly after birth. CHD is also associated with a number of developmental problems in early life, in particular restricted physical activity. Furthermore, CHD may cause complications during pregnancy for the mother, especially when she has an existing heart condition. It is therefore highly desirable to detect CHD antenatally such that the condition can be further investigated and monitored and in order to prepare for any interventions that may be necessary. Various researchers have concluded that antenatal diagnosis of a number of heart defects improves outcomes for the fetus [5–9]. In particular, it has been found that antenatal detection of transposition of the great arteries [6] and coarctation (narrowing) of the aorta [9] reduces neonatal mortality.

1.3 Antenatal Ultrasound Screening for CHD

The standard method for antenatal detection of CHD is a 2D ultrasound scan. This is due to a number of factors including: the low cost and high portability of ultrasound technology relative to most other imaging modalities, meaning that it is suitable for high volume screening procedures; the fact that an ultrasound scan does not expose the developing fetus or mother to potentially harmful ionising radiation; and the high temporal resolution (typically in the range 30 Hz to 60 Hz giving at least 10 frames per cardiac cycle), which is necessary to capture abnormalities in the cardiac function.

In many countries, including the United Kingdom, all mothers are given a general screening scan in the middle of the second trimester of pregnancy (at around 18-22 weeks of gestational age) in order to determine gestational age, assess fetal development, and screen for abnormalities (including, but not limited to, cardiac abnormalities). The International Society of Ultrasound in Obstetrics and

Name	Incidence	Description
Partial/Total Anomalous Pulmonary Venous Drainage (PAPVD/TAPVD)	< 1%	One or more of the pulmonary veins drain into somewhere other than the left atrium.
Mitral/Tricuspid Atresia	Rare/1-3%	Obstruction of flow across the mitral/tricuspid valve.
Hypoplastic Left Heart Syndrome	1%	An underdeveloped and functionally inadequate left side of the heart.
Coarctation of the Aorta	8-10%	Narrowing of the aorta.
Ebstein's Anomaly	< 1%	Unusual placement of the tricuspid valve towards the apex of the heart.
Pulmonary Stenosis	5-8%	Obstruction of blood flow from the right ventricle to the pulmonary artery.
Double Inlet Left Ventricle	1%	Both atrioventricular valves open into the left ventricle.
Persistent Truncus Arteriosus	< 1%	Incomplete separation of the pulmonary artery and aorta during development.
Transposition of the Great Arteries	5%	The aorta arises from the right ventricle and the pulmonary artery from the left ventricle.
Ventricular Septal Defect	25%	An opening in the ventricular septum.
Atrial Septal Defect	10%	An anomalous opening in the atrial septum.
Situs Inversus	2%	Mirror-image location of the heart (and other organs), within the abdomen.
Bradycardia/ Tachycardia	–	Unusually slow/fast heart rate.

Table 1.1: Forms of congenital heart disease. Compiled from information in Archer and Manning [2]. Incidence rates are the estimated proportion of CHD cases (NB *not* the proportion of the general population) with the relevant anomaly.

Gynecology (ISUOG) issues guidelines for these mid-trimester screening scans. The guidelines issued in 2011 [10] suggest that the *four-chamber view* of the fetal heart – a view in which all four chambers of the heart and the atrioventricular valves are visible – should be included in all routine mid-trimester scans, and that further *outflow tract views* – showing the pulmonary artery and aorta leaving the ventricles – can be included for 'extended' screenings (these views are defined in more detail in §1.4.4). Where an abnormality is found or suspected, the mother is referred to a fetal cardiac specialist for a more detailed diagnostic scan, referred to as a *fetal echocardiogram* [11].

However, a more recent set of guidelines specific to the heart [3] was issued in 2013 and emphasise the need to include a number of different views of the fetal heart in the routine scan in order to detect defects of the pulmonary artery and aorta such as coarctation of the aorta and double outlet right ventricle.

1.3.1 Factors Limiting the Rates of Detection of CHD

There are a number of important factors limiting the success of screening procedures for CHD, discussed by Chaoui [12].

Many of these factors are a consequence of the fact that screening scans are a highly skilled task for a clinician to perform. During a diagnostic scanning session, a sonographer must simultaneously operate the imaging system, navigate the probe around the 3D anatomy, communicate with the patient, and make the diagnosis of interest. This requires an excellent knowledge of fetal anatomy and the wide range of potential problems that can occur (Table 1.1), some of which have very subtle appearances in ultrasound images and many of which occur so rarely that a non-specialist will rarely or never be exposed to them [12]. However those performing the routine mid-trimester screening scans will typically not be cardiac specialists.

Interpretation of the ultrasound videos is challenging due to the indistinct appearance of structures and boundaries and the presence of imaging artefacts such as speckle, acoustic shadowing, acoustic enhancement, and drop-out artefacts. Navigating the probe's imaging plane around the 3D environment inside the fetus

and finding the correct views is also an acquired skill and requires good physical co-ordination. This is especially true when imaging the fetal heart, which is approximately 2 cm across at 18 weeks gestation [13]. Furthermore, appropriate imaging parameters (such as frequency and magnification) must be chosen in order to acquire useful images.

The *fetal lie* (the position of the fetus in the uterus) can cause complications if the fetus is in an unfavourable position, for example positioned such that the fetal spine casts the heart into an acoustic shadow. Another factor that significantly reduces image quality is maternal obesity, as body fat attenuates the ultrasound beam. In other cases, the defect may simply not be visible on an ultrasound examination, or may not have developed by the time the screening takes place.

Furthermore, routine scans typically occur in time-pressured environments in which there are a large number of other tasks to perform besides screening for CHD. These include fetal biometry measurements, determining the fetus's sex, and checks on a number of other organ systems. Consequently, the sonographer only has a short time in which to concentrate on the heart, acquire the relevant videos and make a diagnosis.

Several studies have concluded that many defects, mostly those involving the vasculature around the heart, are missed because they are not visible from the four-chamber view of the heart [12, 14, 15]. It was this conclusion that led to the recommendation to include outflow tract views in the routine scans [3]. However this increases the level of training needed to perform a scan and also increases time pressure during scanning sessions.

There are therefore a number of reasons why screening scans may fail to detect cardiac defects. Some of these, such as the fetal lie and maternal obesity, are beyond the control of the sonographer performing the scan. However, many are human errors as a result of the huge complexity of the task. Accordingly, a number of studies have concluded that the skill of the sonographer is a key factor in determining the detection rate of CHD and that improved training for sonographers can increase detection rates significantly [12, 16–18]. For example, Pézard et al. [17]

estimated from a long term study in France that detection rates of CHD were 16% for non-specialists, 36% for non-specialists with training, and 90% for experts.

The problem of low detection rates is particularly acute in the developing world, in part due to a lack of trained personnel [19]. In other areas of medicine, studies have suggested that access to skilled sonographers is a more important barrier to healthcare in the developing world than access to equipment [20]. The recent advent of inexpensive and portable ultrasound hardware, some of which can plug into laptop computers or even phones and tablets (such as the GE Vscan Portfolio and Phillips Lumify product ranges) has the potential to widen access to ultrasound devices dramatically for diagnostic purposes, especially in developing world settings. However, unless this is accompanied by a large increase in the number of trained operators, the level of skill required to operate them will still remain a significant barrier to their deployment.

1.4 Aims and Contributions of this Thesis

1.4.1 Research Direction

Given the dependence of detection rates for CHD on the experience and training of sonographers, it is clear that improved training has an important part to play in improving worldwide detection rates. However, with the recent advances in the fields of machine learning and computer vision it is natural to ask whether this problem can also be tackled from another direction, namely by providing automated tools in order to reduce the skill level needed to perform a scan and reduce the chances of human error contributing to a missed diagnosis.

Previous research has successfully applied image analysis and computer vision algorithms to ultrasound images for a number of other medical applications (see §2.2 for a review of this literature). However the majority of these focus on making diagnoses or measurements as a post-processing step that operates on carefully acquired input images (or volumes for 3D ultrasound). This relies upon the skill and experience of the sonographer to capture the correct videos in the first place. Furthermore, this does not fit into the natural workflow for ultrasound screening

scans, where the sonographer simultaneously acquires and analyses the videos to make an immediate diagnosis. In such a setting, the processes of acquiring and analysing the videos cannot be neatly disentangled because an understanding of the videos is required to guide the probe to the relevant anatomy, and suspected anomalies may prompt further investigations which will dictate which images should be acquired next.

Automated tools to assist in diagnosing congenital heart disease should therefore work at the acquisition stage rather than form a separate post-processing stage. In this thesis, the aim is to develop such a tool to assist sonographers in performing effective screening scans for CHD that function during acquisition, i.e. as the scan is being performed. Previous work in this direction has been minimal, and in particular the author knows of no previous work that has specifically focused on automated analysis of fetal heart screening videos (see Chapter 2 for a more detailed review of the literature).

There are conceptually many ways in which such tools could help, including:

- Automatically inferred information about the video stream could be displayed visually to the sonographer to assist them in understanding the videos. For example, important structures could be highlighted on the screen, or the most probable view plane classification could be displayed. This could potentially be useful for helping less experienced sonographers interpret the videos.
- Stored video footage from a scan could be tagged with automatically determined meta-data (for example label of different viewing planes), to enable efficient review of the relevant parts of the footage at a later date.
- Automatically extracted information could be used for quality assurance, for example in determining automatically whether the sonographer has acquired all the required views in a particular session and spent sufficient time on each. Comparing this information across multiple scans could allow hospitals and healthcare providers to perform audits to identify potential weaknesses in their screening programmes.

- An algorithm with full understanding of the cardiac anatomy could suggest instructions on how to move the probe to reach the views of interest.
- These guidance routines could be used to move towards full automation of scans using machine-actuated probes where appropriate.
- Automated processes could check for a wide range of abnormalities and may be able to flag a potential problem that the human sonographer might have missed. Ideally such a system could work without any user input and would automatically determine when the relevant anatomy is visible and where in the video it appears.

A significant research effort would be needed to develop the above tools, and this would extend far beyond the scope of the current thesis. However there are important basic problems that are common to these tasks and must be solved before progressing to solutions for the higher level tasks. Progress towards these goals depends upon a basic level of interpretation of the ultrasound video stream and its relation to the fetal heart anatomy in order to detect the different views of the heart, and specific anatomical structures within those views. This low-level interpretation of the video is the subject of this thesis, with the hope that it will enable and inspire further work towards the goals listed above.

1.4.2 Technical Challenges

There are a number of technical challenges associated with automated interpretation of ultrasound videos of the fetal heart. Figure 1.2 shows some examples of frames from the experimental dataset used in this thesis (see §1.5 for more information) and demonstrates some of these challenges. As described in §1.3.1 the indistinct appearance of anatomical structures in ultrasound videos makes their interpretation difficult for humans and computers alike. This is compounded by variations in contrast and imaging parameters (compare for example Figures 1.2e and 1.2f), as well as the presence of imaging artefacts such as speckle, shadowing and enhancement. Shadowing artefacts are especially problematic as they can result in dark areas of

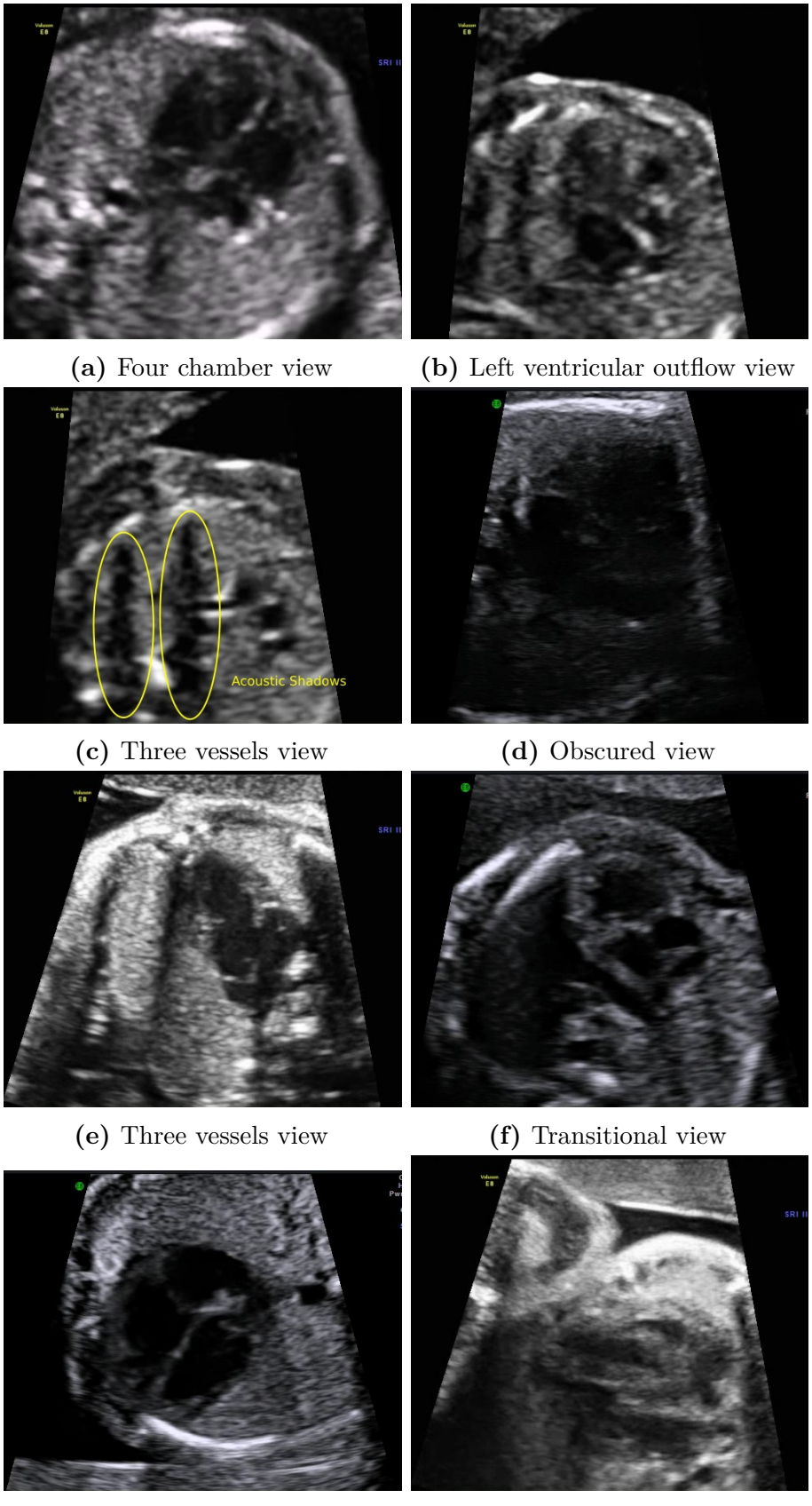
the image that could be mistaken for cardiac structures (e.g. the shadows in the left hand side of Figure 1.2c), or obscure large regions of the image (Figure 1.2d).

In fetal cardiac videos (unlike adult echocardiography), the heart may take up only a small fraction of the ultrasound field of view. Moreover, its location in the image can change due to motion of the probe and/or the fetus during scanning. The orientation of the fetus relative to the direction of the propagation of sound depends on the fetal lie and the probe position, which are unknown and variable during the scan. As a result, the orientation of the heart in the video varies between and within videos (compare, for example, the four chamber views in Figures 1.2a and 1.2g), which poses a particular problem as many image analysis algorithms are designed to work with objects in a particular orientation. Furthermore, a given anatomical view can be obtained in two different mirror image versions by rotating the probe by 180° about the probe's axial direction, meaning there are two mirror image possibilities of the anatomical geometry that should be considered. This can be thought of as looking at the same cross section from above or from below. The appearance and size of the fetal heart changes with gestational age, and the size is also dependent on the magnification factor applied to the video (though this is known to the probe hardware).

The appearance of the heart within a given video changes significantly throughout the cardiac cycle. Additionally, the appearance of the heart is very dependent on the exact location of the probe, as nearby cross-sectional planes can appear very differently even within one standard view of the heart.

Unless the probe is fitted with some sort of position sensor[1], the motion of the probe in the sonographer's hand is unpredictable, but certain reasonable assumptions (such as likely speeds) can be made. This comprises both deliberate motion, for example moving between different views, and small involuntary motions. The mother may make small, unpredictable motions during the scan, and potentially quite large, sudden fetal movements may also be observed.

[1]Incorporating such a sensor would certainly be an interesting and worthwhile research direction, however it was not possible in this work due to equipment limitations.



(a) Four chamber view **(b)** Left ventricular outflow view

(c) Three vessels view **(d)** Obscured view

(e) Three vessels view **(f)** Transitional view

(g) Four chamber view **(h)** Motion blur

Figure 1.2: Example frames from the dataset videos demonstrating the forms of variation in appearance. See text for further details.

When working on unedited data from real scanning sessions there will be many frames that are not relevant to the task at hand or are not useful due to being too difficult to interpret. This may include frames that do not include the fetal heart at all, or that do not match any of the views of the heart described in §1.4.4. For example, the image in Figure 1.2f is captured during a transition between the three vessels and left ventricular outflow tract view. Other frames will be very difficult to interpret due to artefacts or motion. To work well in practice an algorithm must either explicitly or implicitly determine which frames provide useful information for the problem at hand and which do not and should be ignored.

A further important consideration is computational efficiency. Ideally any proposed solution should be able to run quickly enough that it could provide real-time feedback to the sonographer when running on relatively modest computing hardware as found in most ultrasound devices. Portable probes would have even more limited computational resources.

The work in this thesis aims to overcome many of these challenges to enable the creation of systems that could operate reliably in a clinical setting.

1.4.3 Problem Definition

Considering the technical challenges outlined in the previous section, a set of *global* variables was chosen that characterises each video frame (the term *global* is used to distinguish these variables for the entire heart from those specific to individual cardiac structures). This set of variables shall be referred to as the global *state* in line with the systems and control literature and shall be denoted by $\mathbf{s}_t$ at time-step $t \in \mathbb{N}_0$ (corresponding to the frame number in the video). The chosen state consists of the heart's visibility, its centre in the image, its orientation in the image, the view of the heart being observed, the cardiac phase (point in the cardiac cycle), and the cardiac phase rate (heart rate). This characterisation was chosen with the aims of §1.4.1 in mind, i.e. to provide information that could be useful for purposes such as quality assurance and live feedback, and also provide a global coordinate system for higher-level automated processes, such as the diagnosis of specific heart defects.

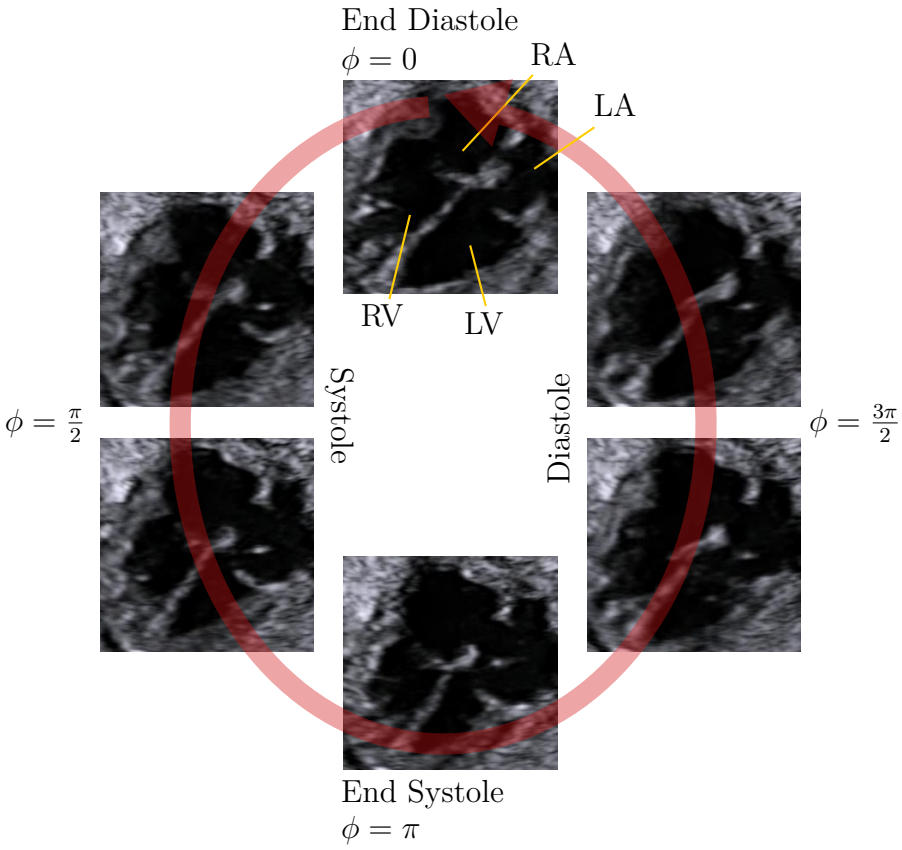


Figure 1.3: Definition of the cardiac phase variable, ϕ. A value of 0 represents end-diastole, a value of π represents end-systole and other values represent other stages of diastole and systole. The example shown is taken from a four-chamber view of the fetal heart.

The *hidden/visible* state variable, $h_t \in \{0, 1\}$ is a binary variable that indicates whether something close to one of the standard views of the heart is visible (represented by a value of 0) or not, in which case the heart is considered to be hidden (a value of 1).

The categorical *view* state variable $v_t \in \{4\text{C}, \text{LVOT}, 3\text{V}\}$ indicates which of the three views is currently visible (see §1.4.4 for a definition of these).

The *position* of the heart centre, $\mathbf{x}_t \in \mathbb{R}^2$, in the image is measured in pixels from the top left of the image and the *orientation*, $\theta_t \in [0, 2\pi)$, of the heart in the image is measured in radians anticlockwise from the image x-axis. $\mathbf{x}_t$ and θ_t have different definitions in each of the viewing planes, see §1.4.4 for details.

The progress of the cardiac cycle is tracked using a second-order model, which reflects the fact that the fetal heart rate is likely to remain fairly constant throughout

the video. The *cardiac phase* variable $\phi_t \in [0, 2\pi)$ tracks the progress of the cardiac cycle as a circular (wrapped) variable in radians. A value of 0 corresponds to end-diastole and a value of π corresponds to end-systole, as shown in Figure 1.3. The *cardiac phase* rate variable, $\dot{\phi}_t$ is the rate of change of the cardiac phase variable with respect to time, in radians per second.

These six state variables collectively form the *state tuple*, which is a collection of variables with heterogeneous data types (binary, discrete, real and circular):

$$\mathbf{s}_t = \left(h_t, v_t, \mathbf{x}_t, \theta_t, \phi_t, \dot{\phi}_t\right) \tag{1.1}$$

The first aim of this thesis is to estimate the value of these state variables from the video data in an 'online' setting, i.e. without access to future video frames, in a manner that is robust to the issues outlined in §1.4.2 and at speeds fast enough to operate on the video as it is captured.

The second aim is to estimate the image locations of a number of anatomical structures of interest. The anatomical structures chosen for this purpose are defined later in §8.1. However, the specific choices of points to track are less important here than the principles behind the method, which could be applied to any anatomical structure of interest for a certain diagnosis.

Three assumptions are made about the videos for the proof-of-concept algorithm described in this thesis. Firstly, all data have been captured in just one of the two possible mirror-image arrangements, corresponding to the case where if the axis of the heart (from base to apex) points to the top of the image, the anatomical left appears on the left of the image and the anatomical right appears on the right of the image. Equivalently, this is the case where the cross-section created by the imaging plane is viewed along the direction from the fetal head to the fetal legs (i.e. looking at the cross-section from above, according the fetus's frame of reference, rather than from below). In this thesis this will be referred to as the 'flip convention'. In practice, both are seen commonly and sonographers' scanning habits vary. Accounting explicitly for different flip conventions would be a relatively minor

extension to the presented work, but was not considered in this thesis in order to simplify the problem and concentrate on the more challenging technical issues.

Secondly, the assumption is made that the approximate size of the heart in the video is known at test time and does not change due to magnification changes during the course of a video. In practice this is not an unreasonable assumption because the heart size is fairly predictable at a known gestational age, and this is typically recorded at the start of a scanning session as a matter of routine for comparison with growth charts during the biometry parts of the mid-trimester scan. Furthermore, the magnification applied to the video will be known to the ultrasound device software. Even when the gestational age and image magnification are known, small natural variations in heart size still occur between patients, but the learning algorithms employed in this thesis should be sufficiently robust to these.

Thirdly, only scans of healthy hearts (with no abnormalities) are considered in this thesis. Whilst an algorithm would need to be robust to the presence of defects in order to be useful in practice, this is a challenging task due to the wide variety of abnormalities that can occur. Consequently, a large amount of abnormal video data would be required to train and validate such models. This thesis therefore concentrates on a proof-of-concept that is validated on healthy subjects, and working with abnormal subjects is left for future work and is discussed further in Chapter 10.

1.4.4 Definition of Fetal Heart Views

The ISUOG guidelines for cardiac screening [3] suggest a total of five axial views of the cardiac heart: the *four-chamber view*, the *left ventricular outflow tract view*, the *right ventricular outflow tract view*, the *three vessels view*, and the *three vessels and trachea view*. These can be found in sequence by locating the four-chamber view and then tilting the probe in a cephalad direction (towards the fetal head). Further views are defined for specific diagnostic purposes, such as sagittal views of the aortic arch, but these are not generally recommended for routine screening scans.

The *right ventricular outflow tract view*, the *three vessel view*, and the *three vessels and trachea view* are all quite similar in composition. Each contains the

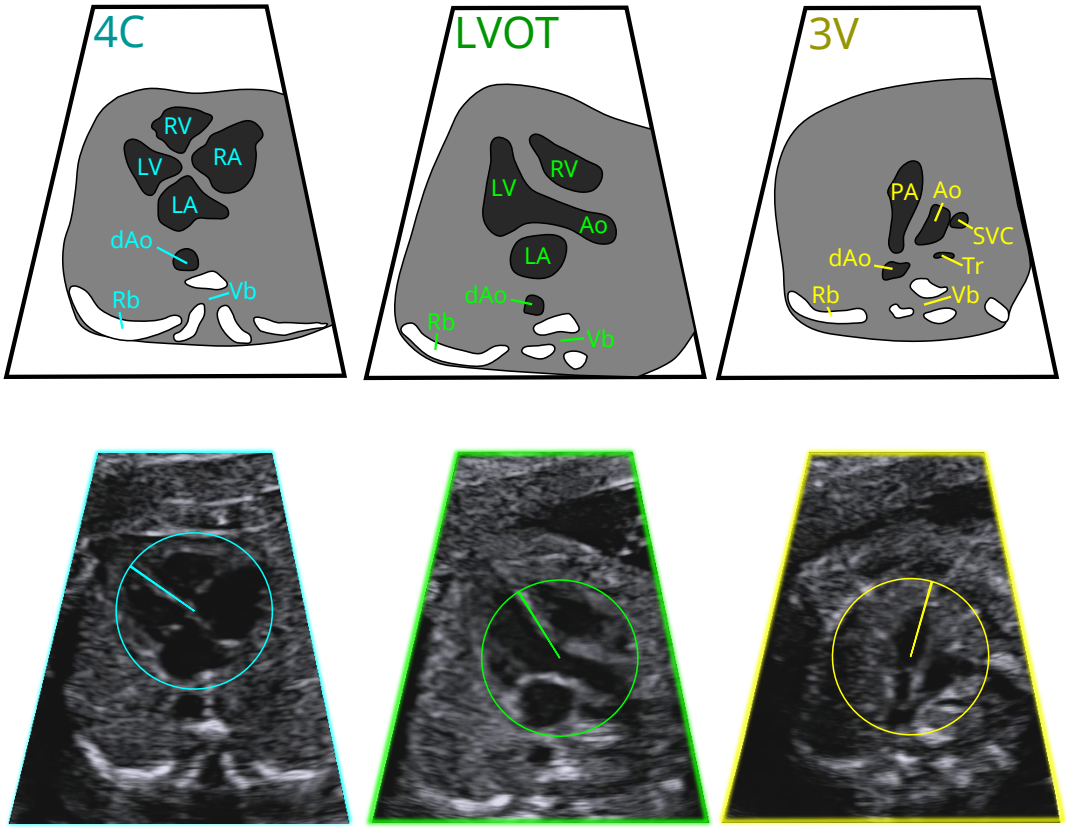


Figure 1.4: Definition of fetal heart scanning views used in this thesis. In each case the centre of the circle represents the heart centre variable, and the radial line represents the orientation variable. The colour scheme in this figure (*cyan* 4C view, *green* LVOT view, *yellow* 3V view) will be used throughout this thesis. Abbreviations used: **LV/RV** left/right ventricle, **LA/RA** left/right atrium, **(d)Ao** (descending) aorta, **PA** pulmonary artery, **SVC** superior vena cava, **Tr** trachea, **Vb** vertebra, **Rb** ribs.

pulmonary artery, the aorta, and the superior vena cava. Due to the availability of data, in this thesis these three views are considered to belong to a single view, which is referred to as the *three vessels view*. Consequently this thesis presents a proof-of-concept validation using a three-view classification that could trivially be extended to the full five-view classification given a suitable annotated dataset.

Figure 1.4 shows the definitions of these three views used throughout this thesis. The definition of each view consists of definitions of the heart centre and orientation that will be tracked by the algorithm. The *four chamber* (4C) view contains all four chambers (left and right atria and ventricles), with the centre at the crux. The orientation is defined by the orientation of the interventricular septum. The

radius is defined in this view as that which encompasses both atria. The left ventricular (aortic) outflow tract (LVOT) view is defined by the presence of the aorta leaving the left ventricle. The centre is defined by the centre of the aorta where it crosses the interventricular septum and the orientation is again defined by that of the interventricular septum. The three vessels view (3V) is defined by the simultaneous presence of the pulmonary artery, aorta and superior vena cava. The centre is defined as the centre of the pulmonary artery at the point where it is in line with the other two vessels, and the orientation is defined by that of the right wall of the pulmonary artery.

1.4.5 Overview of Proposed Solution and Contributions

The solution proposed in this thesis considers the task of estimating values for the state variables at each frame as a *Bayesian filtering* problem that is solved using a particle filtering algorithm (Chapter 6). This provides a principled approach for using a statistical model of the evolution of the state variables that takes measurements in previous frames into account to produce estimates that are consistent between the frames of the video. Within each frame, predictions are made using random forests machine learning models to interpret the image (Chapter 4). To deal with the problem of differing orientations, a method for extracting rotation-invariant features from the images is considered as an alternative to more common features based on rectangular image regions (Chapter 3).

The contributions of this thesis are as follows:

1. The application of rotation-invariant features (RIFs) to detect structures in ultrasound imagery is considered (Chapter 3). An implementation of rotation invariant features that allows them to be extracted by a random forests algorithm far faster than previous implementations and enables processing ultrasound data at tens of frames per second is introduced.

2. Novel adaptations of the random forests algorithm for predicting circular output variables, such as cardiac phase, and for predicting image orientation from rotation-invariant features are introduced (Chapter 4).

3. Partitioned particle filtering architectures are designed for the prediction of the global state variables and the locations of the key structures in 2D ultrasound videos of the fetal heart (Chapters 6 and 8).

4. An experimental evaluation of the above on an annotated clinical dataset is performed, showing that the proposed method is able to estimate the global variables and structure locations with accuracy comparable to inter- and intra-observer variation at high frame rates (Chapters 5, 7 and 9).

1.5 Experimental Dataset

A diverse dataset of 91 short ultrasound videos of the fetal heart was gathered as the results of a collaboration with Dr. Christos Ioannou of the fetal medicine unit at John Radcliffe Hospital, Oxford. Obtaining suitable videos was challenging because of time limtations within screening sessions, and the fact that longer freehand videos (beyond two or three seconds) containing multiple views of the heart are not stored as a matter of clincal routine. Storing such videos can be quite disruptive to a clinical scanning session.

The 91 videos are drawn from 12 subjects during routine clinical scans using a GE Voluson E8 ultrasound device. No videos captured for the purpose of the study were excluded from the dataset. The videos were captured using the probe's video capture function that records the screen exactly as it is presented to the sonographer after the manufacturer's on-board signal processing has been applied (such as demodulation and time-gain compensation). The videos were then cropped spatially to remove all the peripheral information added by the scanner and leave just the ultrasound image itself, with dimensions of approximately 510×430 pixels (± 5 pixels).

Each video had a length of between 2 and 10 seconds and a frame rate between 25 and 76 frames per second (giving several hundred frames per video), and contained one or more of the three views of the fetal heart defined in §1.4.4. The videos captured the healthy fetal heart in a range of magnifications and orientations (see Figure 1.2), though with the heart taking up approximately 30% or more of the

image (as per the scanning guidelines [3]) and without the magnification changing within a given video. There was a range of gestational ages from 20 to 35 weeks. All videos were gathered with the flip convention defined in §1.4.3 or were flipped horizontally for consistency before being used for training and testing.

Each frame of each video was manually annotated by the author with a value for each of the global state variables (except cardiac phase rate, $\dot{\phi}_t$, which is inferred from the labels for cardiac phase, ϕ_t) and the anatomical structures according to the criteria in §1.4.3 and §1.4.4, which were devised in consultation with an experienced clinician (Christos Ioannou). Additionally the approximate size of the heart is also annotated on a per video basis. Annotations for the cardiac phase variables were made by manually selecting each end-diastole and end-systole frame in the video, and linearly interpolating the circular cardiac phase variable value between these key frames. Furthermore, the locations of each structure of interest were also annotated, see §8.1 for more details. The annotation process was performed using custom software tools written by the author, which have been made freely available[2]. This tool displays each frame to the user along with the annoation in a manner similar to images in the lower row of Figure 1.4 and allows them to manipulate the annotation with the keyboard until they are satisfied with it. The next image is then displayed and initialised with the annotation from the previous image. There are further keys used to manually mark the end-systole and endiastole frames.

These annotations are used to provide ground truth labels for training and validating the models and were approved and corrected on a frame-by-frame basis by a clinician experienced in interpreting ultrasound videos of the fetal heart (Christos Ioannou).

The global manual annotations were performed a further two times for one randomly-chosen video from each of the 12 subjects: once by the author again approximately a year after the first set of annotations was completed, and once by another student working on analysis of fetal heart ultrasound videos (Vaanathi Sundaresan) independently in a separate session, but using the same software tool.

[2] `http://github.com/CPBridge/heart_annotation_tool`

These two extra sets were used to provide estimates of the levels of intra-observer and inter-observer variation, respectively, on the annotation task. Although previous studies have attempted to estimate inter- and intra-observer variation in certain tasks related to diagnosis or measurement from ultrasound imagery, the author is not aware of any studies that could be meaningfully compared to these results.

2

Literature Review

Contents

This chapter will review the relevant scientific literature. In the first half of this chapter (§2.1), key current trends within computer vision and machine learning will be briefly described. Then §2.2 will discuss ways in which these techniques have previously been applied to related problems within ultrasound image analysis.

2.1 Relevant Advances In Computer Vision

Since the task of analysing fetal heart ultrasound videos has several facets, there is a wide range of computer vision literature describing approaches to relevant problems. Firstly, several approaches to object detection, one of the fundamental processes within computer vision, will be discussed. Methods for object detection can be broadly broken down into four categories: sliding windows classifiers (§2.1.1), keypoints-based methods (§2.1.2), Hough voting methods (§2.1.3), and convolutional neural networks (§2.1.4).

However, as discussed in Chapter 1, ultrasound imagery is difficult to interpret but contains strong spatial and temporal constraints that can be utilised to disambiguate the content of the video frames. This motivates a discussion of methods of leveraging spatial (§2.1.5) and temporal (§2.1.6, §2.1.7) information within images and videos later in this section.

2.1.1 Sliding Window Methods

The sliding window framework is one of the simplest and most popular paradigms for object detection. It is based upon performing a binary classification in each sub-image ('window') for the presence or absence of an object within that sub-image using some set of features extracted from the sub-image. The same detector is applied to a number of windows in an image with different locations and often also at different scales.

Much of the work on sliding window classifiers in the early 2000s built on the algorithm of Viola and Jones [21] for detecting faces. Their algorithm created features using very simple 'rectangle' filters (also known as 'Haar' or 'Haar-like' filters), consisting of summed intensity values under rectangles with various spatial arrangements. The crucial development in Viola and Jones' work was selecting these features with an Adaboost [22] learning methodology to create a *cascade* of filters designed to reject many negative windows early on, meaning that the features used in later stages only needed to be evaluated on a small number of locations in the input image. The use of integral images also allowed the rectangle filter outputs

to be evaluated extremely quickly, and thus frame-rate detection was possible. The disadvantage of the method is its long training time, as a large number of features have to be evaluated using trial and error. As computing hardware has advanced, however, this has become increasingly less important.

In a separate development, Dalal and Triggs [23] introduced Histogram of Oriented Gradients (HOG) features for sliding window object detection. Firstly, the intensity gradient of the input image is found using standard methods. The gradient vectors are then put into orientation bins according to their direction and weighted by their magnitude. The gradient vectors are also put into spatial bins ('cells') in different parts of the sub-image, which are then concatenated into 'blocks'. The blocks descriptors are then normalised to increase robustness to contrast before being finally concatenated into a feature vector for classification. In the original work, Dalal and Triggs made use of Support Vector Machines (SVMs) with a linear kernel. The method has proved popular due to its robustness to changes of contrast, absolute intensity level and the exact location of edges.

Various authors have since successfully used different variations and extensions of orientated gradient histograms, for example Maji et al. [24] used histograms of orientated edge energy (obtained using an orientated filter bank) combined into blocks at various scales, and then used an efficient implementation of the histogram intersection kernel SVM for improved performance over linear SVMs whilst avoiding the slow test times normally associated with non-linear kernels.

Others have subsequently incorporated elements of the HOG feature extraction method into cascade classifiers to increase flexibility and speed. Unlike Dalal and Triggs' formulation, in which the histogram blocks are rigidly defined, such methods allowed blocks to take a variety of shapes and sizes, whilst using integral histograms to retain high efficiency [25, 26]. Others have combined multiple cues, for example Dollár et al. [27] and Benenson et al. [28] extended rectangular filters to operate on (and indeed between) multiple feature channels (in their case edge detectors, Gabor filters, colour channels and gradient orientation histograms), whilst Wojek et al. [29] combined HOG, rectangle filters and orientated motion

histogram features using both Adaboost variants and linear SVM classifiers. Such an approach can combine a large number of possible cues, for example edge, texture, gradient, intensity and colour information (whilst obviously increasing computational demand, particularly during training).

Some work has instead focused on the orthogonal task of altering the cascade architecture to improve the speed and accuracy of detection. Bourdev and Brandt [30, 31] introduced Soft Cascades, in which each stage produced a real-valued rather than binary-valued output and the result of each stage took into account the response of previous stages. These improvements brought increased accuracy and allowed the accuracy-speed trade-off to be controlled without retraining the classifier, as would be necessary for a standard cascade. Speed improvements were achieved by introducing fast approximations to multi-scale features [32] or conversely approximating the classifiers at multiple scales from classifiers trained at just a subset of scales [28], and by allowing cascades to excite or inhibit those on neighbouring image regions [33].

Others [34] replaced the boosted learning method with the random forests model [35–37], which uses an ensemble of binary decision trees trained with randomised training procedures to classify patches as object or non-object.

The disadvantages of sliding window methods are their limited ability to model objects with large shape variations including different sizes and orientations of objects. Furthermore, they are inherently 'local' detectors, meaning that they do not capture contextual information, which may be important in some contexts (or a distracting in others). For further references and performance comparisons on object detection as applied to pedestrian detection the interested reader is directed to a recent review by Dollár et al. [38].

2.1.2 Keypoint Methods

Though sliding window methods were perhaps the most prolific in object detection algorithms before the rise of deep learning methods, another common framework is based on detecting and describing features at sparse keypoints. The general idea is as follows: at training time interest points are detected using some standard

interest point detector (e.g. Harris [39], SIFT [40] or SURF [41]). The patches around detected points are represented by a feature vector (e.g. using the SIFT descriptor [42] or principal component analysis (PCA) on the patch intensity values [43]). These descriptors are then used to create a 'codebook' of so-called visual words. At test time, the keypoints are detected in the same way and matched to their codebook entries using the descriptor. Then the histogram of codebook entries in the image is taken as a descriptor of the image to be used for object detection. This is often referred to as the 'bag of visual words' or the 'bag of keypoints' paradigm.

Because interest points will be detected at points all over the image, this approach naturally lends itself to the categorisation of the entire image, rather than the localised detection of objects within it (e.g. the work of Csurka et al. [44] and Sivic et al. [45]). In many cases such models have ignored any prior knowledge about the likely spatial arrangements of features, and do not provide an estimate of the location or number of objects. However, many similar models do include such reasoning into the model, for a discussion of these see §2.1.5.

Keypoint models have shown good performance for detecting objects that have distinctive, reliably detected sets of keypoints even where there may be large intra-class variation of overall shape; for example faces with eyes and nose, or cars with corners, headlights and wheels etc. They can often handle partial occlusion as they do not require all the keypoints to trigger a detection. However, the nature of ultrasound imagery means that there are a lack of obvious corners and other distinctive isolated features. Keypoint based approaches are therefore unlikely to be so successful for fetal heart ultrasound image analysis, as they depend upon reliably detecting distinctive feature. This can potentially be overcome by densely sampling interest points, though of course this increases the computation time required and produces a lot of irrelevant information.

2.1.3 Hough Voting Methods

Inspired by the classical Hough transform [46] to detect lines and curves in an image, the generalised Hough transform [47] is a method to detect arbitrary shapes by

accumulating votes for the location of the object in a "Hough" voting space when all edge pixels vote for the location of the object centre. Gall et al. [48] proposed Hough forests, in which the random forests algorithm ([35] see §4 for more details) is used to determine whether an image patch belongs to an object and, if it is, its vote for location of the centre of the object is determined using random forest regression. Each patch in the image is passed into the random forest, and the votes from the positively classified patches are accumulated in a Hough voting array. Simple post-processing of the Hough array leads to the detection of one or more objects.

Several further methods based on Hough forests will be described in §2.1.5.

2.1.4 Convolutional Neural Networks

Over the past 3-5 years, and during the time that the work in this thesis was being performed, the field of computer vision has been dominated by advances in convolutional neural networks (CNNs) [49–53]. A CNN is formed of several stacked layers of artificial *neurons* (often referred to as *units*). Like in general artificial neural networks (ANNs) [54, 55], each unit takes as input the outputs of some or all of the neurons in the previous layer, scales them according to a set of learned weights, and passes the sum through some non-linear *activation function* to create the output, which is passed on to the next layer. The defining characteristic of *convolutional* neural networks is that each unit is connected to only those units in the previous layer that correspond to a certain spatial area, and the input to the first layer is directly connected to the raw image pixels. Furthermore, the learned weight parameters are shared between corresponding neurons for different spatial regions, and hence each layer can be considered to perform a convolution operation (followed by the activation function) with a learned kernel on the previous layer. This both dramatically reduces the number of parameters to learn, relative to a fully connected network (a network in which each unit is connected *all* units in the previous layer), and provides a degree of translation invariance. Consequently, the layers of the network learn increasingly abstract representations of the input image whilst gradually aggregating information from larger and larger spatial areas.

This makes the representations used by CNNs far more flexible than "handcrafted" representations (such as HOG), and allows the entire feature extraction process to be optimised "end-to-end" for the particular task at hand, with no hard differentiation made between the feature extraction process and the predictive model. Learning the weights is performed using stochastic gradient descent, where the gradient of the output cost function with respect to the weights may be found by back-propagating from the output towards the inputs. Key current downsides of CNNs are that they require a large amount amount of labelled training data, and are computationally very intensive to train, meaning that highly-parallelised graphics processor units (GPUs) are often required to train in a reasonable length of time.

Since their initial success, researchers have improved the performance of CNNs by exploring different architectures (arrangements of layers) [51–53] and other tricks to improve the training process such as different activation functions [56], random dropout of connections during training to reduce overfitting [57], and transfering features from pre-trained models [58]. Whilst CNNs are well suited to image level classification, alterations must be made to allow location of objects within the image. The region CNN (R-CNN) architecture [59] and its variants [60, 61] propose regions of interest within images and then feed warped versions of these into a CNN for classification of the presence and class of object. CNNs have also been successfully applied to other problems such as segmentation [62] and regression (see §2.1.5).

2.1.5 Spatial Models and Pose Estimation

The requirement to simultaneously predict the locations of several structures with interdependent locations motivates a review of how spatial models have previously been applied to predicting multiple landmarks in images. In the computer vision literature, the majority of work in this direction has been conducted in two contexts: human pose estimation, and facial landmark localisation. Additionally (and with notable exceptions) most of the approaches for these two tasks have been focused on three key algorithms: constellation models, pictorial structures model, and the random Hough forest model. Each of these is reviewed below.

Constellation models [43, 63–66] are one especially popular class of model. They are generative models for objects that factor the likelihood into terms involving part appearance, layout/shape and sometimes scale. First, potential parts are detected at interest points, and represented through vector quantisation of some descriptor (either with PCA on the image intensity [43, 63] or gradient [64], or k-means clustering [65] on patches). Then possible hypotheses of part identity are evaluated using a layout model, which in earlier work was jointly Gaussian [43, 63, 65], but later work switched to a star-structured graphical model for reasons of computational efficiency [64]. Crandall et al. used graphical models with more complicated structures that are still simpler than a joint Gaussian (essentially using multiple root nodes), but found that the increase in performance is small at the expense of increased computation [66]. An advantage of these models is that they can work in a semi-supervised or unsupervised setting using the expectation-maximisation (EM) algorithm [65] or the more involved latent semantic analysis [45]. However, this is not a relevant advantage for the application in this thesis as there is expert-annotated training data to learn from. Furthermore, the focus is on using parts as a means of robustly locating objects, rather than having accurate part locations as an end in itself, and the performance depends on a reliable interest point detection stage, which is particularly difficult in ultrasound.

Following earlier work by Fischler and Elschlager [67], Felzenszwalb and Huttenlocher [68] introduced a general framework for part-based models called *pictorial structures*. The basis of the method is an undirected graphical model, where the (discrete-valued) locations of each part are modelled as nodes in a star-structured graph. An energy function is defined on this graph in terms of two factors: the local appearance at each node (node potentials) and the spatial relationships between the locations of connected pairs of nodes (edge potentials). This is fundamentally different from the constellation approach, where an unknown number of parts are first detected and used to reason about the presence of an object; by contrast in a pictorial structure the part identities are known *a priori* and the model is used to reason about their positions before deciding upon the presence of the whole object.

The key contribution of Felzenszwalb and Huttenlocher [68] was to note that if the structure of the graphical model is a tree (i.e. contains no cycles) and the edge potentials have a Gaussian form, then the globally optimal match of the locations to the image can be found efficiently using a form of dynamic programming based on min-convolutions. In later work [26], Felzenszwalb et al. demonstrated an effective implementation (the *deformable parts model*) using HOG-like features as a local appearance model and simple quadratic penalty terms for the locations of parts relative to the root node, and improved efficiency using a cascade architecture in order to avoid evaluating unnecessary node appearance functions [69].

Following this work, a number of authors have made successful instantiations of the general pictorial structures framework [70, 71]. Yang and Ramanan [70] generalised the deformable parts model for human pose estimation by using mixture models of appearance for each part, in order to model for example, an open or closed fist, or a face from different viewpoints. Zhu and Ramanan [71] used a similar formulation for facial landmark detection, with a mixture of different tree structures with shared parts to model the face from different angles. This approach therefore allows the spatial relationships between the parts to be image dependent.

Hough forests (§2.1.3) have proved a very popular method for landmark detection and pose estimation [72–79]. In these methods, each image patch votes for the locations of multiple parts. Thus, while the spatial relationships between the different parts are not explicitly modelled, they arise implicitly as a consequence of the learned voting patterns. Consequently, there is no need to manually specify a graphical model and construct local part detectors. This approach is very flexible and works well in practice, as well as being very computationally efficient. Another key advantage is that since any informative patch in the image can vote for the landmark locations, the contextual information captured by the model is not limited to the locations of the specified landmarks as it is in models with local detectors.

Several authors have developed this idea in order to make the technique applicable in a wider variety of situations by conditioning the Hough forests on some global or latent variable, for example face/torso orientation or size. This latent variable

can be estimated either by the forest itself [77–79] or as a pre-processing step [74], and can operate by either selecting a subset of trees in the forest to vote [74, 78], influencing the values of the votes from each tree [78], or weighting the votes [77] according to the value of the latent variable.

Others have tried to enforce some spatial consistency amongst the landmark positions. Yang et al. [75] 'sieved' votes using a latent variable – only patches which give consistent votes for the face centre are allowed to vote for the position of the specific facial landmarks. Jia et al. [76] instead enforced spatial consistency of the face/non-face labels (that allow or inhibit voting for landmark positions) by using a structured output space that predicts the labels of a patch of pixels together.

Dantone [80] combined the Hough forest method with the pictorial structures model by using the Hough forest output to provide the node potential for the pictorial structures model, thus gaining the advantages of both methods to perform the challenging task of human pose estimation from single images. Furthermore, they build context directly into the Hough forest model by using a two stage model, where the second stage can use the intermediate output of the first. This allows the model to overcome certain ambiguities such as distinguishing the left and right legs.

Recently, there has been some success applying deep learning techniques to human pose detection [81–84], facial landmark detection [85], and hand pose detection [86]. This is achieved by altering the cost function of a CNN to simultaneously predict the locations of joints or landmarks of interest. Thus, the spatial connections between the different objects are not explicitly modelled, but rather implicitly encoded in the higher level features of the network. Typically however, due to the global nature of the high level features, the accuracy of this regression is not that high, and it is necessary to use further, local networks – operating at a higher image resolution on a small image patch – to refine the prediction from the initial network [81, 85]. Oberweger et al. [86] showed that it was advantageous to explicitly force the network to form a low-dimensional representation of the pose space, and thereby capture the spatial constraints between the different landmarks, by using a "bottleneck" layer, with a lower dimension than the output layer, before

the output layer. Newell, Yang and Deng [83] expanded on this idea by creating *hourglass* networks that can be thought of as creating several stacked bottlenecks that allows the network that alternate several times between performing inference at the global and local levels, leading to several sets of predictions that are refined through the network. An alternative method for refining and combining pose estimates for the different landmarks is to use a recurrent layer (see §2.1.7) at the end of the network [84]. Gkioxari et al. [82] showed that a pose estimator could be trained alongside a person detector and an action classifier within the same network using multiple loss functions.

2.1.6 Video Analysis

Relevant literature on video analysis can broadly be divided into two groups. The first group is interested in modelling the evolution of the video frames themselves over time in order to identify sequences with certain dynamics. The second models the evolution of some underlying variables of interest over time, rather than the image pixels themselves, in order to infer the values of the variables from an image sequence.

A number of authors have proposed modelling the progression of a video sequence as the output of a linear dynamical system (LDS) [87–89], creating the *dynamic texture* (DT) model. This model, introduced by Doretto et al. [87], assumes that there is a hidden, low-dimensional *state* representation of the video, which evolves over time according to a linear update rule with random perturbations. The observed frame at each timestep is assumed to be related to this hidden state vector at that timestep by a second linear transformation (plus random noise). Given a video sequence, the parameters of the dynamic texture model may be fitted (sub-optimally) using a method based on the singular value decomposition (SVD). As well as providing a generative model for synthesising further similar video sequences, the model parameters from a given sequence my be compared to the parameters from other sequences to provide a way to compare video sequences, cluster similar sequences, and build nearest-neighbour or kernel-based classifiers.

Chan and Vasconcelos [88] extended the DT framework by allowing the frame pixels to be related to the state vector by a more general non-linear mapping learnt using kernel principal component analysis (PCA), giving the *kernel dynamic texture* (KDT) model. They then used the Martin distance between two sets of model parameters to model to construct classifiers based on nearest-neighbour or support vector machines (SVMs), and were able to classify different "video textures" such as moving water, flickering fire, and swaying foliage. A further notable development on this was due to Chaudhry et al. [89], who modelled the evolution of oriented optical flow histograms from the frames rather than the raw pixels themselves through the use of special kernels. With this approach, they were able to classify sequences of human motion according to the action being performed.

Like the work in this thesis, the dynamic texture and kernel dynamic texture models relate the evolution of the video frames to an underlying state space representation. However the emphasis in these models is on *learning* any low-dimensional representation that best explains the observed data, rather than a representation that is useful for any other purpose. Consequently, no particular physical meaning can be ascribed to the value of the state vector at any given time. By contrast the aim in this thesis is to track a state vector consisting of *specified* variables of interest (position, orientation, view plane etc.), and learn how to infer this information from the observed video data. The DT and KDT models are also computationally demanding and operate on full sequences of video at once, meaning that they are not well-suited to real time video processing applications.

Another important goal within video analysis is *tracking* of objects of interest. There are a wide variety of tracking algorithms with a number of different approaches to the task of representing the object, extracting image features, and modelling object dynamics [90], and it is beyond the scope of this thesis to survey them all. Broadly speaking there is a spectrum between tracking and detection [91], where the former has a purely local appearance model and therefore finds similar patches in nearby frames (for example using mean-shift [92] or registration type approaches [93]) and the latter where there is a more general appearance model for the whole object

class. Typically pure tracking approaches require a manually specified initialisation, and are therefore not well-suited to building fully automated tools. The reader is referred to the survey of Yilmaz et al. [90] for a comprehensive discussion.

A number of detection-based tracking algorithms adopt a *probabilistic* approach in which a probabilistic estimate of the object location is updated over time using a *recursive Bayesian filter*, which is a very general framework for probabilistic tracking of a dynamic system including as special cases the Kalman filter and the particle filter [94]. This is also suitable for tracking high dimensional descriptions of objects such as the object boundary [94]. More detail of the theory of recursive Bayesian filters is found in Chapter 6.

The problem of tracking/detecting multiple structures with both temporal and spatial constraints is considerably more challenging. Whilst graphical models such as recursive Bayesian filters and pictorial structures are popular for incorporating either spatial or temporal constraints, connecting different structures in both space and time creates graphical structures containing cycles. This makes exact inference impossible. Approximate inference may be performed with particle-based methods such as particle message passing [95] and non-parametric belief propagation [96], however these are very slow due to the need to multiply probability distributions represented by particle sets.

Sigal [97, 98] and Sudderth [99] used this approach to track human motion and hand motion respectively, defined in both cases by landmark points. Both were able to achieve high quality tracking by fully exploiting both sets of constraints, however in both cases the running time was extremely slow at several minutes per frame, making such models entirely unsuitable for real time applications.

2.1.7 Recurrent Neural Networks

Recurrent neural networks (RNNs) are artificial neural networks in which connections between the units can form cycles such that the hidden states become time-dependent [100]. This is useful when analysing sequential data (such as natural language or video data [101]) as it allows the network to analyse sequences of

arbitrary length while retaining 'memories' of previous inputs. At the time of writing, the application of RNNs to video data is in its infancy, but there is promise for substantial progress in this area over the coming years.

Notable early work in this direction includes that of Fragkiadaki et al. [102], who used an Encoder-Recurrent-Decoder (ERD) architecture to estimate human pose from an input video. This architecture uses a CNN to create a high-level, low-dimensional representation of each frame (the Encoder part). This is then used as the input to an RNN (the Recurrent part), which learns the dynamics of human movement in this high-level space. Finally, there is a further fully-connected neural network model (the Decoder part) that transforms the recurrent representation into the output of interest (in this case joint locations). As well as being highly flexible and possessing considerable representational power, the model has the crucial advantage of being fully end-to-end trainable for the required task.

Another promising application of RNNs is for tracking a complex *state*. For example, Ondrúška and Posner [103] train an RNN to track multiple objects in a cluttered scene using incomplete sensor measurements.

Future work following on from this thesis could consider the use of RNN based methods.

2.2 Ultrasound Image Analysis

2.2.1 Structure Detection in Fetal Ultrasound Imagery

There has been some previous success in applying techniques from the broader object detection literature to the domain of ultrasound imaging. Most of these are variations on the rectangular filters and Adaboost cascade methodology of Viola and Jones [21]. For example, Rahmatullah et al. [104, 105] used rectangular features with Adaboost to detect abdominal landmarks in still images, and detect standard planes in volumetric ultrasound data, achieving reasonable accuracy in both tasks. However, their method was slow because they did not make use of the cascade detector architecture. In later work, they added an earlier stage that used a feature symmetry map [106] and connected component analysis to detect a shortlist of

likely locations for the anatomical landmarks [107]. In this way the detector could be made more efficient. Georgescu et al. [108] and Karavides et al. [109] made use of 2D and 3D rectangular (cuboid) filters respectively in detecting and segmenting ventricles in ultrasound images/volumes of the adult heart, while Zhou et al. [110] used 2D rectangular features for classification of echocardiographic views.

Carneiro et al. [111, 112] also made use of Viola and Jones' rectangular filters to train detectors for a number of different anatomical structures. They used Tu's Probabilistic Boosting Tree (PBT) architecture [113], which can be considered a generalisation of the cascade architecture in which each node in a decision tree contains a strong classifier trained with Adaboost. Their model was later adopted by a number of other works on ultrasound detection by the same group [114–117]. This architecture is no longer focused on early rejection of a large number of false negatives, but, like the more general decision tree framework, allows a more natural extension to multi-class classification problems.

Namburete et al. [118] found that improved results can be obtained by first transforming the input image to statistical images to reduce the effect of speckle artefacts. This was done by modelling the intensity distributions of speckle using a Nakagami distribution [119, 120]. Each pixel was replaced by either the shape or scale parameter of the Nakagami probability distribution fitted to the pixel intensities in a window centred on the original pixel. Using rectangular features and an Adaboost detector framework on these statistical images, they achieved good results for the task of detecting the choroid plexus in 2D ultrasound images of the fetal brain.

More recently, the random forests algorithm [35, 36] has also proved very effective in constructing features in ultrasound images. Ni et al. [121] used random forests with rectangular features for detection of structures in the fetal abdomen. Yaqub et al. [122] also used random forests with rectangular features to locate and classify a range of anatomical structures within fetal ultrasound images. Their method first proposes regions using the normalised cross-correlation measure between the image and a template to propose regions that might contain structures, which are then passed to the random forest for classification. Random forests have also been used

for many segmentation-based detection methods for fetal brain structures [123], fetal femurs [124], or the myocardium in adult echocardiography [125].

Despite the potentially limited applicability of interest point detectors to ultrasound, keypoint approaches have been successfully been used in ultrasound for detection of structures at the whole image level. Maraci et al [126, 127] detected images containing different fetal structures by calculated dense SIFT descriptors from the image and aggregating them in the bag-of-visual-words (BoVW) [126] or Fisher Vector [127] encodings for classification.

2.2.2 Spatial Models in Ultrasound Images

Some previous work has attempted to leverage spatial context information to aid detection in ultrasound imagery [112, 114–117, 121, 128].

Some authors have hand-crafted contextual rules to govern the relative positions of structures. Kumar et al. [128] detect the fetal abdomen, spine and stomach in that order using hand-specified rules about their relative placement, whereas Ni et al. [121] detect the umbilical vein, spine, and stomach bubble by constraining them to be in certain radial segments of the fetal abdomen. Whilst these approaches might work in simple cases, it is not always practical (much less optimal) to hand-specify such rules.

Carneiro et al. [112] incorporated context into their detection system by sequentially detecting different structures and using the location of the previous structures to limit the search space of the detectors for subsequent structures. This was able to increase the efficiency of detections. Sofka et al. [114] took this further in their hierarchical detection network, which detects the locations of a set list of objects in 2D and 3D images sequentially, using the locations of previously detected objects to inform the detection of further objects. The inference required is intractable and so Monte Carlo methods were used, as adapted from sequential estimation techniques in the tracking literature. In this case, the sequence was that of detection of objects in a single image rather than the detection of the same object in a sequence of images as is found in tracking algorithms. For each object there was therefore a prediction step (based on one previously detected object's location)

and an update step (based on observations from a PBT classifier [111]). They used a greedy method to choose the best selection order for the different objects based on training data. The authors of these works and their collaborators have successfully demonstrated the usefulness of the approach for detection of landmarks in 2D ultrasound images of the adult heart [114], detection of structures in 3D images of the fetal brain [114–116], and the detection of the nuchal translucency in fetal images [117]. This algorithm is designed to work with sets of features that are known to all be present in the image in question, and as such they would need to be generalised to be applicable to more general problems of images with unknown content. Furthermore, using sequential detection is always vulnerable to a bad detection early in the process, unlike methods based on inference on a graph.

2.2.3 View Detection in Fetal and Adult Echocardiography

While there is little previous literature on view detection in *fetal* heart images, the problem of view detection in adult echocardiography is better studied. However, there are a number of differences between adult and fetal cardiac imagery. The small size of the heart during the second trimester, and the need to scan through the mother's abdomen means that the image quality in fetal scans is typically considerably worse than in adult chocardiography. Furthermore, the position of the fetus within the uterus is unknown and variable during scanning, meaning that it is more difficult to locate the standard scanning planes. Consequently, the layout of the images, including the size, location and orientation of the heart within the image, can vary signignificantly between fetal cardiac scans, in contrast to adult scans in which these things are relatively consistent. Furthermore, the list of views typically used in fetal cardiac scanning is typically different from the standard views used in adult echocardiography. This is both because certain views are difficult to obtain *in utero*, and because fetal scans are typically screening for a broader range of potential anomalies.

Several approaches to view detection in adult echocardiography make use of global image properties in order to deduce the view label. For example, Agarwal

et al. [129] use a histogram of oriented gradients (HOG) descriptor on the whole image, broken into four non-overlapping blocks. This can distinguish between two very different views (long axis and short axis) with a support vector machine (SVM) classifier. Wu et al. [130] employ a similar method, using 'GIST' descriptors [131] in 16 image blocks instead of HOG descriptors. Zhou et al. [110] use a multi-class classifier based on LogitBoost and rectangular filters ('Haar-like' filters) in order to distinguish between apical two-chamber and four-chamber views. Such global methods are not well-suited to fetal echocardiography because they assume a relatively consistent layout of frames, but in fetal imagery the position and orientation of the heart is unknown. Also, in the application in this thesis, only small areas of the fetal images are relevant to view classification, and the rest of the image is taken up by the fetal abdomen and the womb.

This is overcome, to some extent, in the work of Park et al. [132], which builds on the work of Zhou et al. [110] by adding a left ventricle detection stage, which is then used to position the multi-class view classifier in the image. However, this relies upon the appearance of the left ventricle being fairly consistent between views, and there is unfortunately no such guarantee of consistency in the fetal views of interest to us. Furthermore, although it solves the problem of unknown position it does not solve the problem of unknown orientation.

Other methods rely on first detecting keypoints in the frame. Qian et al. [133] detect space-time interest points in the video stream and describe them using a 3D scale-invariant feature transform (SIFT) descriptor (in the two spatial dimensions plus time). Similarly, Kumar et al. [134] detect interest points using the SIFT keypoint detector in the motion magnitude image, and describe them using local histograms of motion magnitude and intensity. In both cases, the extracted descriptors are quantised according to a pre-trained codebook, and an SVM classifier is used on the codebook histogram for classification. Such approaches are also unlikely to be effective in fetal imagery for the same reasons as the global methods. It is also difficult to estimate other information such as position, orientation and cardiac phase information from the frames using this approach.

Ebadollahi et al. [135] first use the grey-scale symmetric axis transform (GSAT) to detect the "blobs" that are potential heart chambers. They then connect them in a Markov Random Field (MRF) graph structure in order to label the chambers and hence deduce the view label. This approach depends on reliable detection of chambers, and the results showed that accuracy dropped dramatically when chamber detection was not reliable, as is likely to be the case in fetal imaging where structures other than the heart are visible.

A semi-automatic system detecting viewing planes in volumetric *fetal* heart videos has been demonstrated by Yeo and Romero [136]. This system takes a very different approach to this thesis by working as a *post-processing* step on *volumetric* imagery. The user must manually locate seven anatomical points in the 3D image, which is time-consuming and subjective. The system then infers the location of 9 diagnostic planes of interest within the volume, and provides a visualisation assistance tool to view these with small displacements to allow users to look for potential anomalies.

2.2.4 Ultrasound Video Analysis

Despite the inherently temporal nature of 2D ultrasound acquisition, application of video analysis techniques to ultrasound videos in the literature is limited to a small number of cases.

There has been some success using the kernel dynamic textures (KDT) group of models for detecting structures in ultrasound [137–140] in a way that takes both the image appearance and local video dynamics into account. Kwitt et al. [137, 138] used KDT models based to model the evolution of either raw pixel intensities or bag-of-words histograms through the video sequence. The model parameters were found for each short video sequence within a temporal sliding window, and these were used within a classification framework to determine which temporal window contained the structure of interest. However, they only validated their model on a phantom study, not real-world clinical data.

Maraci et al. [139] developed a similar approach in parallel and applied it to real world ultrasound datasets in order to detect when the fetal head appeared with a fetal scanning video. They performed some image pre-processing using feature symmetry [106, 141] to pick out the structures of interest in the video frames, and used an improved kernel (the Binet-Cauchy kernel) for the classification stage. In later work [140] (to which the author of this thesis contributed), this approach was applied to detecting the fetal heartbeat to ascertain fetal viability in a binary classifcation. In this latter application, the advantage of considering the temporal dynamics is clear. Whilst these models provide an elegant way to make use of temporal dynamics, they are able to locate structures in time only, not in space. In principle they could be applied to different windows in both time and space in order to give spatial localisation, however this would be prohibitively slow.

2.2.5 Boundary Tracking in Adult Echocardiography

Another area of related work is automatic boundary tracking in (adult) echocardiography using 2D [142–144] or 3D [145, 146] video data. Like the work in this thesis, these algorithms track a high-dimensional representation of the heart as it evolves through video frames, and they tend to use a strong temporal prior model in order to provide robustness to ambiguous image information. For example, in early work Jacob et al. [142] used a Kalman filter to model the evolution of the left ventricular boundary. Nascimento and Marques [143] built on this with multiple predictive models and robust data association to eliminate erroneous boundary candidates. Carneiro and Nascimento [144] track points on the left ventricle endocardium using a robust particle filtering framework that couples a linear transition model (in fact one model for diastole and another for systole) with an observation model built with deep neural networks. Such techniques are also applicable for the higher dimensional problem of 3D boundary tracking, such as the work of Yang et al. [145], which uses a prediction model based on manifold learning of left ventricle boundary trajectories, and combines it with an observation model using probabilistic boosting trees.

Whilst the methodologies in these papers are related to those used in this thesis, their aims are somewhat different as they specifically aim to track the ventricle boundary, and assume carefully captured data that reliably contains the boundary of interest and in which there are no changes in viewing plane or significant changes in heart location. By contrast the aim of this thesis is to provide a more broadly applicable set of measurements and descriptions of fetal heart scans, that could provide useful information in less constrained scanning sessions.

2.2.6 Speckle Tracking

Speckle tracking [147–149] is a technique used to produce accurate estimates of tissue strain and motion in echocardiography. The underlying principle is that the speckle patterns produced by soft tissues in ultrasound images remain relatively stable as the tissue deforms. Therefore the speckle pattern can be used as a 'fingerprint' that can be used to track an anatomical point as the tissue deforms over the cardiac cycle. Tracking is typically performed using a patch-based correlation technique using the sum-of-absolute-differences [147, 148] to quantify the similarity between patches in consecutive frames. Tracking a number of points in this way can be used to characterise heart function by deriving quantities such as *rotation*, *twist* and *torsional gradient* [149] and thereby assess cardiac abnormalities.

Although speckle tracking has mostly been used in adult echocardiography, it has also proved a useful tool in fetal imaging [150]. However its applicability to this thesis is limited as it requires a well-constrained and high-quality video stream on which to operate, i.e. one with minimal out-of-plane motion and with a number of pre-defined points (manually or semi-automatically) of interest to track. The level of detail it provides is also not necessary for the basic assessment that is attempted in this thesis, but it is likely to have a role in further developments towards automated CHD detection.

2.2.7 Deep Learning in Ultrasound Images

Recently, researchers working on ultrasound image analysis have adopted deep learning approaches from the wider computer vision community. Chen et al. [151] use a deep architecture that combines a spatial convolutional neural network and a temporal recurrent neural network (similar to the Encoder-Recurrent-Decoder architecture §2.1.7) to make use of temporal context features for standard viewing plane detection in fetal ultrasound videos, including the four chamber view of the heart as one of the three views considered (alongside the abdominal standard plane and the facial standard plane). Unfortunately, this approach requires a large amount of labelled training data, which is considerably harder to come by in the case of medical ultrasound then in many of the image classification tasks addressed by computer vision researchers. Gao et al. [152] have recently demonstrated that the data requirements for using deep networks with fetal ultrasound can be reduced by using transfer learning [58] from models trained on natural images. Though both of these papers perform view detection in fetal ultrasound video, neither deal specifically with the fetal heart or attempt to extract other useful information such as cardiac phase, position or orientation.

Baumgartner et al. [153, 154] also used CNNs to detect various anatomical structures in fetal ultrasound videos, including four heart views. Their method used a fully convolutional network [62] to simultaneously localise structures in the image and label them. Although, like in this thesis, they are explicitly aiming for real-time detection at high frame rate from video, they do not use any temporal constraints or information in their model. Their results show that the different heart views are the hardest to identify, with recall rates of 0.66 and 0.64 for the 3V view and RVOT views, respectively [154]. However due to use of a different dataset with different acquisition and annotation protocols, it is not appropriate to compare these results directly to those presented in this thesis. Furthermore, their method is not able to provide the level of detail that the method presented in this thesis can achieve.

2.3 Summary

Automatic analysis of ultrasound imagery is a well-studied problem, and learning methods including Adaboost, random forests, and CNNs have previously proved effective at detecting and segmenting anatomical structures. Despite the obvious applicability of spatial constraints to ultrasound imagery to reflect prior knowledge of the anatomy, this is not widespread in the literature. Analysis of ultrasound video using temporal constraints is less well-studied, and has mostly been confined to the somewhat different task of accurate boundary segmentation in highly constrained adult echocardiography. There is however a rich literature within computer vision on tackling these sorts of problems.

While automatic analysis of adult echocardiography is well-studied, there has been very little work on fetal heart analysis, which is more challenging in a number of ways. A few papers have detected frames containing the heart in broader scans, and one very recent work [154] also approximately localised different views of the heart. However, to the best of the author's knowledge, the work in this thesis (and the journal articles published alongside it) represents the only work to date to focus on the automatic analysis of 2D fetal cardiac screening specifically, and gives the most detailed analysis of this type of data yet attempted.

3

Rotation-Invariant Image Features for Analysis of Ultrasound Imagery

Contents

This chapter considers methods to deal with the unknown orientation in fetal ultrasound images. An existing method for rotation-invariant feature extraction is discussed, and extensions and alterations are developed in order to make the method suitable for use with fetal ultrasound videos.

The main ideas presented in this chapter were first presented at the IEEE International Symposium on Biomedical Imaging (ISBI) 2015, Brooklyn, New York, USA [155].

3.1 Background

The vast majority of algorithms for object detection in the computer vision literature are designed and tested on objects that occur in a very limited set of poses in real images, for example faces, cars and pedestrians, which nearly always appear upright. In such situations, rotation invariance of the detectors is not important, nor even desirable in many cases where the orientation contains useful contextual information.

However, a number of the anatomical structures in fetal ultrasound images can occur in a wide range of orientations due to differences in fetal lie and probe positioning during scanning (see §1.4.2). The nature of ultrasound as an imaging modality means that the angle of insonification *does* affect the appearance of the image and image features in more complex ways than simply the orientation, due to direction-dependent effects such as shadowing and specular acoustic reflection. However for a number of structures such as the heart this effect is not very pronounced. This chapter therefore investigates the application of rotation-invariant detection methods to heart detection in fetal ultrasound videos.

In order to detect objects at arbitrary orientations, it is common to train a sliding window detector on just one orientation, and at test time apply it to several rotated versions of the test image (e.g. [111]). This brute-force approach is rather clumsy and inefficient, which motivates the consideration of rotation-invariant object detection methods for localising anatomical structures in ultrasound. Alternatively, when using rotationally-variant features, a degree of robustness to small rotations can be learnt providing there are training examples with a range of such variations, essentially absorbing the complexity of rotations into the learning process. However, when applied to large orientation changes, this is likely to significantly decrease the performance of the learning algorithm.

Many approaches to achieving rotation-invariant description involve first estimating a dominant orientation and using this to rotate the window to a standard orientation before the feature extraction stage [40]. However, such techniques are obviously sensitive to the incorrect estimates of dominant orientation.

Villamizar et al. [156] performed an initial pose estimation step by passing one sliding window classifier over the image. This yielded a number of possible poses, defined by their position and orientation, which are then tested with a pose-specific classifier. Both stages used simple comparisons between HOG features within a Random Fern framework [157].

A more elegant line of work using rotationally-invariant basis functions has recently been proposed, exemplified by the work of Liu et al. [158, 159], and Skibbe and Reisert [160]. In this work, a circular region of an image is described using a set of basic features that are defined in such a way as to be analytically invariant to rotations of the underlying region. In this way the rotation invariance comes from neither learning nor normalisation, but rather from the image features themselves. Other works [158, 160] use these basis functions within voting frameworks to localise objects in images in a way that is analogous to a generalised Hough transform.

Some success has been achieved using these approaches for detecting simple objects such as cars in aerial imagery [159], but it is unclear whether the constraints imposed upon the basis functions to achieve rotation invariance reduce the descriptive and discriminative power of these features on more complex shapes. Another potential downside is speed, since the 'integral image' formulation [21] that so effectively speeds up many feature extraction methodologies cannot be used with the circular basis functions.

In the remainder of this chapter, the original feature extraction methodology of Liu et al. [159] is first described and some improvements are proposed in order to increase its computational efficiency. In Chapter 5 the suitability of these features for fetal heart detection in ultrasound imagery and is investigated experimentally.

3.2 Definition of Rotation-Invariant Features

This section describes the 'rotation-invariant histogram of gradients' method of Liu et al. [159]. The aim of the method is to create an image descriptor that captures the same information as the successful 'histogram of gradients' method, but in a way that is invariant to rotations of the underlying image patch.

When an image patch rotates, the change to the gradient field of the patch can be decoupled into two parts. The first is that the arrangement of the gradient vector rotates in 2D spaces. The second is that the orientation of the gradient vectors changes. This method comprises two distinct developments that deal with these two changes independently and together give rise to the rotation invariance of the framework: rotation-invariant basis functions and Fourier orientation histograms. Each of these is now described in turn.

3.2.1 Rotation-Invariant Basis Functions

Suppose that a circular image patch $I(r, \theta)$, $I : \mathbb{R} \times [0, 2\pi) \to \mathbb{R}$, of radius $R \in \mathbb{R}$ that is represented as a function of polar coordinates with radial coordinate $r \in \mathbb{R}$ and angular coordinate $\theta \in [0, 2\pi)$ (defined anti-clockwise from the increasing x-axis), where for notational convenience the origin of the polar coordinate system is defined to be the centre of the image patch. This requires choosing a fixed patch size for features, which for the purposes of this thesis can be based on the size of the heart, which is assumed to be known (§ 1.4.3). A description of the image patch is formed by multiplying the image patch by a basis function defined over the same spatial region and integrating over the region (in other words taking the *inner product* of the region and the basis function). The basis functions $u(r, \theta)$, $u : \mathbb{R} \times [0, 2\pi) \to \mathbb{C}$, are in general *complex-valued* and have the following general form comprised of the product of a radial part and an angular part:

$$u(r, \theta) = p(r)e^{\mathbf{i}k\theta} \tag{3.1}$$

where $p(r) \in \mathbb{R}$ represents some (real-valued) *radial profile* (more on this later), the integer $k \in \mathbb{Z}$ is referred to as the *rotation order* of the basis function, and $\mathbf{i}$ is the imaginary unit. When the product of the image region and the basis function is integrated over the region, the result is a complex-valued *raw feature*, $f \in \mathbb{C}$, as follows:

$$f = \int_0^{2\pi} \int_0^R u(r, \theta)\, I(r, \theta)\, r \,\mathrm{d}r \,\mathrm{d}\theta \tag{3.2}$$

Note that due to the representation in polar coordinates, there is an *area expansion factor* of r in Equation 3.2 representating the determinant of the Jacobian of the transformation between rectangular and polar co-ordinates. In practice, this integration takes place as a discrete convolution on the rectangularly sampled image grid. This area expansion factor therefore disappears to give a simple pixel-wise multiplication between the image patch and the basis function, followed by summation over the circular patch. However, a continuous polar representation will be used for the purpose of illustrating the technique as it is far more straightforward to represent mathematically in this way.

The important part of the definition of the basis function form in Equation 3.1 is that the angular part forms a Fourier series basis in the θ co-ordinate. Consequently, when the underlying image patch rotates by an angle ϕ (in radians with anti-clockwise defined as positive) the resulting feature value f simply undergoes a complex phase shift to give a new value $f' = fe^{-\mathbf{i}k\phi}$. This is analogous to the shift property of Fourier series coefficients. The complex magnitude of these features, i.e. $|f|$, therefore gives an analytically *rotation-invariant* quantity that represents some aspect of the appearance of the image patch. Perfect rotation invariance will not be achieved in the case of discrete images because of resampling effects when the image is rotated, however these discrepancies are very small.

By using a feature set of many such features with different rotation orders, k, a description of the image patch is built up in an analogous way to how the Fourier series coefficients form a description of the shape of a signal. The low rotation-order features (small $|k|$) represent the low frequency, more spatially smoothed information, whereas the high rotation-order features (large $|k|$) represent the higher-frequency, more detailed appearance information. Most of the useful information about appearance is contained within the lower few rotation orders, so in practice an image description is formed from just the features gathered using the basis functions with the lowest few rotation orders $k \in \mathbb{Z}_{0,K}$, where $K \in \mathbb{N}$ is some positive integer[1].

[1]In the case of scalar-valued image representations, the feature values from basis functions with rotation orders k and $-k$ are complex conjugates, so using the negative basis functions provides no new information. With vector-valued image representations 3.2.2, this is no longer the case, so

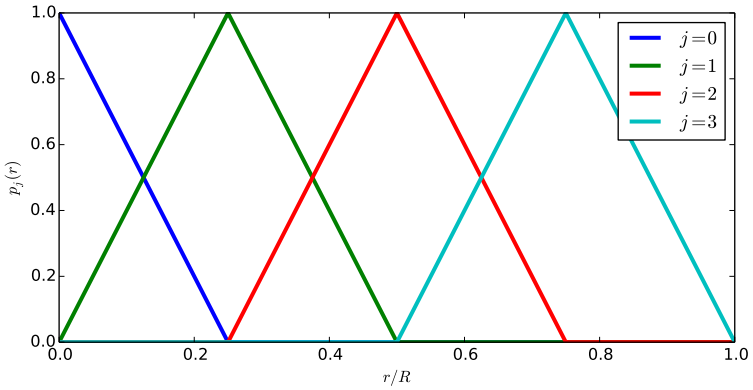


Figure 3.1: The radial profiles in an example set of basis functions with $J = 4$.

In principle, the radial profile $p(r)$ that defines the radial 'shape' of the basis function can be any real-valued function of the radial coordinate r. Furthermore a set of basis functions can be constructed using a number of such profiles (each appearing with a number of different rotation orders), in order to capture different information about the appearance of the image patch. *Zernike moments* [161] were in fact an early example of this technique where the radial profiles were described by Zernike polynomials. Following Liu et al. [159], a set of 'soft histogram' profiles, which form a set of soft bins with triangular profiles along the radial coordinate such that different features describe the appearance in different annular regions of the image patch (see Figure 3.1), is used here. If a set of $J \in \mathbb{N}$ such profiles indexed by the integer $j \in \mathbb{Z}_{0,J-1}$ is used, then the radial locations of the bin centres are given by $a_j = j\frac{R}{J}$ giving a set of J profiles $\{p_j(r)\}_{j\in\mathbb{Z}_{0,J-1}}$ defined by:

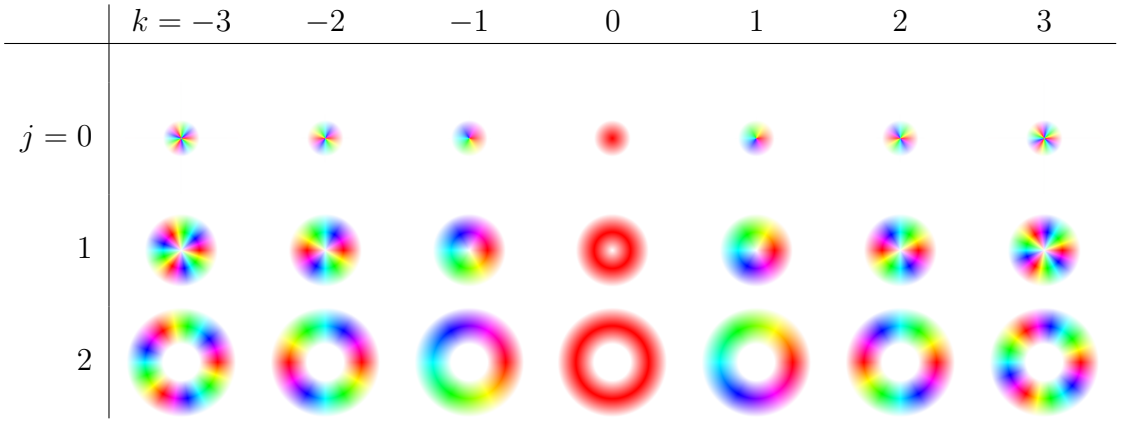
$$p_j(r) = \max\left(1 - \frac{|r - a_j|}{\frac{R}{J}}, 0\right) \tag{3.3}$$

Using these profiles, a set of basis functions $\{u_{j,k}(r,\theta)\}_{j\in\mathbb{Z}_{0,J-1},\, k\in\mathbb{Z}_{0,K}}$ is created using the different radial profiles and rotation orders where

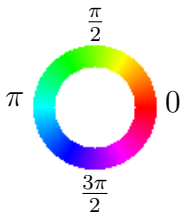
$$u_{j,k}(r,\theta) = p_j(r)e^{\mathbf{i}k\theta} \tag{3.4}$$

An example set of basis functions is shown in Figure 3.2.

negative rotation orders greater than $-K$ are also used.



(a) An example set of rotation-invariant basis functions with $J = 3$ and $K = 3$. In this representation, the image intensity represents the magnitude of the complex number and the hue represents the complex argument (complex phase).



(b) Legend for the interpretation of colour in the above table.

Figure 3.2: Set of rotation-invariant basis functions.

3.2.2 Fourier Orientation Histograms

The rotation-invariant basis functions in §3.2.1 provide a means to create a rotation-invariant description of a circular image patch based on the image intensity (or any transformation of it that results in a scalar image). However, many successful object detection algorithms have made use of the image intensity gradient, which is a vector-valued quantity. Typically these approaches form a magnitude-weighted orientation histogram [23] that places the intensity gradient vectors into bins according to both image location and intensity gradient orientation, and weights the contributions to the bins by the intensity gradient magnitude. Unfortunately this has poor orientation behaviour because the discrete histogram representation changes in a complex way when the underlying image rotates due to the intensity gradient vectors being re-allocated into bins (Figure 3.3).

Instead, and again following Liu et al. [159], the Fourier orientation histogram method maintains a *continuous* histogram of the orientations found within an image

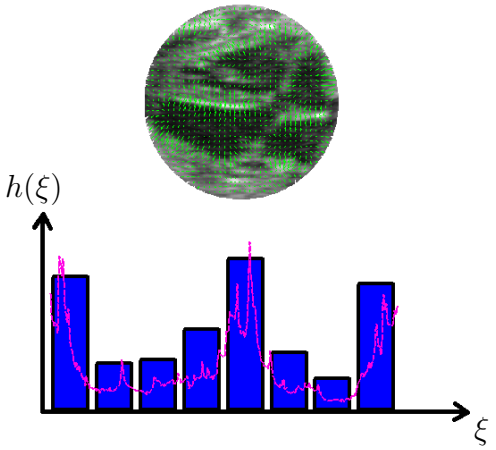
region, i.e. a function of orientation $h(\xi)$, $h : \mathbb{R} \to \mathbb{R}$, that may be evaluated for any orientation ξ (in rad). Since this function is by its nature periodic in orientation, a natural way to represent the continuous orientation histogram $h(\xi)$ is using its Fourier series coefficients, which shall be denoted $\{c_m\}_{m\in\mathbb{Z}}$, where $c_m \in \mathbb{C}$, i.e.

$$h(\xi) = \sum_{-\infty}^{\infty} c_m e^{\mathbf{i}m\xi} \tag{3.5}$$

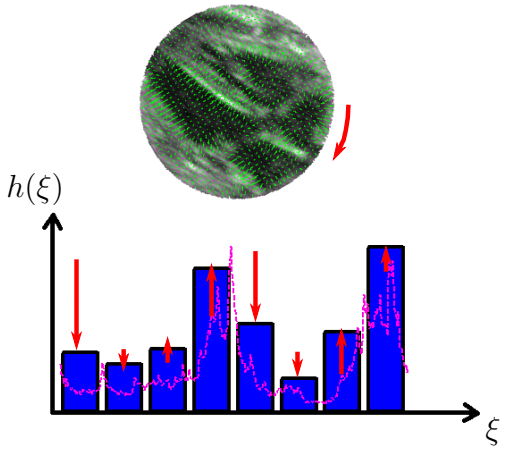
Note that because $h(\xi)$ is a real valued quantity, $c_m = \overline{c_{-m}}$ and therefore the coefficients for $m < 0$ carry no new information and may be ignored for the purposes of image description. Furthermore in practice only a finite, and typically quite small, number $M \in \mathbb{N}$ of the positive coefficients is maintained resulting in a smoothed Fourier histogram function. Some degree of smoothing is generally beneficial as it increases the robustness of the description to small changes in appearance.

The advantage of this representation is that when the underlying image patch rotates by an angle ϕ, in rad, (and therefore all the individual gradient vectors rotate by the same angle), the continuous Fourier histogram representation simply shifts cyclically giving $h'(\xi) = h(\xi - \phi)$ (Figure 3.3). This manifests itself as a phase shift of the Fourier coefficients, specifically $c'_m = c_m e^{-\mathbf{i}m\phi}$. Consequently the complex magnitude of the Fourier coefficients $\{|c_m|\}_{m\in\mathbb{Z}_{0,M}}$ gives a rotation-invariant description of the image patch.

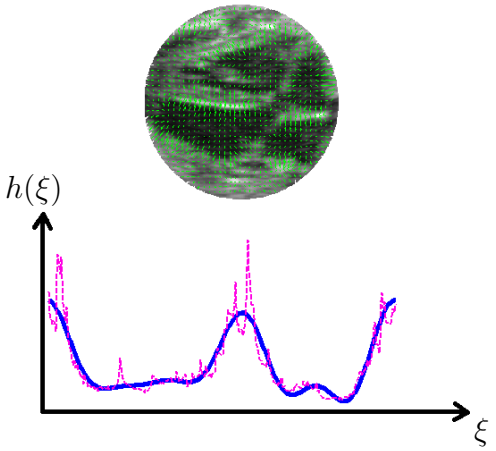
It remains to be shown how such an orientation histogram (represented by its Fourier coefficients) can be computed from a set of gradient vectors on an image grid. This proceeds in two stages: first a histogram representation is formed for each pixel individually, and then these are accumulated across a spatial region. Assume that an estimate of the image gradient can be generated using some standard method, yielding a vector valued gradient image $g(\mathbf{x})$, $g : \mathbb{R}^2 \to \mathbb{R}^2$ where the orientation of the gradient (defined anticlockwise from the increasing x-axis) at $\mathbf{x}$ is given by $\angle g(\mathbf{x})$ and the gradient magnitude at $\mathbf{x}$ is given by $\|g(\mathbf{x})\|$. The continuous orientation histogram of a single pixel $h(\mathbf{x}, \xi)$ is simply a δ-function at the gradient orientation of that pixel, with magnitude $\|g(\mathbf{x})\|$. Therefore, the



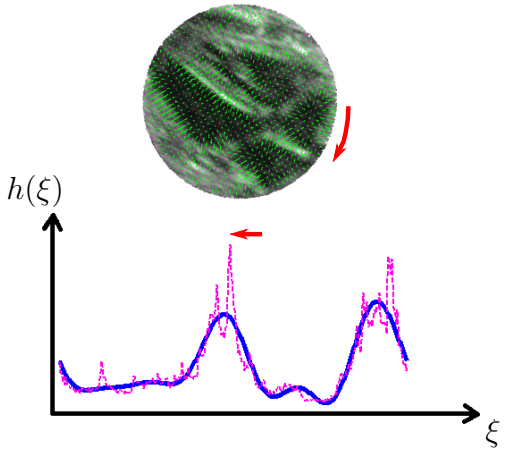
(a) An image patch with gradient vectors and an eight-bin discrete histogram of the magnitude-weighted gradient orientation. The magenta line indicates the 'true' continuous histogram (in fact a discrete histogram with a very large number of bins).



(b) When the patch rotates, the gradient vectors are re-binned and the histogram representation changes in an unpredictable way.



(c) A magnitude-weighted Fourier orientation histogram representation of the same image patch, using six complex-valued Fourier coefficients.



(d) The histogram changes via a cyclic shift (modulo small resampling effects) when the image patch rotates, which manifests itself as a phase shift of coefficients.

Figure 3.3: Illustrative comparison between discrete orientation histograms and Fourier orientation histograms.

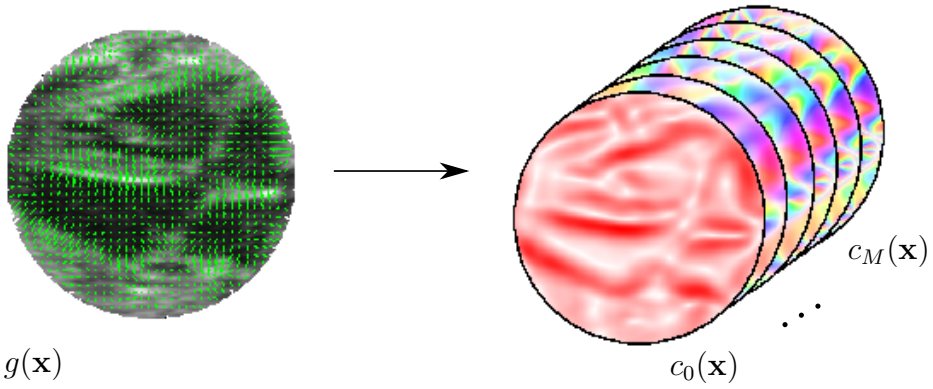


Figure 3.4: Illustration of the expansion process from the gradient image to the Fourier *coefficient images*. In the expanded images, the hue intensity represents the complex magnitude and the hue represents the complex argument using the convention of Figure 3.2

coefficients of the finite Fourier representation can be found from the standard Fourier series expansion of a δ-function:

$$h(\mathbf{x}, \xi) = \delta(\xi - \angle g(\mathbf{x})) \tag{3.6}$$

$$\Rightarrow c_m(\mathbf{x}) = \|g(\mathbf{x})\| e^{-\mathbf{i}m\angle g(\mathbf{x})} \tag{3.7}$$

It is helpful to think of this process as taking a vector-valued gradient image $g(\mathbf{x})$ and producing a set of complex-valued *coefficient images* $\{c_m(\mathbf{x})\}_{m\in\mathbb{Z}_{0,M}}$, where $c_m : \mathbb{R}^2 \to \mathbb{C}$. This is illustrated in Figure 3.4.

The orientation histogram over a patch of pixels can be formed by summing the orientation histograms for all of the individual pixels in the patch. Because of the linearity of the Fourier series representation, this can be achieved straightforwardly by accumulating the coefficients over the image patch within each of the expanded coefficient images.

3.2.3 Rotation-Invariant Histogram of Gradients

The rotation-invariant basis functions in §3.2.1 provide a method for rotation-invariant description of circular image patches in a single channel (scalar-valued) image. The Fourier orientation histograms in §3.2.2 provide a method to describe a rotation-invariant orientation histogram for a vector-valued image (such as an

image gradient field), but this only results in fully rotation-invariant features if the spatial aggregation process is rotation-invariant also. The method of Liu et al. [159] therefore combines these two methods to create a fully rotation-invariant orientation-histogram-style description.

A rotation-invariant orientation histogram feature, $f_{j,k,m} \in \mathbb{C}$, is found by applying the basis function $u_{j,k}(r,\theta)$ with profile index j and rotation order k to the Fourier orientation histogram coefficient image $c_m(r,\theta)$ of the patch (now represented in polar coordinates) as follows:

$$f_{j,k,m} = \int_0^{2\pi} \int_0^R u_{j,k}(r,\theta)\, c_m(r,\theta)\, r \,\mathrm{d}r \,\mathrm{d}\theta \tag{3.8}$$

The *effective rotation order*, $\hat{k}$, of the resulting raw image feature is $\hat{k} = k - m$. In other words if the image patch (and therefore the gradient field) rotates through an angle ϕ, the raw feature value undergoes a phase shift of $\hat{k}\phi$ to give $f'_{j,k,m} = f_{j,k,m} e^{-\mathbf{i}\hat{k}\phi}$. Consequently the complex magnitudes of these raw feature values are rotation-invariant.

Note that for $m > 0$, the coefficient images are complex-valued, and therefore the raw features gathered from basis functions with negative rotation orders $k < 0$ now carry information that is not present in the corresponding raw feature with $k > 0$.

3.2.4 Derived Rotation-Invariant Feature Sets

In order to use the rotation-invariant features in standard machine learning algorithms, real-valued rotation-invariant quantities must be extracted from the (generally complex-valued) raw features, f, that have been discussed so far. The term *derived* features shall be used to describe these features and differentiate them from the *raw* features. In fact this is slightly more complicated than simply taking the complex magnitude as has been suggested so far. To help with the notation in this section, the set of all raw features gathered from an image patch as shall be denoted $\mathcal{R}_{J,K,M}$, parametrised by $J \in \mathbb{N}$, $K \in \mathbb{N}_0$, and $M \in \mathbb{N}_0$. This includes raw features gathered with non-negative basis function rotation orders

$k \geq 0$ when $m = 0$ and all available (positive and negative) basis function rotation orders when $m > 0$. Specifically, in set builder notation:

$$\begin{aligned}\mathcal{R}_{J,K,M} = &\{f_{j,k,m} : (j \in \mathbb{Z}_{0,J-1}) \wedge (k \in \mathbb{Z}_{0,K}) \wedge (m = 0)\} \ldots \\ &\cup \{f_{j,k,m} : (j \in \mathbb{Z}_{0,J-1}) \wedge (k \in \mathbb{Z}_{-K,K}) \wedge (m \in \mathbb{Z}_{1,M})\}\end{aligned} \tag{3.9}$$

In the following subsections, various sets of derived features that can be created from these raw features are described. The 'basic' and 'coupled' features sets described below are similar to features used the original work [159], whereas the 'extra' set is a novel set of features.

'Basic' Rotation-Invariant Features

A number of rotation-invariant quantities can be derived from the individual raw features. These features shall be grouped together to form the 'basic' derived feature set. Recall that in general the raw features, $f_{j,k,m}$, are complex numbers. However a small number of the raw features will have $k = m = 0$ and will therefore be the result of a convolution of a purely real basis function with a purely real coefficient image and hence themselves be rotation-invariant purely real numbers (simply representing the gradient magnitude image weighted by a real-valued rotationally-symmetric basis function and integrated over some image region). These features shall collectively be known as the *first part* of the *basic derived feature set*, $\mathcal{B}^1_{J,K,M}$.

$$\mathcal{B}^1_{J,K,M} = \{f_{j,k,m} : (f_{j,k,m} \in \mathcal{R}_{J,K,M}) \wedge (k = m = 0)\} \tag{3.10}$$

Furthermore, some further raw features will be complex-valued but have an effective rotation order, $\hat{k}$, of 0 and therefore be rotation-invariant. Both the real parts and imaginary parts of these values can be used in the derived set. This set of derived features shall be collectively known as the *second part* of the *basic derived feature set*, $\mathcal{B}^2_{J,K,M}$:

$$\begin{aligned}\mathcal{B}^2_{J,K,M} =&\{\mathrm{Re}(f_{j,k,m}) : (f_{j,k,m} \in \mathcal{R}_{J,K,M}) \wedge (k-m=0) \wedge (k \neq 0)\} \ldots \\ &\cup \{\mathrm{Im}(f_{j,k,m}) : (f_{j,k,m} \in \mathcal{R}_{J,K,M}) \wedge (k-m=0) \wedge (k \neq 0)\}\end{aligned} \tag{3.11}$$

The remaining raw features in $\mathcal{R}_{J,K,M}$ are those with non-zero effective rotation orders, $k - m \neq 0$. These are therefore complex numbers whose arguments *do* vary with rotation of the image patch. Therefore, the third and final part of the basic derived feature set, $\mathcal{B}^3_{J,K,M}$, is comprised of the complex magnitudes of these raw features:

$$\mathcal{B}^3_{J,K,M} = \{|f_{j,k,m})| : (f_{j,k,m} \in \mathcal{R}_{J,K,M}) \wedge (k-m \neq 0)\} \tag{3.12}$$

The *basic derived feature set* $\mathcal{B}_{J,K,M}$ for use with machine learning algorithms is the union of these three parts:

$$\mathcal{B}_{J,K,M} = \mathcal{B}^1_{J,K,M} \cup \mathcal{B}^2_{J,K,M} \cup \mathcal{B}^3_{J,K,M} \tag{3.13}$$

Coupled Rotation-Invariant Features

The basic feature set (§3.2.4) contains derived features that are each obtained from a single raw feature. However, this ignores the fact that *relative* phase information between two raw features can also contain rotation-invariant information about the structure of the image patch. New derived features can therefore be created by *coupling* the raw features to give quantities that encode relative phase information. A coupling function, $d(f_1, f_2)$, $d : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$, is defined between two raw features as follows:

$$d(f_1, f_2) = \frac{f_1 \overline{f_2}}{|f_1||f_2|} \tag{3.14}$$

This will result in a rotation-invariant quantity provided that the effective rotation orders of the two raw features are the same. This is satisfied when $\hat{k}_1 = \hat{k}_2$ or equivalently $k_1 - m_1 = k_2 - m_2$.

To show this, consider writing the two features in the exponential form of a complex number, using magnitude r and argument θ:

$$f_1 = r_1 e^{\mathbf{i}\theta_1} \tag{3.15}$$

$$f_2 = r_2 e^{\mathbf{i}\theta_2} \tag{3.16}$$

Now the result of the coupling function is

$$\begin{aligned} d(f_1, f_2) &= \frac{f_1 \overline{f_2}}{|f_1||f_2|} \\ &= \frac{r_1 e^{\mathbf{i}\theta_1} r_2 e^{-\mathbf{i}\theta_2}}{r_1 r_2} \\ &= e^{\mathbf{i}(\theta_1 - \theta_2)} \end{aligned} \tag{3.17}$$

This shows how the coupling function can intuitively be understood to encode the *relative phase* between the two raw features.

Then, according to the definition of these features, if the image patch rotates through an arbitrary angle ϕ, the values of the features undergo a complex phase shift of $-\hat{k_n}\phi$ where $\hat{k}_n$ is the effective rotation order of feature n, equivalent to $k_n - m_n$ (see §3.2.3). Therefore after rotation, the new feature values are f_1' and f_2', where

$$f_1' = r_1 e^{\mathbf{i}\left(\theta_1 - \hat{k_1}\phi\right)} \tag{3.18}$$

$$f_2' = r_2 e^{\mathbf{i}\left(\theta_2 - \hat{k_2}\phi\right)} \tag{3.19}$$

In this form, it is straightforward to show that the coupling function in Equation 3.14 results in a rotation-invariant quantity:

$$
\begin{aligned}
d(f_1', f_2') &= \frac{f_1' \overline{f_2'}}{|f_1'||f_2'|} \\
&= \frac{r_1 e^{\mathbf{i}\left(\theta_1 - \hat{k_1}\phi\right)} \overline{r_2 e^{\mathbf{i}\left(\theta_2 - \hat{k_2}\phi\right)}}}{r_1 r_2} \\
&= \frac{r_1 e^{\mathbf{i}\left(\theta_1 - \hat{k_1}\phi\right)} r_2 e^{-\mathbf{i}\left(\theta_2 - \hat{k_2}\phi\right)}}{r_1 r_2} \\
&= e^{\mathbf{i}\left(\theta_1 - \hat{k_1}\phi - \theta_2 + \hat{k_2}\phi\right)} \\
&= e^{\mathbf{i}(\theta_1 - \theta_2)} \\
&= d(f_1, f_2)
\end{aligned} \tag{3.20}
$$

where the fifth line follows from the fact that the two raw features have the same effective rotation order $\hat{k_1} = \hat{k_2}$, and therefore ϕ cancels out from the expression.

If the raw features (both) have an effective rotation order, $\hat{k}$, of 0, then the coupled features will contain information that is already contained within the basic feature set and are therefore not placed in the coupled feature set. Note also that dividing by the raw feature magnitude in Equation 3.14 is not strictly necessary in order to produce a rotation-invariant quantity. However, by normalising by the magnitude this method ensures that the derived features represent only the relative phase information and the effect of the raw feature magnitudes, which is captured by other derived features in the basic feature, is removed.

The results of the coupling function are complex-valued and rotation-invariant, and therefore both the resulting real and imaginary parts may be used as derived features. A derived coupled feature set, $\mathcal{C}_{J,K,M}$, is defined in the following way:

$$\begin{aligned}\mathcal{C}_{J,K,M} =&\{\mathrm{Re}\left(d(f_{j_1,k_1,m_1}, f_{j_2,k_2,m_2})\right) : (f_{j_1,k_1,m_1} \in \mathcal{R}_{J,K,M}) \wedge \dots \\ &(f_{j_2,k_2,m_2} \in \mathcal{R}_{J,K,M} \setminus f_{j_1,k_1,m_1}) \wedge \dots \\ &(k_1 - m_1 = k_2 - m_2) \wedge (k_1 - m_1 \neq 0)\} \dots \\ &\cup \{\mathrm{Im}\left(d(f_{j_1,k_1,m_1}, f_{j_2,k_2,m_2})\right) : (f_{j_1,k_1,m_1} \in \mathcal{R}_{J,K,M}) \wedge \dots \\ &(f_{j_2,k_2,m_2} \in \mathcal{R}_{J,K,M} \setminus f_{j_1,k_1,m_1}) \wedge \dots \\ &(k_1 - m_1 = k_2 - m_2) \wedge (k_1 - m_1 \neq 0)\} \end{aligned} \tag{3.21}$$

The coupled derived feature set contains a large number of extra features that encode finer level detail about the appearance of the image patch than the features in the basic feature set, and are not computationally expensive to calculate if the raw feature values are already known.

Extra Coupled Rotation-Invariant Features

In fact, coupling between raw features is not limited to raw features of the same effective rotation order. Here, a novel method for creating 'extra' coupled features (beyond those described in [159]) is introduced. These are found by coupling raw features with *different* effective rotation orders. This is achieved by first raising the raw features to a power, giving an alternative, generalised coupling function $\tilde{d}(\cdot,\cdot)$, $\tilde{d} : \mathbb{C} \times \mathbb{C} \to \mathbb{C}$, as below:

$$\tilde{d}(f_1, f_2) = \frac{f_1^{p_1}\overline{f_2^{p_2}}}{|f_1^{p_1}||f_2^{p_2}|} \tag{3.22}$$

where $p_1 \in \mathbb{R}$ and $p_2 \in \mathbb{R}$ are powers to be determined. It is again assumed that the two features f_1' and f_2' are calculated from the same image patch as f_1 and f_2 after it has been rotated through an arbitrary angle ϕ. However, now is it *not* assumed that the effective rotation orders, $\hat{k_1}$ and $\hat{k_2}$, of the two raw features f_1 and f_2 are the same. The result of the generalised coupling function on f_1 and f_2 before rotation is then:

$$\begin{aligned}\tilde{d}(f_1, f_2) &= \frac{f_1^{p_1}\overline{f_2^{p_2}}}{|f_1^{p_1}||f_2^{p_2}|} \\ &= \frac{r_1^{p_1} e^{\mathbf{i}p_1\theta_1} r_2^{p_2} e^{-\mathbf{i}p_2\theta_2}}{r_1^{p_1} r_2^{p_2}} \\ &= e^{\mathbf{i}(p_1\theta_1 - p_2\theta_2)}\end{aligned} \tag{3.23}$$

after the rotation by ϕ the result of the generalised coupling calculation is:

$$\begin{aligned}\tilde{d}(f_1', f_2') &= \frac{f_1'^{p_1}\overline{f_2'^{p_2}}}{|f_1'^{p_1}||f_2'^{p_2}|} \\ &= \frac{r_1^{p_1} e^{\mathbf{i}p_1\left(\theta_1 - \hat{k_1}\phi\right)} r_2^{p_2} e^{-\mathbf{i}p_2\left(\theta_2 - \hat{k_2}\phi\right)}}{r_1^{p_1} r_2^{p_2}} \\ &= e^{\mathbf{i}\left(p_1\theta_1 - p_1\hat{k_1}\phi - p_2\theta_2 + p_2\hat{k_2}\phi\right)}\end{aligned} \tag{3.24}$$

Comparing Equations 3.23 and 3.24 shows that in order for the generalised coupling function to result in a rotation-invariant quantity p_1 and p_2 must satisfy:

$$p_1\hat{k_1} = p_2\hat{k_2} \tag{3.25}$$

such that the rotation angle ϕ cancels from Equation 3.24. Note that this is not possible in cases where either $\hat{k_1} = 0$ or $\hat{k_2} = 0$, so coupling cannot be performed in these cases. Notice also that the original coupling function $d(\cdot, \cdot)$ in Equation 3.21 may be considered a special case of the generalised coupling function $\tilde{d}(\cdot, \cdot)$ where $p_1 = p_2 = 1$ and Equation 3.25 is satisfied because $\hat{k_1} = \hat{k_2}$.

In the general case, there are many ways of choosing p_1 and p_2 to ensure that Equation 3.25 is satisfied. Perhaps the simplest is to choose $p_1 = \hat{k_2}$, $p_2 = \hat{k_1}$, meaning that each raw feature is raised to the power of the other feature's effective rotation order before coupling. This gives the following form for the generalised coupling function:

$$\tilde{d}(f_1, f_2) = \frac{f_1^{\hat{k_2}} \overline{f_2^{\hat{k_1}}}}{|f_1^{\hat{k_2}}||f_2^{\hat{k_1}}|} \tag{3.26}$$

Note that it is better to evaluate this using the following equivalent form to avoid the possibility of numerical overflow or underflow by normalising before raising to the power:

$$\tilde{d}(f_1, f_2) = \left(\frac{f_1}{|f_1|}\right)^{\hat{k}_2} \cdot \left(\frac{\overline{f_2}}{|f_2|}\right)^{\hat{k}_1} \tag{3.27}$$

Equipped with this coupling function, an *extra coupled feature set*, $\mathcal{E}_{J,K,M}$, can be created by combining all possible raw feature couplings except those that are already in the previously defined *coupled feature set*. This means omitting any combinations where either $\hat{k_1} = 0$ or $\hat{k_2} = 0$ (since coupling is not possible in these cases) and cases where $\hat{k_1} = \hat{k_2}$ (since these are already in the coupled feature set).

$$\begin{aligned}
\mathcal{E}_{J,K,M} =& \{\mathrm{Re}\left(\tilde{d}(f_{j_1,k_1,m_1}, f_{j_2,k_2,m_2})\right) : (f_{j_1,k_1,m_1} \in \mathcal{R}_{J,K,M}) \wedge \ldots \\
& (f_{j_2,k_2,m_2} \in \mathcal{R}_{J,K,M} \setminus f_{j_1,k_1,m_1}) \wedge \ldots \\
& (k_1 - m_1 \neq k_2 - m_2) \wedge (k_1 - m_1 \neq 0) \wedge (k_2 - m_2 \neq 0)\} \ldots \\
& \cup \{\mathrm{Im}\left(\tilde{d}(f_{j_1,k_1,m_1}, f_{j_2,k_2,m_2})\right) : (f_{j_1,k_1,m_1} \in \mathcal{R}_{J,K,M}) \wedge \ldots \\
& (f_{j_2,k_2,m_2} \in \mathcal{R}_{J,K,M} \setminus f_{j_1,k_1,m_1}) \wedge \ldots \\
& (k_1 - m_1 \neq k_2 - m_2) \wedge (k_1 - m_1 \neq 0) \wedge (k_2 - m_2 \neq 0)\}
\end{aligned} \tag{3.28}$$

Rotation-Equivariant Features

For some purposes in this thesis, it will be useful to derive not only rotation-invariant features from the raw features but also rotation-*equivariant* features. These are angular features that vary at the same rate as the rotation of the underlying image patch, i.e. if the image patch rotates by ϕ, the equivariant features also rotate by ϕ. Such features therefore encode information about the rotation of the patch that can be used to predict the orientation of structures. Each feature in the set is an angular feature in the range $[0, 2\pi)$ and the set of such features in constructed from

Symbol	Name	Description
$\mathcal{B}_{J,K,M}$	Basic Set	Contains rotation-invariant quantities from a single raw feature
$\mathcal{C}_{J,K,M}$	Coupled Set	Contains rotation-invariant quantities resulting from coupling two raw features with the same effective rotation order
$\mathcal{E}_{J,K,M}$	Extra Coupled Set	Contains rotation-invariant quantities resulting from coupling two raw features with different effective rotation orders
$\mathcal{Q}_{J,K,M}$	Equivariant Set	Contains rotation-equivariant quantities derived from a single raw feature

Table 3.1: Summary of derived feature sets used in this thesis.

the complex arguments of all raw features with effective rotation order $\hat{k} = 1$, and the negative complex argument of those with $\hat{k} = -1$.

$$\begin{aligned}\mathcal{Q}_{J,K,M} =& \{\angle\left(f_{j,k,m}\right) : \left(f_{j,k,m} \in \mathcal{R}_{J,K,M}\right) \wedge \left(k - m = 1\right)\} \ldots \\ & \cup \{-\angle\left(f_{j,k,m}\right) : \left(f_{j,k,m} \in \mathcal{R}_{J,K,M}\right) \wedge \left(k - m = -1\right)\}\end{aligned} \tag{3.29}$$

Summary

Table 3.1 provides a summary of the different feature sets used in this thesis.

3.3 Alternative Image Representations

In its original formulation [159] the rotation-invariant histogram of oriented gradients was used to gather rotation-invariant image features from the image gradient field. However, exactly the same formulation may be used with any 2D-vector field that describes an image patch. Furthermore, any scalar-valued image representation (such as the unaltered image intensity itself) can be used within the same framework by considering the image scalar representation as the single coefficient image of the Fourier orientation histogram with $M = 0$ (this is the description procedure described in §3.2.1 before the Fourier orientation histogram was introduced). These different options shall be referred to as *image representations*, and the following will be used in the remainder of this thesis:

Parameter	Value
Number of Levels	1
Averaging Window Size	10
Polynomial Degree, n	7
Standard Deviation σ of Smoothing Gaussian	1.5

Table 3.2: Table of parameters used for motion estimation.

Image Intensity (`int`) : The unprocessed image is used as a scalar image representation.

Image Gradient (`grad`) : The image gradient is computed using a Sobel filter with a kernel size of 5×5 pixels and thereafter considered to be a vector image representation.

Motion (`motion`) : A motion estimate is computed using the current video frame and the previous video frame. This is performed with the Farnebäck's motion estimation algorithm [162] with the parameters given in Table 3.2.

These parameters have been chosen to give a fast, approximate motion field estimate, rather than an accurate one. Once calculated, the motion estimate is used within the rotation-invariant framework in exactly the same way as the image gradient. Pilot experiments suggested that choosing parameters to give a more accurate estimate did not significantly affect results, but did significantly increase processing time.

3.4 Implementation of Fast Rotation-Invariant Features

Liu et al. [159] have made their MATLAB implementation of the Rotation-Invariant HOG feature extraction method publicly available[2]. According to their results, this implementation requires 18 seconds to calculate features densely for a 792×636 image and "most time is spent on the convolutions" [159]. Unfortunately this is too slow for the purposes here, where ideally the entire pipeline should run

[2] `http://lmb.informatik.uni-freiburg.de/resources/opensource/FourierHOG/`

at frame rate. A new C++ implementation has therefore been developed that is tailored for the purposes of this thesis and makes a number of changes in order to achieve a significant performance increase. The implementation used for experiments is publicly available as a documented C++ library on Github[3]. This section describes the various changes that have been made, listed approximately in the order contribution of the speed increase (largest contrbution first). The effect of some of these is explored experimentally in §5.6.1.

Frequency Domain Convolutions: The convolution of each of the coefficient images in the Fourier histogram expansion of a video frame with each basis function is by far the most time-consuming step in the process. This step is sped up by replacing the convolution operations with frequency domain multiplications. The frequency domain representations of the basis functions are pre-calculated and stored. Whenever a new frame arrives, its Fourier histogram expansion is found (in the spatial domain) to give a set of coefficient images. The 2D fast Fourier transform (FFT) algorithm is used to calculate and store the frequency domain representations of these coefficient images. Then, when a raw feature (that has not already been calculated) is requested by the random forest model, the pre-calculated frequency domain representations of the relevant basis function and coefficient image are multiplied element-wise and the resulting frequency domain image is then brought back to the spatial domain via the 2D inverse fast Fourier transform (IFFT) to give the requested raw feature image. See §3.5 for further details on finding the frequency domain representation of the basis functions analytically.

Image Subsampling: In practice ultrasound images can be resized to give a smaller image (and therefore fewer pixels to test and faster calculations for each pixel) without a significant loss of accuracy. Experience suggests that reducing the image size by a factor of approximately 3 can give a speed-up of around 4× whilst retaining high accuracy on the videos in the dataset.

[3] `http://github.com/CPBridge/RIFeatures`

On-Demand Feature Calculations: One significant practical advantage of random forests over a number of other learning algorithms is that only a (typically quite small) subset of the possible features needs to be evaluated in order to make a decision about a given data point. It is therefore highly wasteful to calculate all features for all datapoints, particularly in a performance-critical setting. This is especially useful when feature coupling is used, because then the number of possible features becomes very large (typically thousands). This is exploited by only calculating features as they are requested by the random forests. Calculated values are stored so that they may be re-used if requested later by another node in the random forest.

Low-level Programming: One simple reason the new implementation is significantly faster than the previous implementation is the use of a low-level compiled programming language (C++) with heavy use of compiler optimisations, rather than a high-level, interpreted language (MATLAB). However, MATLAB uses techniques such as just-in-time (JIT) compilation to improve performance and common tasks such as convolution are heavily optimised, reducing the performance difference.

Multi-threading: The random forests implementation exploits multi-core processors to speed up the calculation of random forest results by using an independent thread for each tree in the forest. This is not trivial to achieve in combination with the rotation-invariant feature calculations because 'cached' feature images (those that have been previously calculated and stored) must be shared between the threads in a thread-safe manner. The implementation uses thread locks to control read and write access to these shared resources in order to enable parallel threads make calls to the feature extraction routines. A simplified programme listing is shown in §A.3 and full details may be found in the Github repository. Experience suggests that using 8 threads, this can provide an approximately 2-3× speed-up relative to a single-threaded implementation.

Automatic Choice of Calculation Method: As discussed above, the use of frequency domain calculations can result in significant speed increases over spatial domain convolutions. Such calculations necessarily involve calculating the features for every pixel in the input image at once. However, in the low levels of a random forest (nodes far from the root) there are typically a small number of data points arriving at each node. For such nodes it may therefore be more efficient to use spatial domain convolutions for just the requested points, rather than use frequency domain calculations for the entire image. The implementation of the random forests algorithm used in this thesis requests the features for all points that arrive at a node together, allowing the feature extraction routine to decide which way will be the most efficient way to perform the calculations for the requested number of points.

3.5 Frequency Domain Calculations

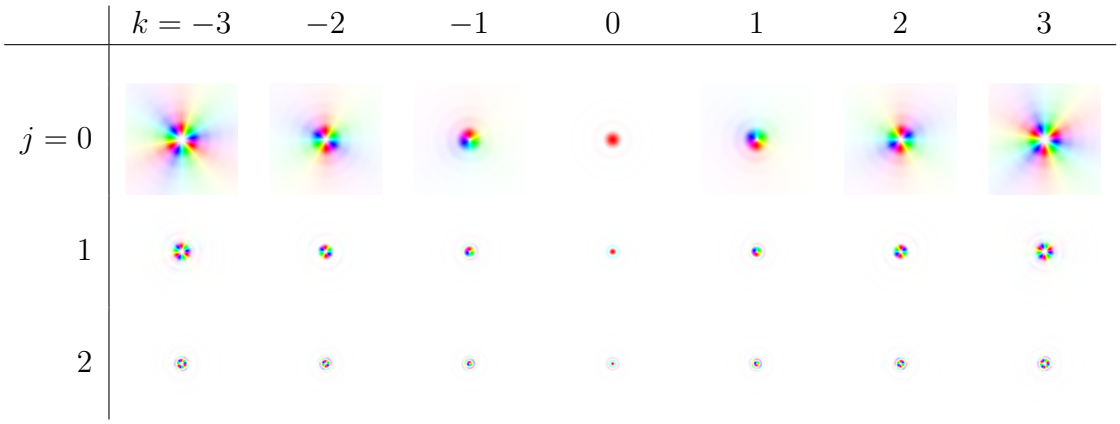
So far the procedure for creating a rotation-invariant description of some image patch has been considered. This is achieved by integrating the product of the basis function and the patch (Equation 3.2) or a Fourier orientation histogram coefficient image generated from the patch (Equation 3.8). For the purposes of object detection, this procedure must be performed on all such circular patches within the test image. The results are images of raw features, $f_{j,k,m}(\mathbf{x})$, that are generated through *convolution* of the basis functions with the input image (or Fourier orientation histogram coefficient channel). This section shall consider all images to be coefficient images without loss of generality since a scalar input image can be considered as the sole coefficient image $m = 0$ in an expansion with $M = 0$. Expressing the (spatial domain) basis functions in Cartesian coordinates, the convolution operation may be expressed:

$$f_{j,k,m}(\mathbf{x}) = \iint_{\mathcal{U}} u_{j,k}(\mathbf{u}) c_m(\mathbf{x} - \mathbf{u}) \, \mathrm{d}\mathbf{u} \tag{3.30}$$

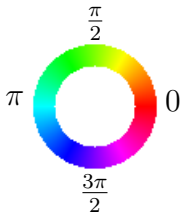
where $\mathcal{U}$ is the region of spatial support for the basis functions.

In many situations large speed improvements can be made to the convolution operation by moving from spatial domain convolution to the equivalent frequency domain multiplication. If the size of the image is $X \times Y$, and the size of the basis function is $R \times R$ (though the basis function is circular it must still be represented on a square rectangular grid), then performing a 2D convolution operation has complexity $\mathcal{O}(XYR^2)$. By contrast, if the fast Fourier transform (FFT) of the filter can be pre-calculated and stored, filtering in the frequency domain involves finding the FFT of the input image with complexity $\mathcal{O}(XY \log(XY))$, followed by pixel-wise multiplication by the frequency domain filter with complexity $\mathcal{O}(XY)$, and finally performing the inverse FFT with complexity $\mathcal{O}(XY \log(XY))$ again. Due to the lower computational complexity, this can be achieved more quickly than direct convolution, especially when the image and the filter are relatively large.

Therefore, this section explores the frequency domain representations of the basis functions described in §3.2.1. Whilst it is possible to calculate the frequency domain representations numerically via the FFT, an analytical representation is derived in the Appendix (§A.4) for increased flexibility and increased understanding of the filtering process. The frequency domain representations of an example set of basis functions calculated using this analytical representation is shown in Figure 3.5.



(a) Frequency domain representations of the basis functions shown in Figure 3.2 (page 53). In this representation, the image intensity represents the magnitude of the complex number and the hue represents the complex argument (complex phase).



(b) Legend for the interpretation of colour in the above table.

Figure 3.5: Frequency domain representations of an example set of basis functions.

4

Random Forests For Framewise Predictions

Contents

In this chapter, the random forest formulations for heart detection, heart classification, heart orientation, and cardiac phase prediction are presented. The formulation of the circular regression forest was presented in a journal article in *Medical Image Analysis* [163].

4.1 Overview

The random forests algorithm [35, 36] is a popular machine learning algorithm that has been used in a number of contexts, including analysis of natural and medical images [37, 72, 73, 121–125, 164]. It is well suited to the analysis of

fetal cardiac ultrasound videos because it is highly flexible, and can therefore be adapted to the tasks of view classification and phase estimation, it can be highly computationally efficient, due to only needing to calculate a subset of the possible feature in order to make a prediction, and is less vulnerable to overfitting than many other learning methods.

The presentation of the random forests algorithm in this chapter largely follows that of Criminisi et al. [37]. As a general framework, random forests can be considered as a method for predicting an *output label*, l, belonging to some label space, $\mathbb{L}$, based on some d-dimensional vector of possible features, $\mathbf{f} \in \mathbb{R}^d$, measuring different aspects of the image information. Note that although for notational simplicity the set of all possible features is described as a vector, in practice it may not be necessary to ever create the full vector when implementing the algorithm. In many applications, the feature values can be calculated on-the-fly when they are needed for both training and testing.

A forest consists of an ensemble of binary decision trees, where each tree is a set of *nodes*. Some nodes are *split nodes*, which carry out some test on the input features and pass the input down to either its left or right child node depending upon the result of that test. The input is passed down the tree from a single *root node* through split nodes until it reaches a *leaf node*. Each leaf node contains a *leaf distribution* over the value of the output label $l \in \mathbb{L}$ given the image information contained in $\mathbf{f}$, i.e. the distribution $p(l \mid \mathbf{f})$. See Figure 4.1 for an illustration of a single decision tree in a random forest.

Here the form of the the split function, $g(\cdot)$, at each split node with index $n \in \mathbb{N}_0$ shall be constrained to one that compares the value of one element of the feature vector to a threshold value. Let $\xi(\mathbf{f})$ be a function that extracts a single value from the feature vector, i.e. $\xi : \mathbb{R}^d \to \mathbb{R}$. This value is then compared to a threshold τ resulting in a binary split function $g(\mathbf{f}, \Theta) \in \{0, 1\}$ where $\Theta = (\xi, \tau)$ is a tuple representing the parameters of the split node.

$$g(\mathbf{f}, \Theta_n) = \mathbb{I}\left[\xi_n(\mathbf{f}) > \tau_n\right] \tag{4.1}$$

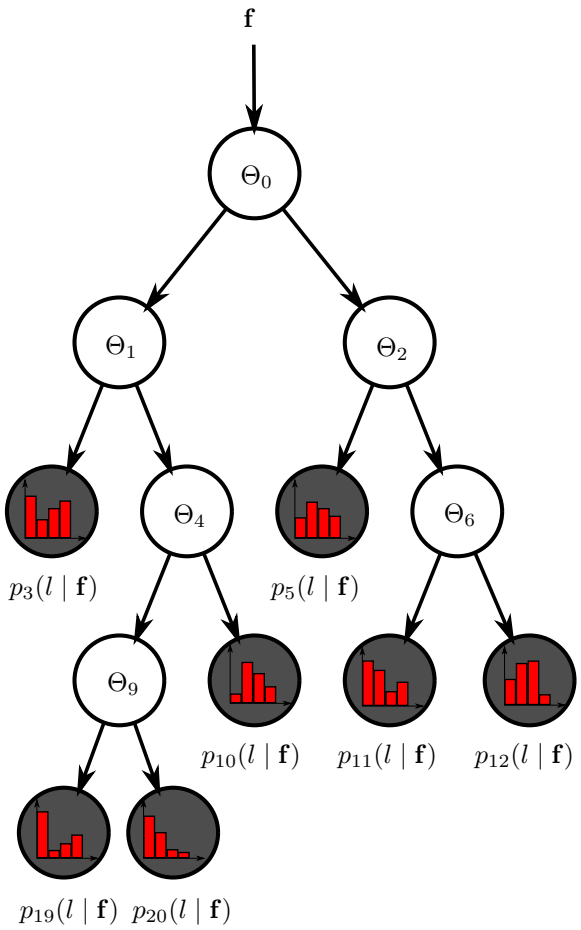


Figure 4.1: Cartoon illustration of a single decision tree in a random forest. The white nodes represent the split nodes, which are described by the feature index, i_n, and threshold, τ_n, used to split the data that arrive at the node. The grey nodes represent the leaf nodes that contain a distribution over the output label space. The form of this leaf distribution varies according to the prediction task, but a four class discrete distribution is shown here for illustrative purposes.

where $\mathbb{I}[\cdot]$ is the *indicator function*.

At test time, the unseen test samples are passed into the root node of each tree and move down the nodes until they arrive at one leaf node per tree. Depending on the application, the results from the set of trees are somehow aggregated across the forest.

4.1.1 Training

Training the forests involves deciding which nodes are leaf nodes, choosing the parameters of each leaf distribution, and choosing the parameters of each split

node (ξ_n and τ_n) given a set of labelled training data, for which the label value and all feature values are known.

This is done by first randomly partitioning (*bagging*) the labelled training data into partitions, and training each tree in the forest on a different partition in order to increase the diversity of the trees and hence reduce overfitting. Bagging is implemented by randomly and independently choosing a training set for each tree in the forest consisting of a fraction λ_{bag} of the full training set.

The set of output labels for training data that arrive at node n during training will denoted by $\mathcal{S}_n = \{s\},\ s \in \mathbb{L}$.

The aim of the training process is to find which features split the training data in a way that leads to the training points with similar training labels being grouped together in the children nodes, leading to increasingly 'pure' nodes towards the bottom of the forest. The notion of 'purity' is captured by an *information gain* function, $G(\cdot)$, that gives high scores to splits that lead to purer child nodes. The form of this information gain function varies for specific implementation of forests.

The trees are trained in a greedy, node-wise manner, starting at the root node. At each node, a set of randomly-generated feature selection functions ξ, are chosen and evaluated using the information gain function. The feature that gives the best split and the corresponding threshold (found by brute-force optimisation) are chosen for that node, and training proceeds to the children nodes.

A number of different *stopping criteria* are used to decide when a node should become a leaf node rather than a split node, thereby terminating the training process for that branch of the tree. These stopping criteria are:

- The number of training samples in the node goes below some minimum level, N_{nodemin}.
- A certain maximum depth, $D_{\max}$, in the forest is reached.
- The information gain of the best feature goes below some minimum value.

and a leaf node is declared if any of the above criteria are met.

4.2 Classification Forests

Classification forests are among the most commonly used formulations of the random forests algorithm. Their purpose is to classify the input into one of a number of discrete classes given the input data. Consequently, the label space $\mathbb{L}$ is some set of discrete labels. Later in this thesis, a classification forest will be used to classify image windows as belonging to one of the different fetal heart views or a background (BG) class, giving a label space $\mathbb{L} = \{\text{BG}, \text{4C}, \text{LVOT}, \text{3V}\}$. In this case, the leaf distributions are empirical discrete distributions over these four class labels. Whilst the formulation of random forest classifiers is well known, it is presented briefly here for completeness and to provide a comparison with the circular regression forests in §4.3.

The information gain function in the case of classification forests measures the change in discrete entropy before and after the split:

$$G(\mathcal{S}_n, \mathcal{S}_n^{(L)}, \mathcal{S}_n^{(R)}) = h(\mathcal{S}_n) - \sum_{j \in \{L,R\}} \frac{|\mathcal{S}_n^{(j)}|}{|\mathcal{S}_n|} h(\mathcal{S}_n^{(j)}) \tag{4.2}$$

where $\mathcal{S}_n$ is the set of labels in the n^{th} node (being trained), and $\mathcal{S}_n^{(L)}$ and $\mathcal{S}_n^{(R)}$ are respectively the labels of the training samples that would be moved to the left and right nodes after the proposed split. $h(\cdot)$ represents the Shannon entropy of a set of discrete labels:

$$h(\mathcal{S}) = -\sum_{l \in \mathbb{L}} p(l) \log p(l) \tag{4.3}$$

where the label probabilities $p(l)$ represent the empirical probabilities within the set $\mathcal{S}$. The value of information gain below which training is terminated is denoted $G_{\text{min},c}$.

4.2.1 Discrete Leaf Distributions

The leaf nodes in classification forests are empirical discrete distributions over the discrete output label space. During training, they are fitted by simply counting

the proportion of occurrences of each class label in the training set that arrives at each node. The resulting leaf distribution may be written

$$p(l \mid \mathbf{f}) = p_l \tag{4.4}$$

where p_l are constants $0 \leq p_l \leq 1$ and

$$\sum_{l \in \mathbb{L}} p_l = 1 \tag{4.5}$$

In order to generate a single prediction for the output label, the single most likely label is chosen after combining the leaf probabilities from each of the T trees:

$$l^* = \arg\max_l \frac{1}{T} \sum_{t=0}^{T-1} p_l \tag{4.6}$$

4.3 Circular Regression Forests

One aim of this thesis is to predict the cardiac phase (i.e. point in the cardiac cycle) given the image features. This will be referred to here as a *circular regression* problem because the output label is an angle $\mathbb{L} \in [0, 2\pi)$ belonging in a circular space. It is inappropriate to use information gain functions intended for standard regression (onto a real-valued output label space), because this would not respect the circular nature of the label space. For example, consider two training samples with labels $s_1 = \epsilon$ and $s_2 = 2\pi - \epsilon$, where ϵ is some small positive number ($\epsilon \ll \pi$). Any information gain function intended for regression onto a real-valued variable would heavily penalise a split that resulted in s_1 and s_2 being sent to the same child node, but in fact s_1 and s_2 are very close when they represent an angular variable.

This section therefore presents a novel formulation of random forests that is designed for regression onto a circular output space. This is achieved by adapting the following popular information gain function designed for linear regression forests [165]:

$$G(\mathcal{S}_n, \mathcal{S}_L, \mathcal{S}_R) = \sum_{s \in \mathcal{S}_n} (s - \mu(\mathcal{S}_n))^2 - \sum_{j \in \{L,R\}} \left(\sum_{s \in \mathcal{S}_n^{(j)}} \left(s - \mu(\mathcal{S}_n^{(j)}) \right)^2 \right) \tag{4.7}$$

where $\mu(\mathcal{S})$ calculates the mean label of the set $\mathcal{S}$, defined in the standard way. One can understand this information gain function as comparing the sum of square distances from the mean within the node and the proposed child nodes. A split that substantially reduces the sum of square distances from the mean will have a high information gain score.

In order to adapt this information gain function for circular regression two things are needed: a method for calculating a meaningful mean of a set of angular variables and a meaningful way to measure distance between two angular variables. For the mean label, the following measure taken from Jammalamadaka and Sen Gupta [166] is used:

$$\mu(\mathcal{S}) = \text{atan2}\left(\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \sin s, \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \cos s \right) \tag{4.8}$$

where atan2$(\cdot)$ is the four-quadrant arctangent function. One can intuitively understand this as the orientation of the resultant vector formed by summing unit vectors in the directions of each of the angular labels in the set.

The measure of distance on a circle, also from Jammalamadaka and Sen Gupta [166], is as follows:

$$\frac{1}{2}\left(1 - \cos\left(s_1 - s_2\right)\right) \tag{4.9}$$

This measure of distance evaluates to its minimum value of 0 if the two angles are the same (or differ by an integer multiple of 2π) and evaluates to its maximum value of 1 if the two angles differ by π (or an odd integer multiple of π such as $\pm 3\pi$ or $\pm 5\pi$). Combining these two, an information gain measure can be created that measures the sum of squared circular distances from the circular mean:

$$G(\mathcal{S}_n, \mathcal{S}_n^{(L)}, \mathcal{S}_n^{(R)}) = \sum_{s \in \mathcal{S}_n} \frac{1}{2} \left(1 - \cos\left(s - \mu\left(\mathcal{S}_n\right)\right)\right)^2 - \ldots$$
$$\sum_{j \in \{L,R\}} \left(\sum_{s \in \mathcal{S}_n^{(j)}} \frac{1}{2} \left(1 - \cos\left(s - \mu\left(\mathcal{S}_n^{(j)}\right)\right)\right)^2 \right) \tag{4.10}$$

The value of information gain below which training is terminated is denoted $G_{\min,\phi}$.

4.3.1 Von Mises Leaf Distributions

In addition to a suitable information gain function, the circular regression forests need a leaf distribution that is suitable for the circular output label variables. The *von Mises distribution* (also known as the *circular normal* distribution) is an appropriate distribution that is commonly used for circular data. It is analogous in many ways to the Gaussian (normal) distribution for linear data except that it defined on a circular domain [166][1]. Unlike many distributions for circular data it has a relatively simple form for the PDF, and is also the maximum entropy distribution for a circular variable with a known circular mean and circular variance. A von Mises distribution is defined by two parameters: the *circular mean* μ, analogous to the mean of a Gaussian distribution, and the *concentration* κ, analogous to the reciprocal variance (or *precision*) of a Gaussian distribution $1/\sigma^2$. The PDF of the von Mises distribution has the following form:

$$\mathcal{V}(x \mid \mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa \cos(x-\mu)} \tag{4.11}$$

where $I_0(\cdot)$ is the zero-order *modified Bessel function of the first kind*. Figure 4.2 shows some example von Mises distributions for different parameters.

In each leaf node, of the circular regression forest, a von Mises distribution is fitted to the training samples in the node using a maximum likelihood estimate of the distribution parameters. The mean parameter μ can be found using the definition in

[1]This should not be confused with the *wrapped normal distribution*, which is simply a Gaussian PDF wrapped around the circular domain and is *not* the same as a von Mises distribution.

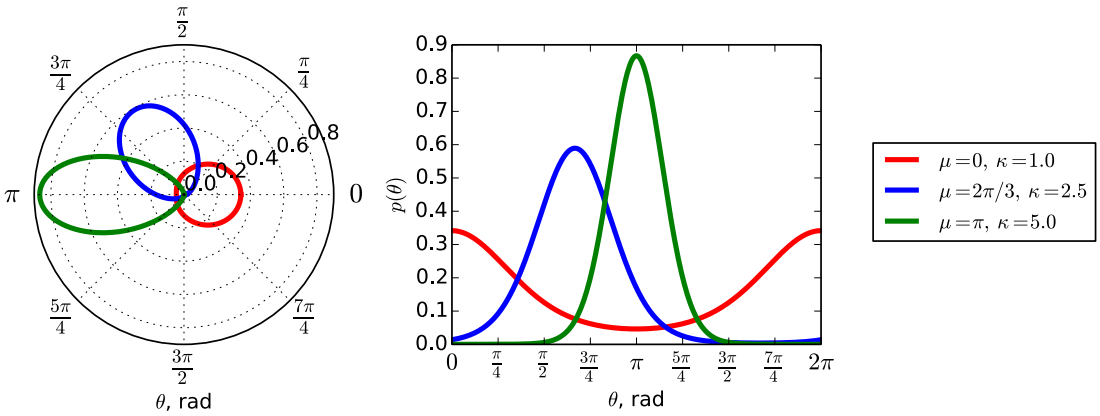


Figure 4.2: PDFs of three von Mises distributions defined over the interval 0 to 2π shown in both polar (*left*) and Cartesian (*right*) form. The μ parameter represents the mean angle, and the κ parameter describes the concentration of probability mass around this mean.

Equation 4.8. The concentration parameter κ can be found by first finding R, which is the length of the average vector when the angular data are treated as unit vectors:

$$R = \sqrt{\left(\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\cos s\right)^2 + \left(\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\sin s\right)^2} \tag{4.12}$$

Given R, κ can be estimated by numerically solving the following equation [166]:

$$R = \frac{I_1(\kappa)}{I_0(\kappa)} \tag{4.13}$$

This was implemented by re-writing the equation as a function $f(\kappa)$ to solve for $f(\kappa) = 0$, and solving using a standard hybrid non-linear solver that makes use of this function and its derivative:

$$f(\kappa) = I_1(\kappa) - RI_0(\kappa) \tag{4.14}$$

$$\frac{\mathrm{d}f}{\mathrm{d}\kappa} = \frac{1}{2}\left(I_0(\kappa) + I_2(\kappa)\right) - RI_1(\kappa) \tag{4.15}$$

In order to give a point estimate for the output label from the forest, the relevant leaf distributions are found in the usual way giving a set of leaf node parameters $\{(\mu_t, \kappa_t)\}_{t=0}^{T}$ from each tree. These are then combined following the maximum likelihood fusion approach of Stienne et al. [167]:

$$l^* = \operatorname{atan2}\left(\sum_{t=0}^{T} \kappa_t \sin(\mu_t), \sum_{t=0}^{T} \kappa_t \cos(\mu_t)\right) \tag{4.16}$$

This can be understood intuitively as a weighted average of the means μ_t of the leaf distributions where the weights is are concentration parameter κ_t (c.f. Equation 4.8).

4.4 Orientation Prediction with Rotation-Invariant Features

When using rotation-invariant features as the image features, the estimation of view classification and cardiac phase takes place in a way that is invariant to the orientation of the image. In order to estimate the orientation, a further stage must be added after cardiac phase estimation that makes use of the rotation-equivariant feature set (§3.2.4).

Experience suggests that it is not necessary to build new decision forests for the task of orientation prediction. Rather, the clustering of the training data that results at the end of the phase prediction forest is sufficient to give good results for orientation prediction, even though it is not optimised for this purpose. Therefore, after a phase prediction forest has been trained, an individual orientation prediction model is fitted to the data in each leaf node[2] based on a single feature from the equivariant set, $\mathcal{A}_{J,K,M}$. For each data point the *offset angle*, δ, between the orientation label s_i and the j^{th} equivariant feature a_j is calculated as:

$$\delta_{ij} = a_j - s_i \tag{4.17}$$

A von Mises distribution (μ_c,κ_c) of this offset angle is then fitted across all the datapoints i in the leaf node for each feature j of rotation order one, and choose the feature j^* that has the largest concentration parameter κ. Then, at

[2]to continue the established tradition of arboreal terminology it might be helpful to think of this as an *epiphytic* distribution, as it is placed on top of the existing tree structure but does not influence its training.

test time, the PDF at each leaf node, n, is calculated using this chosen feature and its von Mises distribution

$$p(s) = \mathcal{V}(a_{j^*} - s \mid \mu, \kappa) \tag{4.18}$$

If a single output prediction is required, the von Mises distributions from the leaf nodes of the different forests are combined using the same approach as Equation 4.16.

4.5 Implementation Details

A bespoke implementation of the random forests algorithm was developed using the C++ programming language and used in all experiments in this thesis. The implementation uses OpenMP directives for parallel execution with multiple threads and permits on-the-fly feature calculations in order to avoid evaluating unnecessary features. It also uses code templates in order to use the same core software classes to implement the various specialisations of the random forests algorithm. The source code for the implementation is publicly available[3].

[3] `https://github.com/CPBridge/canopy`

5

Experimental Validation of Framewise Predictions

Contents

This chapter describes experiments to evaluate predictions of the heart state variables (visibility h, position $\mathbf{x}$, orientation θ, viewing plane v, and cardiac phase

ϕ) from each frame of a video in isolation (*framewise* predictions). The rotation-invariant feature (RIF) method in Chapter 3 is tested and compared to a similar method using traditional rectangular filter feature sets. Some of the results in this chapter formed part of an article in *Medical Image Analysis* [163]. The results here expand upon that article by considering the effect of further parameters (such as the composition of the random forest models), and performing a comparison of RIFs with more standard rectangular features.

First, the evaluation metrics used to assess the predictions of each of the different variables are introduced in §5.1, Then, the cross-validation methodology used is explained in §5.2. Then details of the models used for the RIF (§5.3) and rectangular filter (§5.4 and §5.5) experiments are presented. Finally, results are presented and discussed in §5.6.

5.1 Evaluation Metrics

The full framework estimates values for visibility h, position $\mathbf{x}$, orientation θ, viewing plane v, and cardiac phase ϕ from each frame. Consequently, a number of different metrics are used to compare the test results to the ground truth annotations:

Detection Error The heart is considered to be *correctly detected* if the distance between the predicted location ($\mathbf{x}_t$) and the ground truth annotation is less than $0.25R$, where R is the annotated radius, *and* the predicted view label v_t matches the ground truth annotation. The *detection error* is defined as the proportion of those frames in which the heart was annotated as being visible where the heart was not correctly detected (as defined above). Frames where the heart was annotated as not visible did not affect the value this metric, but are considered in the *false positive rate* metric.

Orientation Error The orientation error in a single frame is defined using the following circular distance metric between the estimated and ground truth orientation values (respectively θ_t and θ_t^*):

$$\frac{1}{2}\left(1 - \cos\left(\theta_t - \theta_t^*\right)\right) \tag{5.1}$$

This value is averaged over those frames that were *correctly detected* (as defined above), since the orientation error for an incorrectly detected frame is meaningless. A value of 0.0 would indicate that all orientation values matched exactly, whereas a value of 1.0 would indicate that all estimated values were wrong by exactly π rad (or 180°). If the estimates were random, the expected value for the orientation error averaged over a large number of frames would be 0.5.

Ground Truth Position Orientation Error This is the same as the orientation error metric except that the orientation is estimated at the ground truth position and using the ground truth class label for the heart rather than the position and view label predicted by the detection stage. This allows the orientation prediction stage to be evaluated independently of the detection and classification stages.

Cardiac Phase Error This is defined exactly as for the orientation error but between the estimated and ground truth cardiac phase, ϕ, values.

Ground Truth Position Cardiac Phase Error This is the same as the cardiac phase error metric except that the cardiac phase is estimated at the ground truth position and using the ground truth class label for the heart (and at the ground truth orientation in the case of the rectangular filters case) rather than the position and view label (and orientation) predicted by the detection stage. This allows the cardiac phase prediction stage to be evaluated independently of the detection and classification stages.

True Positive Rate This is defined as the proportion of those frames containing the heart (according to the manual annotation) in which the heart was detected within $0.25R$ of the ground truth annotation, regardless of the detected class label.

False Positive Rate This is defined as the proportion of frames *not* containing the heart in which a heart detection was made. In this case, frames where a detection was made but where the heart was annotated as 'obscured' are considered as false positives.

'Generous' False Positive Rate As above except that frames where a detection was made but where the heart was annotated as 'obscured' are *not* considered as false positives.

5.2 Cross-Validation Procedure

Due to the relatively small number of subjects in the dataset (§1.5), all experiments in this thesis (in this chapter and in Chapters 7 and 9) were conducted using a *leave-one-subject-out* cross-validation. In each cross-validation fold, all videos from one of the 12 subjects were removed from the dataset and the videos from the remaining 11 subjects were used to train all the models. This set of models was then tested on the videos from the left-out subject. In this way, each video is tested once in just one of the cross-validation folds. The presented results are the averaged results from each video tested in this manner. Furthermore averages for each evaluation metric are performed first across the frames in each video, and then across videos in the dataset, meaning that each video is given equal weight in the average regardless of its length. This prevents the performance on the longest videos in the set dominating the results.

5.3 Framewise Predictions Using Rotation-Invariant Features

This section describes the process used to obtain an estimate of the complete state of the heart from a single frame using models based on RIFs.

5.3.1 Models Used

The following models are used for framewise predictions using RIFs:

Detection and View Classification Forest (`RIFDetection`): This is a classification forest (§4.2) with an output space consisting of the three viewing plane labels and a background label (abbreviated BG) $\mathbb{L} = \{\text{BG}, \text{4C}, \text{LVOT}, \text{3V}\}$, and thus is used to detect the heart and its viewing plane. This is trained using a set of labelled heart patches, and a set of random non-heart patches chosen from the video dataset. The classification forest is invariant to the orientation of the heart by virtue of the choice of features. It is also trained with a training set drawn from all points in the cardiac cycle, and as such should be relatively robust to the image variation that it introduces.

View-Specific Cardiac Phase Regression Forests (`RIFPhase`): These are circular regression forests (§4.3) trained to give a continuous prediction of the cardiac phase value from the patch appearance. There is one such forest model for each of the three viewing plane classes, and each forest is trained using training data belonging exclusively to that class. They are invariant to orientation by virtue of the choice of features.

Orientation Regression Models (`RIFOri`): An orientation regression model (§4.4) is attached to each leaf node in each of the `RIFPhase` forests. It is trained using only the training data that arrive in the leaf node it is attached to.

5.3.2 Training

The models were trained on each cross validation fold of the database described in §1.5. From the videos in the fold's training partition, a training set of image patches was created containing 5,000 image patches of the heart in each of the three viewing planes and an equivalent number of 'background' patches, selected randomly from non-heart areas of the frames in the dataset. This gives a total of 30,000 training patches (15,000 positive patches containing the heart and 15,000

Symbol	Value	Unit	Description
N_{nodemin}	50	-	number of training data in a node below which a leaf is declared
$G_{\text{min},c}$	0.5	bits	minimum value of the information gain function for classification forests, below which a leaf is declared
$G_{\text{min},\phi}$	0.01	-	minimum value of the information gain function for circular regression forests, below which a leaf is declared
λ_{bag}	0.5	-	fraction of the training set used to train each tree
R_{train}	30	pixels	radius of the rotation-invariant basis functions

Table 5.1: Training parameters

background patches). Each positive image patch is circular and is centred at the annotated heart centre with the annotated radius as its radius. Background patches are also circular, have a random centre chosen such that the background patch centre is at least $0.3r$ from the annotated heart centre (if any) in the frame, where r is the annotated heart radius, and at least r from the edge of the ultrasound fan area in order to ensure that the entire patch sits within the valid image region.

The fixed training parameters were chosen empirically to give good results and are shown in Table 5.1.

All these models are trained using some derived set of RIFs of a certain basis size R_{train}. Before the feature extraction process began, the training patches were resized with a scale factor of R_{train}/r such that radius of the training patch matched the fixed basis size for the feature extraction. The number of features tested at each split node was set to a quarter of the total number of features available.

5.3.3 Testing

As discussed in §1.4.3, the approximate size of the heart in the image is assumed to be known at test time and described by the radius R_{test}. The process for predictions in a single frame using RIFs (Chapter 3) consists of the following stages:

1. The input frame is resized by a factor of $R_{\text{train}}/R_{\text{test}}$ such that the radius of the heart area in the image features matches the basis function radius used to calculate the features that were used to train the model.

2. Each pixel that is at least a distance R_{test} from the edge of the ultrasound fan area in the resized image is passed into the `RIFDetection` forest. After this step, the pixel with the highest probability of each of the non-background classes is found. The predicted viewing class v is then chosen to match the class of whichever of these pixels has the highest prediction score. In order to refine the position and smooth out local variations in prediction score, the location of the pixel with the highest prediction score then serves as the starting point of a mean-shift procedure (with window size 30 pixels) that operates on the prediction score values and gives a refined predicted position $\mathbf{x}$.

3. The probability value of the winning view class at the predicted location from the previous step is thresholded using some predetermined threshold τ. If the probability value is lower than the threshold, the heart is determined to be hidden ($h = 1$), otherwise it is determined to be visible.

4. The chosen pixel location is passed into the `RIFPhase` forest that matches its viewing plane label. The forest outputs a point estimate of the cardiac phase (the mean parameter of the von Mises distribution after combining the leaf nodes as in Equation 4.16), which is used as the estimated cardiac phase value for this frame.

5. The `RIFOri` model attached to each leaf node reached in the previous step is used to give an orientation estimate, the estimates from each tree are then combined to give a single orientation estimate for the frame.

5.3.4 Shorthand Names for RIF Feature Sets

Throughout the remainder of this thesis, a shorthand notation will be used to identify feature sets based on RIFs. This includes an abbreviated name of the image representation (`int` for intensity, `grad` for intensity gradient and `motion` for motion), followed by the J, K and M parameters, in that order (though the M parameter is omitted for intensity as it is a scalar representation and therefore there is no

Fourier orientation histogram expansion), and finally a description of the feature set as including `basic`, `coupled` or `extra` features. Multiple image representations may be used and are represented by concatenating the names of the feature sets

So for example, `int43_basic` represents a basic feature set (no coupled features) using intensity features with $J = 4$ and $K = 3$, whereas `grad432motion432_extra` represents a feature set using extra feature coupling and intensity gradient and motion image representations, both using $J = 4$, $K = 3$, and $M = 2$.

5.4 Rectangular Filter Feature Sets

In order to compare the performance of the RIFs with a more traditional method, the experiments were repeated using a different set of models trained on 'rectangular' features. Such features have been used extensively in computer vision in highly influential works such as those of Viola and Jones [21, 168], and Shotton et al. [72, 169] and in the latter works were found to work well as weak learners within learning algorithms based on random forests. They have also been used in several works on ultrasound image analysis, for example [104, 107–112, 114–118, 164], and other medical image analysis applications [73]. They have sometimes been referred to as *Haar* or *Haar-like* features due to a loose connection with Haar wavelets.

The definition of the features used in this is shown in Figure 5.1. At training time, random candidate features are chosen by choosing type, position, width, height and (where relevant) offset parameters such that all rectangles fit within the square $R \times R$ detection window.

At test time, the features may be evaluated very efficiently by first finding the *integral image* of the input image. Thereafter, the sum of pixels under a given rectangular region may be evaluated using just a few addition and subtraction operations using certain elements of the integral image, regardless of the size of the region [21].

The method of Zhu et al. [25] is used to extend this framework to vector-valued image representations (motion and intensity gradient fields). The orientation of each pixel is first discretised into one of N_b orientation bins, and weighted according

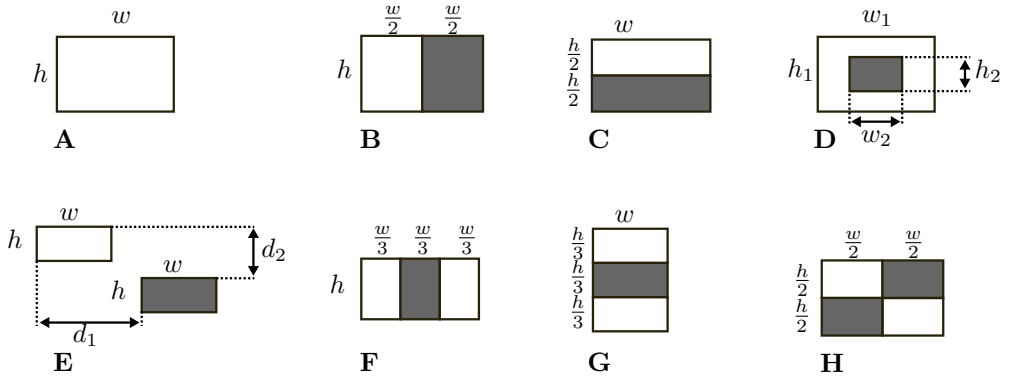


Figure 5.1: Definition of rectangular features. There are eight types of rectangular features, labelled **A** to **H**. Each type consists of a set of 1, 2, 3, or 4 rectangular regions laid out as in the diagram. The feature value is equal to the sum of the pixel values under the white rectangles minus the sum under the grey rectangles.

to its magnitude. This gives a set of N_b scalar-valued images. Each feature value selects just one randomly chosen orientation bin, and uses the rectangular features exactly as before.

5.5 Framewise Predictions Using Rectangular Filters

The procedure for predictions in a single frame using the rectangular features is similar to that outlined in §5.3. However in this case the features are not rotation-invariant and therefore a number of different models are trained for a number of orientations.

5.5.1 Models Used

The procedure for predictions in a single frame using the rectangular features makes use of the following models:

Detection and View Classification Forests (`RECDetection`): Like in the RIF case, these are classification forests (§4.2) with an output space consisting of the three viewing plane labels and a background label (abbreviated BG) $\mathbb{L} = \{\text{BG}, \text{4C}, \text{LVOT}, \text{3V}\}$. However these forests are not invariant to orientation and therefore there is a set of N_o such forests, trained to detect and classify

the heart at a set of equally spaced orientations in the range $[0, 2\pi)$. These orientations are referred to as $\theta_{\text{train},n}$ for the n^{th} orientation.

View-Specific Cardiac Phase Regression Forests (`RECPhase`): These are circular regression forests (§4.3) trained to predict the cardiac phase given an image patch containing a certain viewing plane in a certain orientation. As such there is one model for each of the classes in each of the orientations giving $3N_{\text{o}}$ such forest models.

Note that there is no separate orientation regression model, as the estimated orientation arises naturally as a results of performing the detection step at multiple orientations (see §5.5.3).

5.5.2 Training

The models based on rectangular filters were trained on each of the cross-validation folds. The training procedure and parameters were identical to those described in §5.3.2 except for the differences described below:

- The image patches in this case were square image patches (rather than circular patches) of side length $2R_{\text{train}}$.
- Before feature extraction began, the patches were rotated such that the ground truth orientation of the heart matched the training orientation for the forest. For the phase prediction forests, a random offset angle in the range $[-\frac{2\pi}{2N_o}, \frac{2\pi}{2N_o}]$ was was then applied to each patch. This means that the forest is trained to predict the cardiac phase for a heart with any orientation within the relevant bin.
- The total number of features tested at each split node was set to a fixed value of 3,000.

5.5.3 Testing

The testing procedure using random forest models is described below. It is similar to the one used for RIFs (§5.3.3) except that the orientation is now determined by applying the classification models trained at each orientation and choosing the one that gives the highest detection score.

1. The input frame is resized by a factor of $R_{\text{train}}/R_{\text{test}}$ such that the radius of the heart area in the image features matches the basis function radius used to calculate the features that were used to train the model.

2. Each pixel that is at least a distance R_{test} from the edge of the ultrasound fan area in the resized image is passed into each of the N_{o} `RECDetection` Forests. For each of the combinations of viewing plane and orientation, the pixel with the highest detection score from the output of the detection forest is chosen as a candidate detection. Out of these, the orientation and viewing plane with the highest scoring candidate is chosen as the predicted viewing plane and orientation combination.

3. The position prediction is then refined with mean shift, as in the rotation-invariant case.

4. The orientation prediction is refined by performing a weighted average of the orientations of the maximum scoring detector and the two either side of it, where the weights are the detection scores evaluated at the refined position estimate. This gives a continuous orientation prediction from the discrete models.

5. The cardiac phase prediction is then performed using the `RECPhase` forest for the correct viewing plane and the highest scoring orientation. The forest outputs a point estimate of the cardiac phase (the mean parameter of the von Mises distribution after combining the leaf nodes as in Equation 4.16).

5.5.4 Shorthand Names for Rectangular Feature Sets

Throughout the remainder of this thesis, a shorthand notation will be used to identify feature sets based on rectangular filter features. This begins with `rec` to indicate that rectangular filters are used. An abbreviated name of the image representation (`int` for intensity, `grad` for intensity gradient and `motion` for motion), follows. For vector representations, the number of orientation histogram N_b then follows. Multiple image representations may be used and are represented by concatenating the names of the feature sets.

So for example, `rec_int` represents a set using intensity features, and `rec_gradmotion_8` represents a set using intensity gradient and motion image representations, both using $N_b = 8$ orientation histogram bins.

5.6 Results

5.6.1 Calculation Speeds for Rotation-Invariant Features

To investigate the efficiency of calculation methods for the RIF extraction, the per-frame calculation times were measured and averaged across every video in the database using a number of different calculation methods. Each test was conducted with an `RIFDetection` forest model using gradient and motion features with the `grad322motion322_extra` RIF feature set consisting of $N_{\text{trees}} = 16$ trees and a maximum depth of $D_{\max} = 10$ levels, and an `RIFPhase` forest model $N_{\text{trees}} = 32$ trees and a maximum depth of $D_{\max} = 8$, representing a typical set of values that achieve high accuracy (see later sections).

The calculation methods are defined by two characteristics:

Raw Feature Calculation Method The method by which the raw RIFs are calculated. The `spatial` method uses entirely spatial (image) domain convolution operations at the individual points when requested by a node in one of the forest models, whereas the `freq` method uses frequency domain multiplications (§3.5) for the entire image, and stores the result for future queries. The `auto` method determines automatically for each query from a

Method	Average Calculation Time (ms)
`spatial`	516.4
`spatial` + `memo`	149.1
`freq`	13.5
`auto`	13.4
`auto` + `memo`	13.5

Table 5.2: Average calculation times per frame using different calculation methods.

node, which of the above methods will be fastest for the requested number of points. The intuition is that using spatial convolutions may be faster for cases where the raw feature is required at only a few image locations, but that the frequency domain method will be faster when a large number of points are required. The threshold number of requested points is set automatically based on empirical measurements of the speed of each method on images of the relevant size at the start of a test of a video.

Spatial Feature Memo-isation When the raw features are calculated using the spatial method, they may either be calculated for the specific image locations required and then discarded, or stored such that they may be used again later if the same feature is required at the same image location. The latter option may reduce redundant calculations, but there is an overhead associated with storing the calculated features in a thread-safe manner. Note that it only makes sense to use memo-isation with the `spatial` and `auto` method.

Note that these calculation methods do not affect the values of features calculated, except due to negligible numerical effects resulting from finite precision arithmetic. However, they do affect the speed at which the calculations are performed.

The results of the experiment are shown in Figure 5.2 and Table 5.2. It can clearly be seen that the use of the frequency domain calculations rather than spatial domain calculations (as used by Liu et al. [159]) results in a very significant speed up in the calculation of RIFs. Furthermore, it is clear that as a result of this and the various other improvements outlined in §3.4, it possible to use RIFs and random forests to construct a powerful classifier that can run at several tens of frames per second on

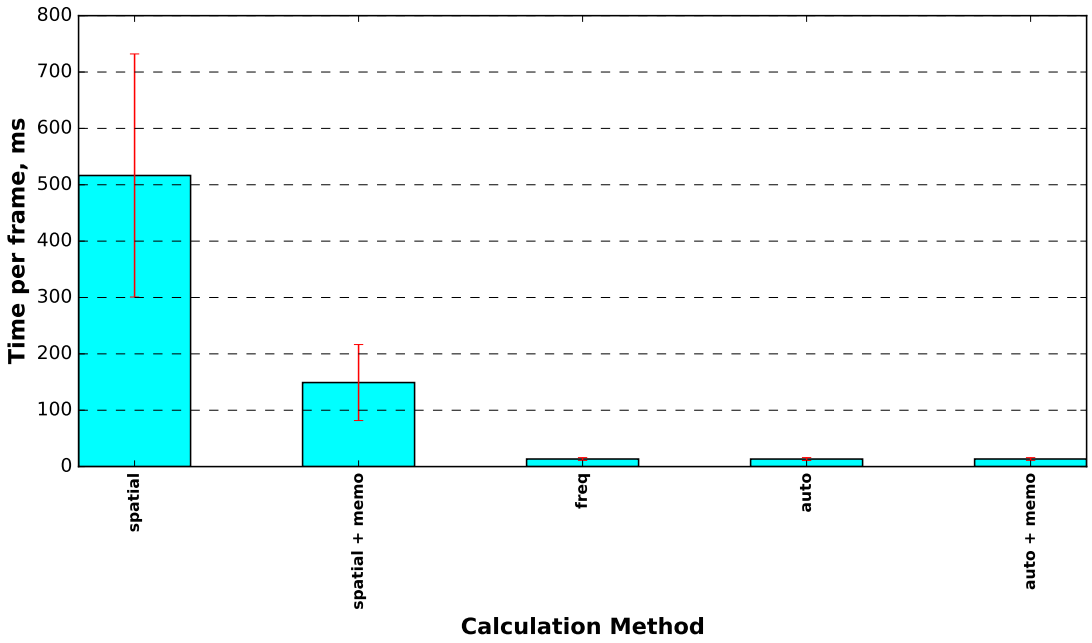


Figure 5.2: Average calculation times per frame for the different calculation methodologies. The blue bar represents the mean time and the red error bars show one standard deviation, in both cases calculated over each video in the dataset.

desktop hardware. The precise calculation times depend on a number of factors, including the number of trees in the forest, their depth, the image representation and the size of the image. Some of these factors will be investigated in later sections.

It is also clear that the `auto` calculation method results in very similar calculation times to the plain `freq`. This suggests that, although the `auto` method chooses the fastest method (spatial or frequency) for a given node, this does not necessarily mean that this is the fastest method for the entire forest model. In many cases it may happen that the feature extraction routine chooses the spatial method at a tree node that requests a small number of image locations, but later on another node requires the same raw feature at a number of other locations, in which case it may have been faster to have used the frequency domain calculation for the first node's request. This effect is more pronounced with larger, deeper forests.

Another factor is that the speed difference between the spatial and frequency methodologies is so great that the automatically-determined threshold for using the frequency domain method is very low. Typically this value is a few dozen image

locations. Consequently, there are only a few cases where the spatial method is chosen, and therefore the effect on the overall calculation time is negligible.

Furthermore Figure 5.2 shows that the memo-isation of spatial raw feature calculations significantly improves the calculation time, however not enough to make it competitive with the use of frequency domain calculations.

5.6.2 Detection Performance

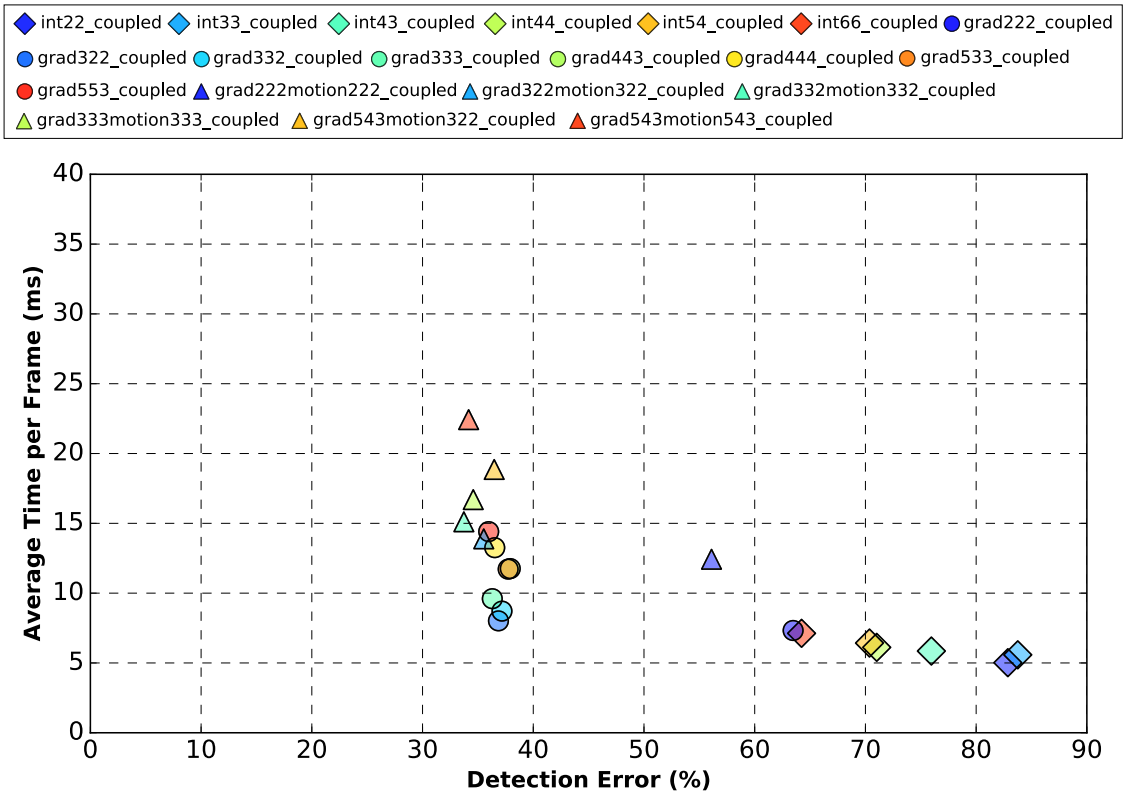
In this section, the performance of the different feature detection algorithms in detecting the heart and classifying its view is considered. This was assessed by experiments in which the feature set, the number of trees in the random forest model (N_{trees}), and the number of levels in the random forest model ($D_{\max}$) were varied.

Each model was tested using the cross-validation methodology described above, and evaluated using the Detection Error metric. The parameter values in Table 5.1 were used to train the forest models. For testing the RIF models, frequency domain feature extraction with automatic coupling was used. For rectangular features, the number of orientations was fixed at $N_o = 8$.

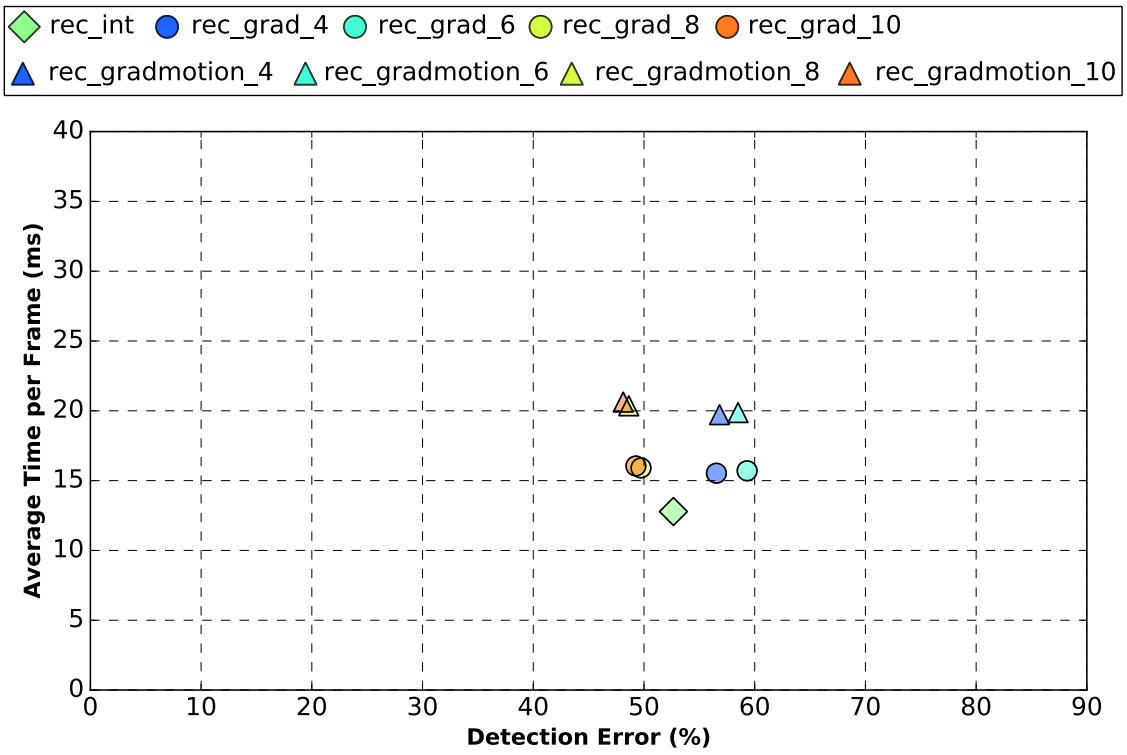
Figure 5.3 shows the trade-off between detection accuracy and calculation time for a number of different feature sets using 16 trees and 10 levels. The x-axis shows the detection error (as defined above) expressed as a percentage, and the y-axis shows the time taken per frame, in milliseconds. Consequently, the 'ideal' detection algorithm (fast and accurate), would appear in the bottom left of the plot.

Whilst the specific results vary with the number of trees and levels, the trend from Figure 5.3 is similar to that with other parameters. It can be seen that in general there is a trade-off between fast feature extraction and good classification performance. For both RIFs and rectangular features, using image-intensity-based representations gives very fast feature extraction, but generally much lower detection performance than gradient-based representations.

The forest models using RIFs are significantly more accurate than those using rectangular features, which demonstrates the advantage of building rotation invariance into the feature set. It is likely that using forest models trained at a larger



(a) RIF feature sets (only 'coupled' sets are shown).



(b) Rectangular feature sets.

Figure 5.3: Detection error and calculation time for RIF (5.3a) and rectangular (5.3b) feature sets, shown using forest models (`RIFDetectionRECDetection`) with $N_{\text{trees}} = 16$ trees and a maximum of $D_{\max}$ levels per tree.

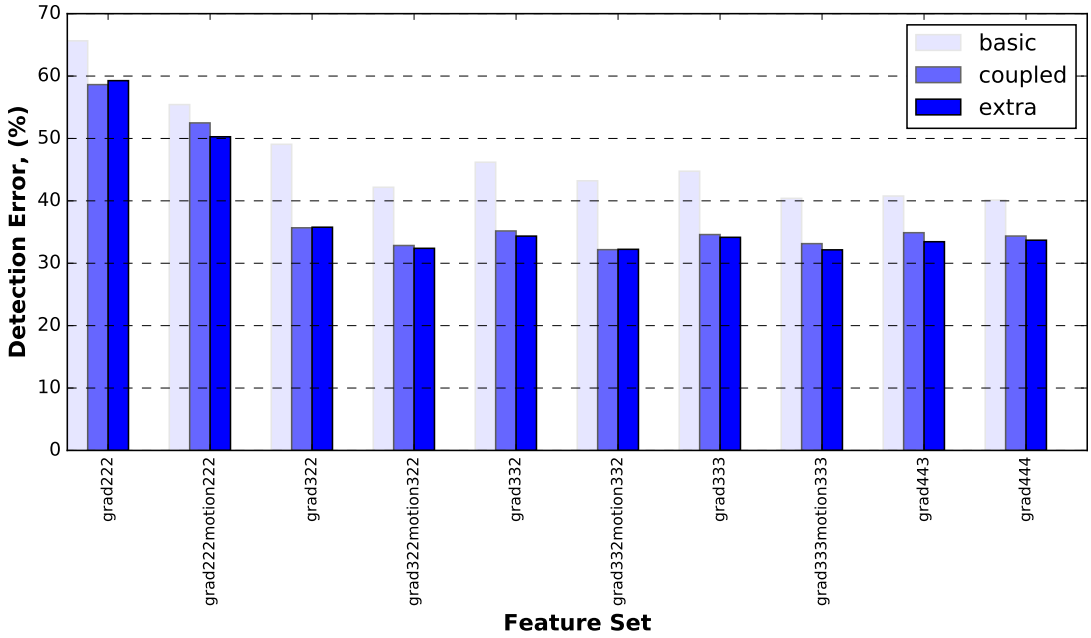


Figure 5.4: Comparison of detection error using the 'basic','coupled' and 'extra' RIF feature sets for a number of different image representations and J, K and M parameters. In each case, the error value shown is the minimum value over all the combinations of trees and levels that were tested.

number of orientations N_o would improve the performance of the rectangular filters, but this would accordingly increase the test time. Note that another important disadvantage of the rectangular filters is the dramatically longer training time due to two factors: the need to train several forest models, one for each orientation, and the larger number of features to consider at each node.

Within the RIF sets, increasing the J, K or M parameters increases the computational time due to the need to calculate more raw features. It can also be seen that the increases in detection performance from larger feature sets drop off quite quickly. Above around $J = 3$, $K = 3$, $M = 2$, there is little further improvement in detection performance.

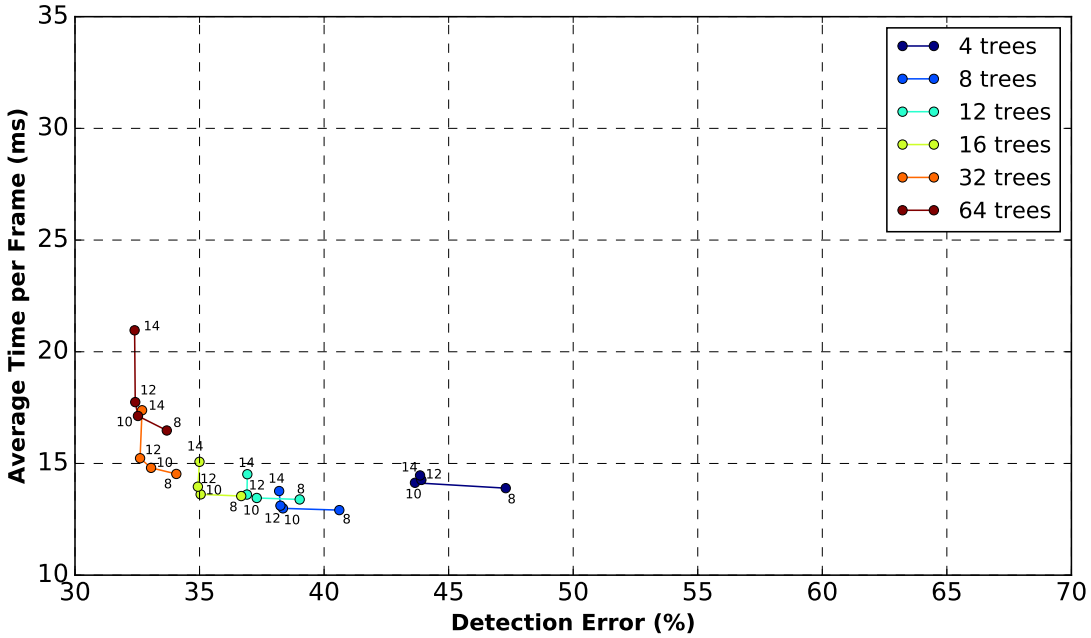
The bar chart in Figure 5.4 compares the performance of RIFs using the 'basic', 'coupled' and 'extra' feature sets. The results suggest that the addition of the coupled features significantly increases the performance of the classifier. The effect of introducing the 'extra' feature set is not particularly clear, the results suggest a small and inconsistent improvement over the coupled set.

Within the rectangular feature sets, the accuracy tends to increase with the number of bins in the orientation histogram N_b, suggesting the larger, more detailed feature sets aid in classification. There is little resulting change to the run time because the complexity of the binning procedure does not depend upon the number of bins.

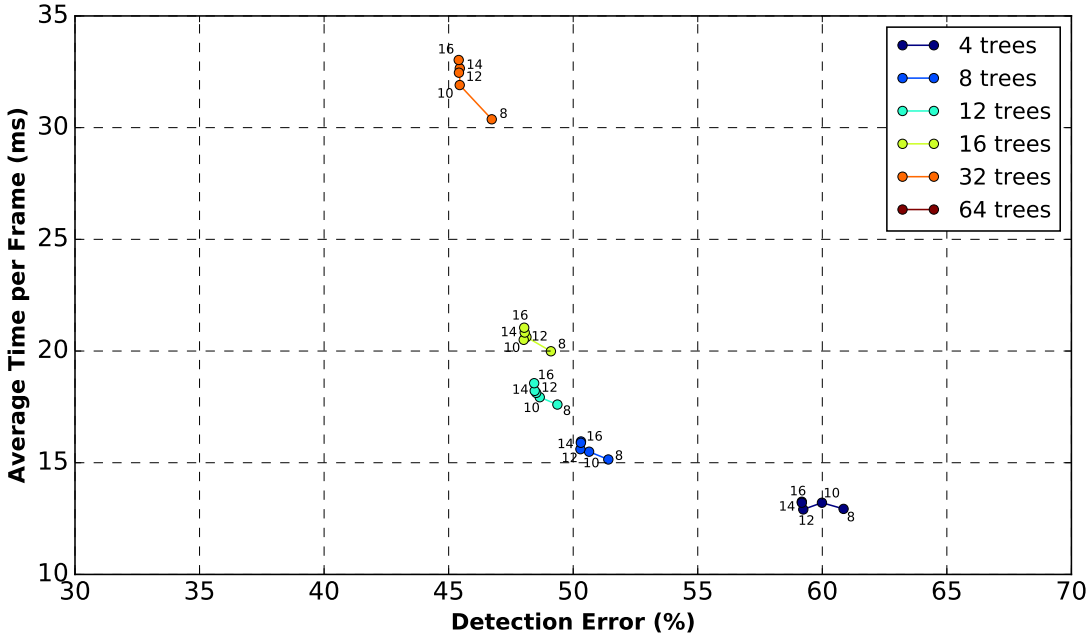
Figure 5.5 shows how the detection accuracy depends on the number of trees and the maximum number of levels in each tree for a promising feature set using RIFs and rectangular features respectively. Other features sets show a similar trend. As expected, increasing the number of trees and the number of levels generally improves the detection accuracy but this saturates at some point in both cases. Also, a larger number of trees or levels reduces the speed of the detection. The plots suggest that values around 32 trees and 10 levels represent a good compromise between accuracy and efficiency, but this varies on the feature set and the relative importance of speed and accuracy.

5.6.3 View Confusion Performance

Figure 5.6 shows confusion matrices for select feature sets and forest parameters, again chosen to show the general trends. These measure the ability of the detection forests to discriminate correctly between the three views of the heart. In these figures, any detection further than the distance threshold of $0.25R$ from the ground truth annotation is considered to be a 'miss', and only detections within this threshold are considered for calculating the confusion matrix between the three view classes. Additionally Cohen's kappa statistic is used to measure the agreement between the automatic output and the manual ground truth within the three heart classes only (i.e. excluding missed detections and frames in which the heart is hidden). The confusion matrices show that the 3V view is the most commonly missed view. This is most likely due to the relatively indistinct appearance of the view (it can appear quite similar to acoustic shadowing artefacts) and the large amount of intra-class variation arising from variation in probe position, partly

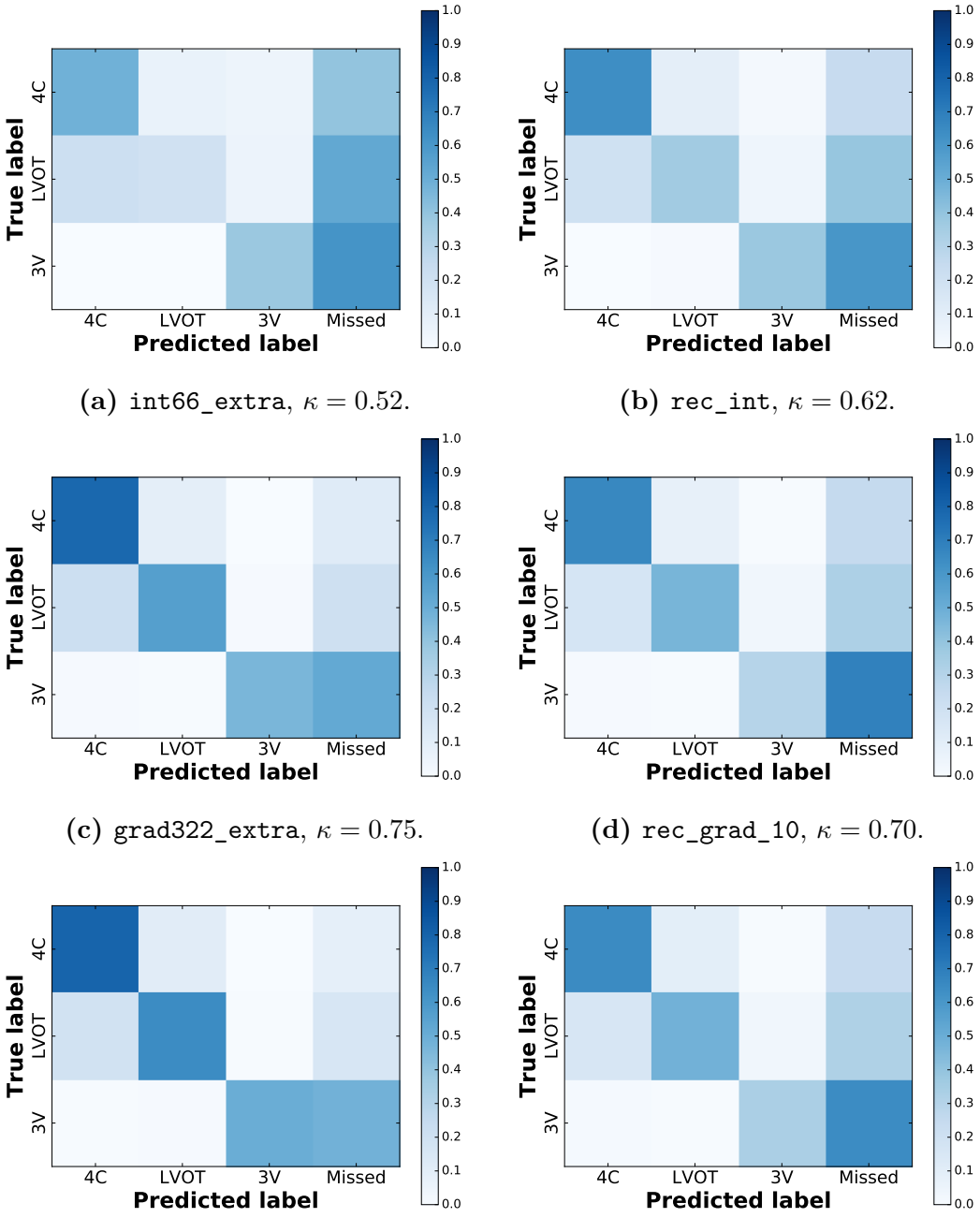


(a) The `grad322motion322_extra` feature set.



(b) The `rec_gradmotion_10` feature set. The points relating to 64 trees are off the scale of the plot.

Figure 5.5: Detection error and calculation time as the number of trees in the `RIFDetection`/`RECDetection` models and the maximum number of levels in the trees are varied. Each coloured line represents a certain number of trees(N_{trees}) in the forest model, and each point on the represents a certain maximum number of levels ($D_{\max}$, written next to the point).



(a) `int66_extra`, $\kappa = 0.52$.

(b) `rec_int`, $\kappa = 0.62$.

(c) `grad322_extra`, $\kappa = 0.75$.

(d) `rec_grad_10`, $\kappa = 0.70$.

(e) `grad322motion322_extra`, $\kappa = 0.78$.

(f) `rec_gradmotion_10`, $\kappa = 0.73$.

Figure 5.6: Confusion matrices for a number of feature extraction methods. Tests were performed using $N_{\text{trees}} = 32$ and $D_{\text{max}} = 10$.

due to the fact that several distinct views have been grouped together to form the 3V category used in this thesis (§1.4.4).

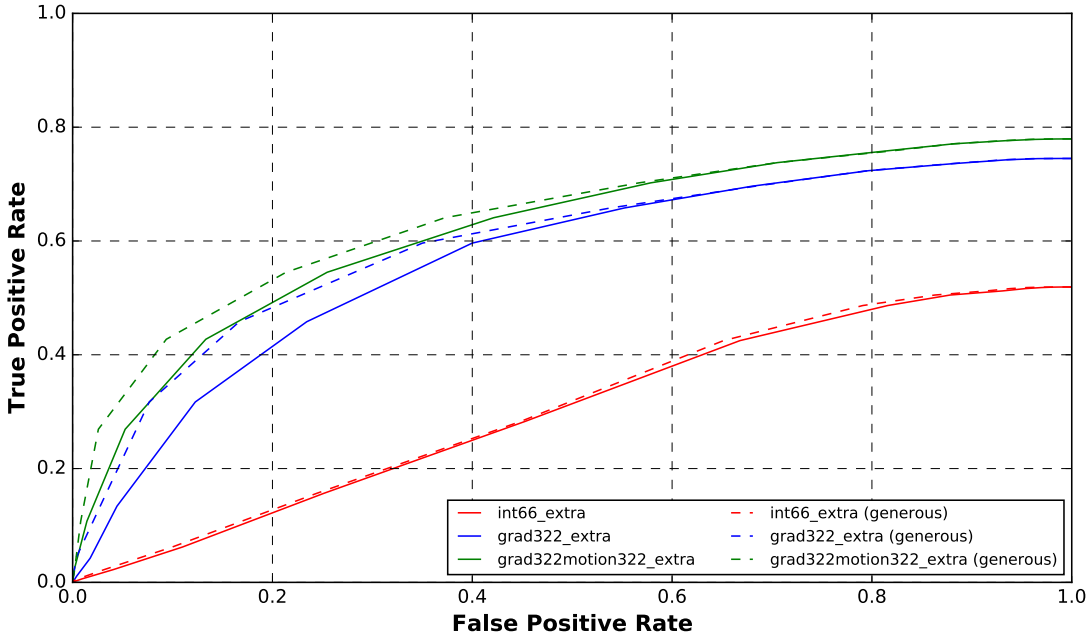
Furthermore, there is significant inter-class confusion between the 4C and LVOT views. This is also unsurprising given the similarity in appearance between the two views, and the existence of an ambiguous region when the probe is located above the 4C and below the LVOT such that a very small part of the aorta and the aortic valve is visible.

5.6.4 Rejection of Negatives

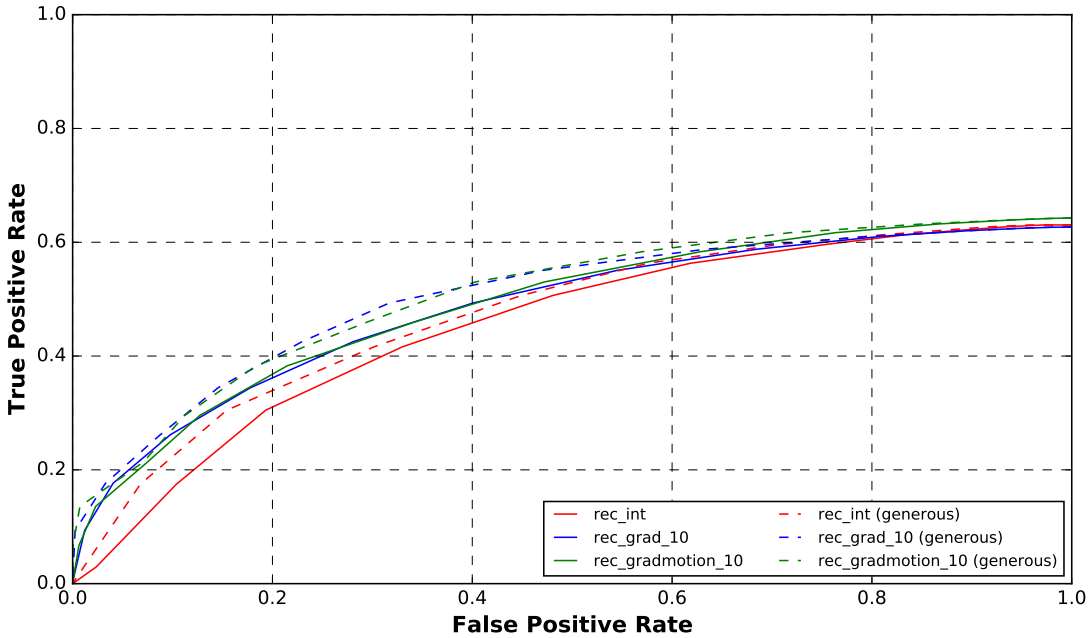
As well as detection performance, it is also important that the classifier is able to correctly reject those frames that do *not* contain a view of the heart. This is done via a threshold on the detection score, as described in §5.3.3.

Figure 5.7 shows the true positive rate and false positive rate (§5.1) as this threshold is varied. The resulting plot is similar to a receiver operating characteristic (ROC) curve, but is not exactly an ROC because the location of the detection is also considered in determining the true positives. Consequently the line does not intersect the top right corner of the axes, because no matter how low the threshold is, the true positive rate will not reach 1 unless the within-frame detection is also perfect (this is a consequence of the fact that the algorithm will only output at most one detection per frame).

It may be seen that performing rejection based on a simple thresholding of the detection score performs quite poorly. In order to achieve a high true positive rate, a high false positive rate must be accepted. The results are improved to some extent if the 'generous' false positive rate metric is considered. This does not penalise the algorithm for making detections in borderline negative cases (those annotated as negative but close to being positive). It is therefore clear that such cases are the cause of some, but not all of the false positives. This motivates the development of a more sophisticated way of determining when the heart is not visible in the image. This is will be explored further in Chapter 6.



(a) A selection of RIF features sets.



(b) A selection of rectangular feature sets.

Figure 5.7: True positive rate versus false positive rate (solid lines) and 'generous' false positive rate (dashed lines) for a selection of feature sets. Testing was performed with $N_{\text{trees}} = 32$ and $D_{\text{max}} = 10$.

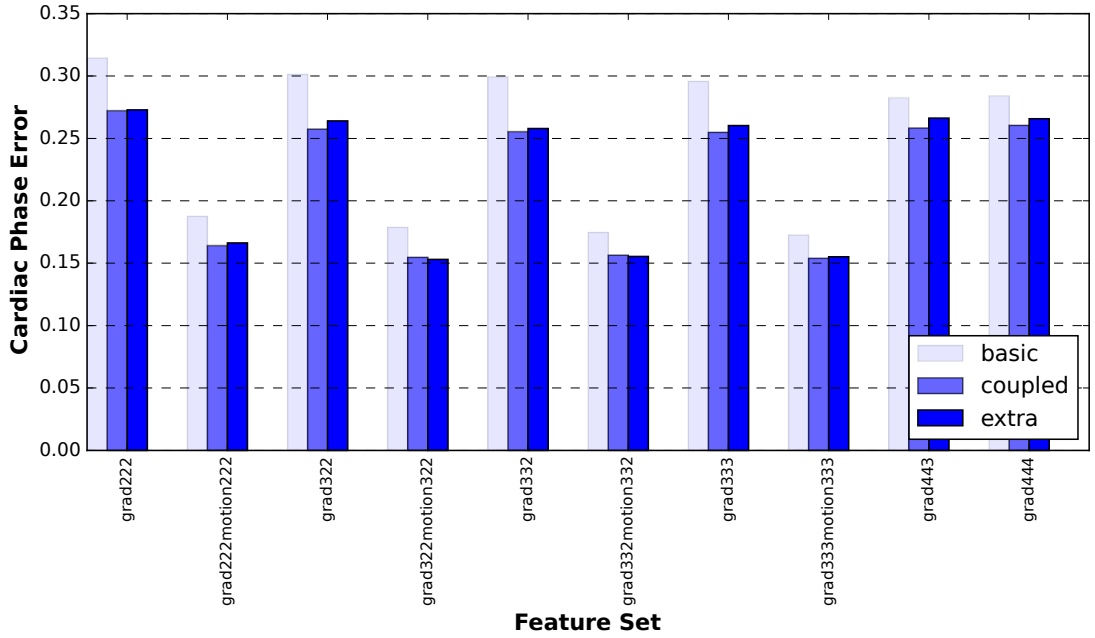


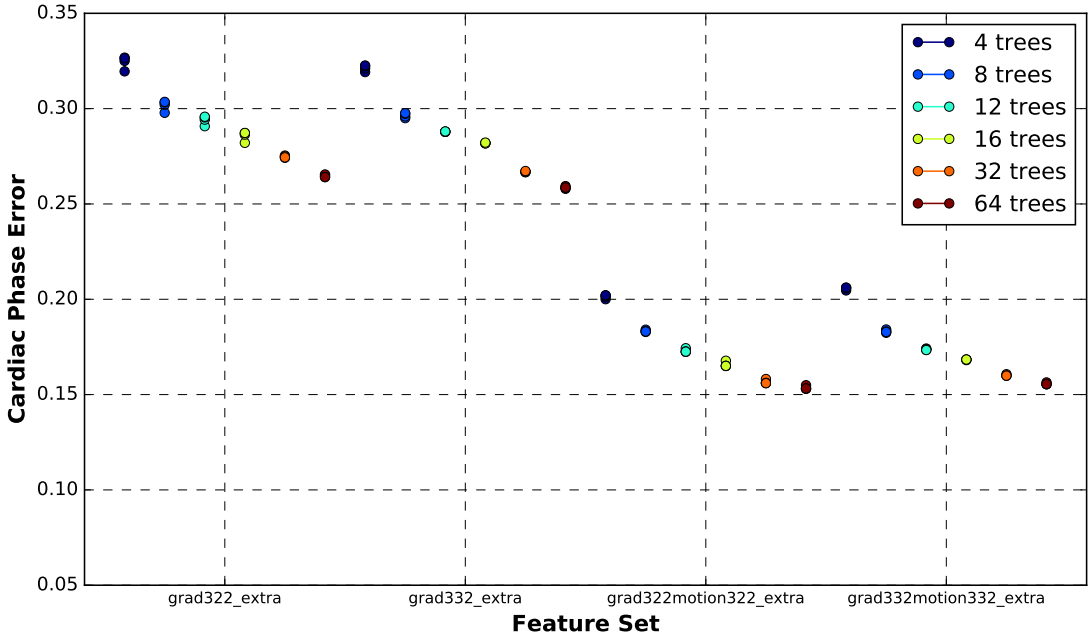
Figure 5.8: Ground truth position cardiac phase error for a number of RIF feature sets. In each case, the value show is the minimum value over all the combinations of trees and levels that were tested.

5.6.5 Orientation and Cardiac Phase Estimation Performance

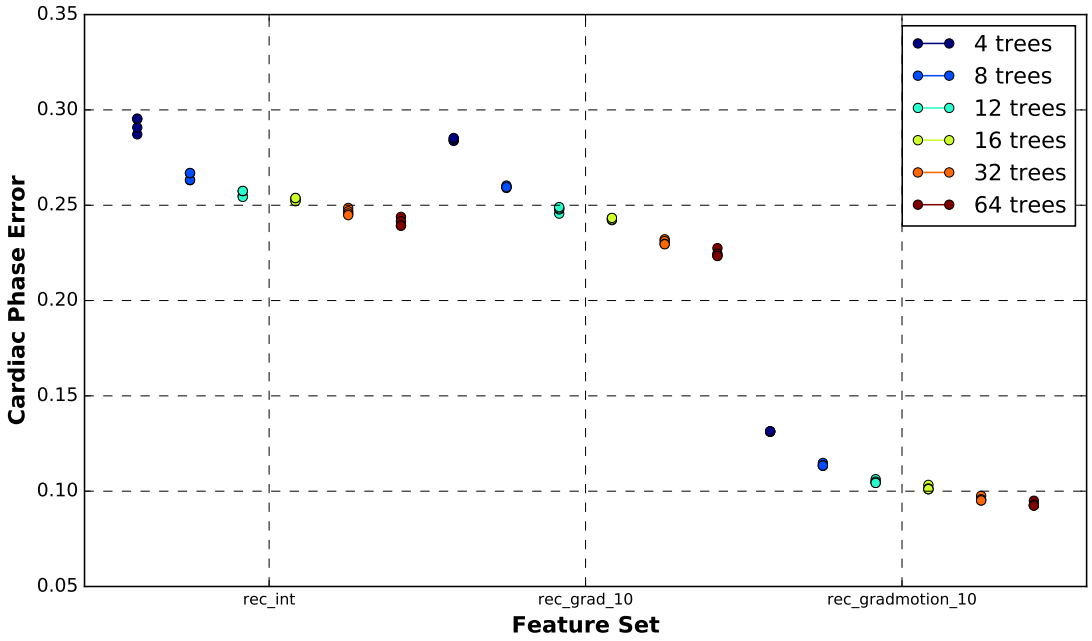
The orientation and cardiac phase estimation performance was analysed in a set of experiments in which the ground truth position orientation error was calculated when the feature sets and number of trees and levels in the relevant forest models were varied. The training parameters were as shown in Table 5.1, and for testing the RIF models, frequency domain feature extraction with automatic coupling was used.

In these experiments, the orientation and cardiac phase estimation was performed at a single location (the ground truth position). Consequently, the speed of the estimation calculation is not a good indicator of the likely speed when estimates are needed at a number of points, as will be required for the particle filtering framework in later chapters. Therefore, the speed of the algorithm was not considered at this stage.

The bar plot in Figure 5.8 shows the ground truth position cardiac phase error for some of the more promising RIF feature sets, and compares the 'basic', 'coupled'



(a) A selection of RIF feature sets.



(b) A selection of rectanglar features sets

Figure 5.9: Ground truth position cardiac phase error as the number of trees in the `RIFPhase`/`RECPhase` model varies. For each number of trees (N_{trees}, shown in the legend), the cardiac phase error is plotted at different values for the maximum numbers of levels ($D_{\max} \in \{8, 10, 12, 14\}$, not labelled for legibility).

and 'extra' features sets. Generally it is observed that the circular regression trees are able to give reasonable regression performance on a challenging problem. The better feature sets are able to achieve an average normalised error of 0.15, which corresponds to just over one tenth of a cycle.

It can be seen that the inclusion of motion features significantly improves the cardiac phase estimation over gradient features alone. This is to be expected, as the motion patterns of the heart walls and valves are key indicators of the cardiac phase. Again it is observed that the use of the 'coupled' feature set improves the performance with respect the using the 'basic' features alone, but that adding the 'extra' features does not significantly change the performance. Note that because the orientation estimation uses the equivariant feature set, it is not meaningful to compare the orientation estimation performance when using the 'basic', 'coupled' and 'extra' sets.

Figure 5.9 shows results for cardiac phase estimation when different numbers of trees and tree levels are used, for a few feature sets. They suggest that for all models, increasing the number of trees improves the prediction performance, and this effect reaches saturation near 64 trees. However changing the number of tree levels (above 8) makes very little difference to the results. This might be due to the fact that the other stopping criteria (§4.1.1) are terminating the training at lower number of levels. It is again clear from Figure 5.9 that the introduction of motion features dramatically improves cardiac phase prediction performance. Also, the rectangular feature sets in general give better cardiac phase estimation performance than the RIF feature sets. This is likely due at least in part to the fact that the cardiac phase estimation model is trained on a known orientation, which in these experiments was assumed to be known exactly at test time (in order to investigate the cardiac phase estimation process separately from the detection process), whereas the RIF models are trained to be rotation-invariant and so cannot use this information. The relative performance of the two when an estimated orientation is used is investigated in Chapter 7.

Figures 5.10 and 5.11 show the orientation error for a selection of feature sets and forest model arrangements. With a large number of trees, the RIFs and rectanglar

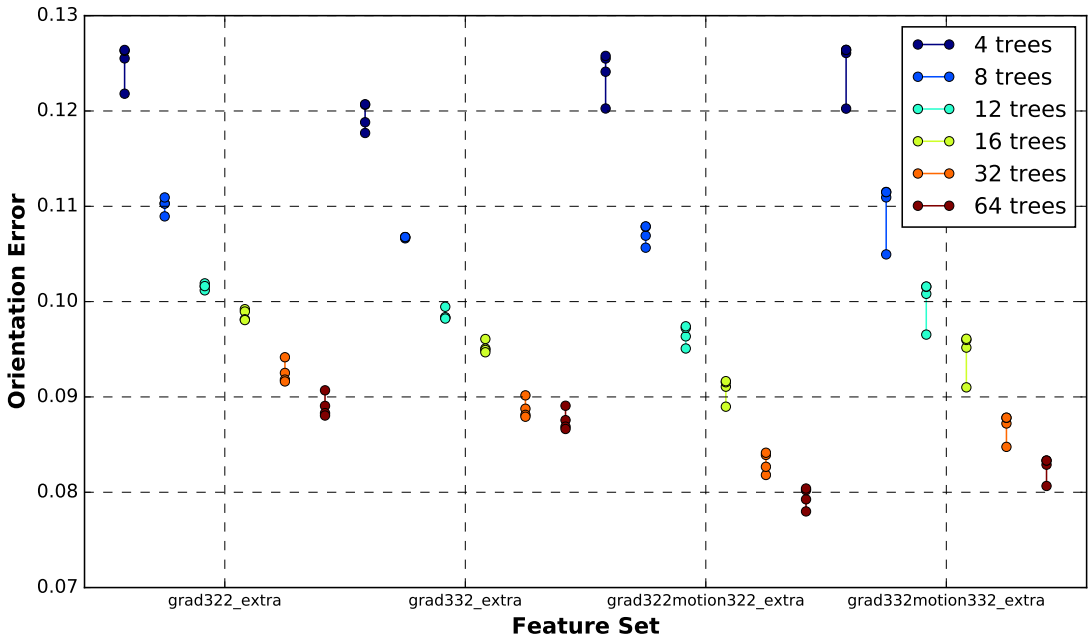


Figure 5.10: Ground truth position orientation error as the number of trees in the `RIFPhase` model (containing the `RIFOri` regressions models) varies. For each number of trees (N_{trees}, shown in the legend), the orientation error is plotted at different values for the maximum numbers of levels ($D_{\max} \in \{8, 10, 12, 14\}$, not labelled for legibility).

features give comparable performance at a normalised error of around 0.08, which corresponds to an angular error of about 30°. The inclusion of motion features is shown to improve orientation estimation results slightly. Unlike the detection and cardiac phase estimation tasks, it appears that increasing the number of trees, N_{trees}, above the values tested could result in further improvements to accuracy in the RIF case. This suggests that performing orientation regression based on the complex phase of a single RIF is rather noisy. Again, altering the number of levels in the trees, $D_{\max}$, is seen to have only a small effect.

5.7 Conclusions

This chapter has presented results for estimation of the variables of interest on a frame-by-frame basis. It was found that the computational efficiency of RIF calculation is dramatically improved by using frequency-domain calculations, making it possible to analyse entire images at around 13 ms per frame, which is fast enough

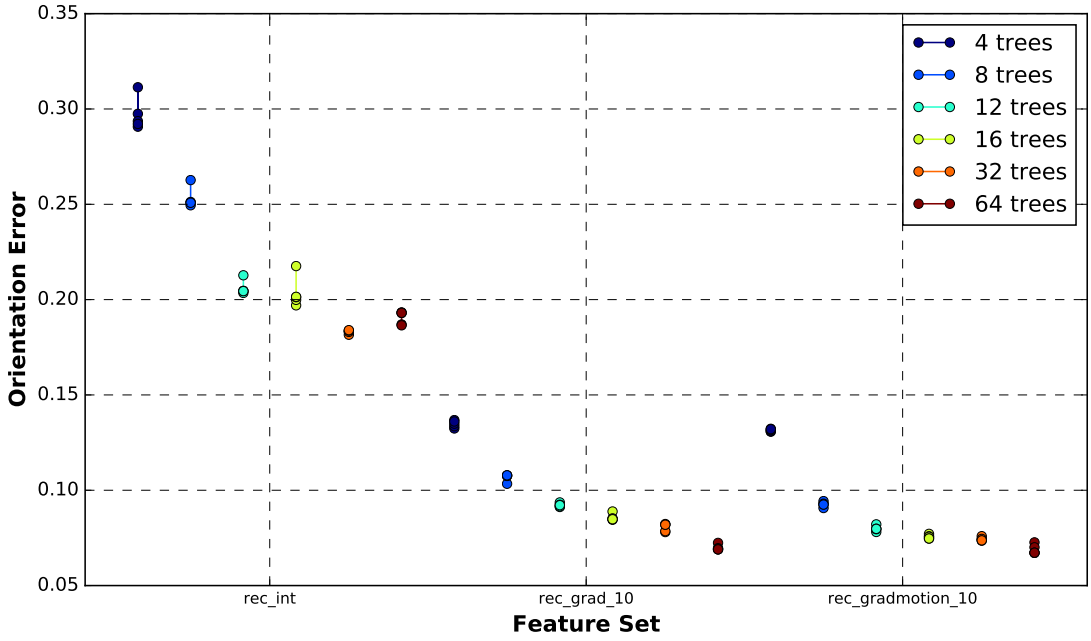


Figure 5.11: Orientation error as the number of trees in the `RECDetection` models varies. For each number of trees (N_{trees}, shown in the legend), the orientation error is plotted at different values for the maximum numbers of levels ($D_{\max} \in \{8, 10, 12, 14\}$, not labelled for legibility). Note that unlike Figure 5.10, the orientation estimation at the estimated heart position (rather than the ground truth position) is used, because the orientation estimation is a by-product of the detection process.

for video analysis tasks. It has shown that the combination of RIFs and random forest models is able to give reasonable and regression performance at these high frame rates, with the best models able to correctly localise and classify the heart views in around 65% of frames, estimate the cardiac phase value with an average error of about 0.1 cycles, and predict the orientation with an average error of around 30°. It was found that RIFs based on both image intensity gradient and optical-flow-based motion features tended to give better performance across all measures than those based on image intensity or intensity gradient alone. Furthermore it was shown that using coupled RIFs is highly beneficial for the accuracy of the estimates, but additionally using the 'extra coupled' features does not significantly alter the results.

The above results also show that the circular regression forest model introduced in Chapter 4 is able to give reasonable estimates of the circular output variable for the challenging problem of cardiac phase estimation task.

The comparison between the RIFs and rectangular features suggest that RIFs perform better for the detection task, approximately the same for the orientation estimation task, and worse for the cardiac phase prediction task than the more common rectangular features. However the latter assumes that the true position and orientation is known exactly as test time, which is unrealistic is practice. This is explored further in Chapter 7.

Although results in this chapter are reasonable for the challenging nature of the task, there is significant room for improvement in the results by taking advantage of the relationships between nearby frames. Chapter 6 will present a method for doing this, which uses the detection and regression models presented in this chapter as a basis for particle filtering framework.

6

Particle Filtering for Tracking the Fetal Heart in Videos

Contents

This chapter describes a methodology for combining framewise predictions across multiple frames of video in order to use contextual information and give consistent predictions. An earlier version of this work is described in an article published in *Medical Image Analysis* [163]. That paper did not use the partitioning of the particle filtering framework.

6.1 Introduction

The experiments performed in Chapter 5 were limited to *framewise* predictions, i.e. treating each frame as entirely independent from all other frames in the video. However, this assumption of independence between frames is very *naïve*, as clearly there are very strong relationships between the values of the state variables in consecutive frames at typical clinical imaging frame rates. The position and orientation of the heart in the image is unlikely to change significantly between consecutive video frames. The timing of transitions between the different viewing planes is unpredictable, but they will be accompanied by predictable changes in the centre and orientation of the heart (because these are defined differently in different viewing planes, see Figure 1.4). Furthermore, for a typical, healthy heart the cardiac phase advances between frames at a fairly constant rate within a known range of anatomical plausibility.

Given how ambiguous and difficult to interpret a single video frame can be, it is logical to use this prior knowledge to inform the estimates in each frame. Furthermore, the most principled approach to this is to use a *probabilistic* model to account, explicitly, for the inherent uncertainty in both the system dynamics and the interpretation of images, and therefore provide robustness to this uncertainty. The rest of this chapter reviews the theory of sequential Bayesian filtering and different ways to implement it (§6.2 and §6.3), and then describes a particle filtering procedure for estimating the state variables from ultrasound videos of the fetal heart, beginning at §6.4.

6.2 Sequential Bayesian Filtering

Bayesian recursive estimation or *sequential Bayesian filtering* is a standard formulation for online estimation of the state of a *dynamic system* given a probabilistic model of the system's behaviour and probabilistic measurements [170, 171].

The state of a system at a given time point[1] $t \in \mathbb{N}_0$ is assumed to be fully described by its *state vector*, $\mathbf{s}_t \in \mathbb{S}$ at that time point, where $\mathbb{S}$ is the *state space*. The system dynamics can be captured by some *state evolution distribution* $p(\mathbf{s}_t \mid \mathbf{s}_{t-1})$ that describes the likelihood of a given state value at time t given the state value at the previous time-step $t-1$, and is conditionally independent of all measurement information and state values before time $t-1$. This is equivalent to assuming that the system dynamics obey the Markov assumption.

The measurement information at a given time point t shall be denoted $\mathbf{z}_t$, and the measurement information at all time points up to and including time t shall be denoted $\mathbf{z}_{0:t}$. $\mathbf{z}_t$ provides information about the value of the state at time t through some generative *measurement distribution* $p(\mathbf{z}_t \mid \mathbf{s}_t)$ that is assumed to be independent of all previous measurements and the system dynamics. This distribution describes the likelihood of seeing the observed measurement given a certain value of the state

The aim of Bayesian filtering is to maintain a probabilistic estimate of the state of the system, $\mathbf{s}_t$, at each time step that takes into account *all* the previous measurement information and the constraints imposed by the dynamic model. This distribution is referred to as the *filtering distribution* and denoted $p(\mathbf{s}_t \mid \mathbf{z}_{0:t})$.

Given the two key distributions (the state evolution distribution and the measurement distribution), Bayesian filtering proceeds by recursively applying the following two equations at each time step, t:

[1] the focus here will be on discrete-time systems as the frames of a video are naturally a discrete-time signal.

$$p(\mathbf{s}_t \mid \mathbf{z}_{0:t-1}) = \int_{\mathbb{S}} p(\mathbf{s}_t \mid \mathbf{s}_{t-1})\, p(\mathbf{s}_{t-1} \mid \mathbf{z}_{0:t-1})\, \mathrm{d}\mathbf{s}_{t-1} \tag{6.1}$$

$$p(\mathbf{s}_t \mid \mathbf{z}_{0:t}) = \frac{p(\mathbf{s}_t \mid \mathbf{z}_{0:t-1})\, p(\mathbf{z}_t \mid \mathbf{s}_t)}{p(\mathbf{z}_t \mid \mathbf{z}_{0:t-1})} \tag{6.2}$$

The first of these (Equation 6.1) represents the *prediction* step. It predicts the distribution over the state at time t given the state evolution model by marginalising out the filtering distribution at the previous time-step, resulting in a *predicted distribution* $p(\mathbf{s}_t \mid \mathbf{z}_{0:t-1})$. The second equation (Equation 6.2) is an *update* step that adjusts the predicted distribution in order to reflect the measurement information. The denominator of this expression is constant with respect to $\mathbf{s}_t$ and therefore in many implementations it is easier to simply note that the filtering distribution is proportional to the numerator, i.e. that

$$p(\mathbf{s}_t \mid \mathbf{z}_{0:t}) \propto p(\mathbf{s}_t \mid \mathbf{z}_{0:t-1})\, p(\mathbf{z}_t \mid \mathbf{s}_t) \tag{6.3}$$

and then ensure that the filtering distribution is valid by normalising it afterwards.

6.2.1 Exact Inference

Although the procedure for Bayesian recursion is simple in principle, there are only a few situations in which it is possible to implement it in a way that results in an exact and computationally tractable inference procedure [170].

One such situation is when the state space is discrete, or can be approximated to sufficient accuracy by a discrete representation. In this case the state evolution distribution can be represented by a matrix of transition probabilities, and the model is commonly referred to as a *Hidden Markov Model* (HMM). The filtering problem for an HMM may be solved with an algorithm variously known as the *forward algorithm* [172] or the histogram filter [170], which is a special case of dynamic programming in which the integral in Equation 6.1 is a discrete sum. Whilst an exact solution is always possible in a Hidden Markov Model, the problem

becomes highly computationally intensive when the state is high-dimensional and/or each state dimension has a large number of possible values.

A second important case where exact inference is possible is the *Kalman Filter* for continuous state spaces [170, 172]. This requires that the state transition model is a linear transformation of the previous state vector with additive Gaussian noise, and the measurement distribution is a point estimate corrupted by additive Gaussian noise. Under these conditions, the filtering distribution is itself Gaussian, and its parameters can be updated in closed form given the parameters of the state evolution and measurement models. This is a consequence of the fact that Gaussian distributions, unlike most probability distributions, are closed under the operations of multiplication and convolution.

The Kalman Filter results in a simple and highly efficient inference algorithm for continuous state spaces. However, the assumptions it makes are very restrictive in many real-world situations. Of particular importance is the fact that the Gaussian distribution cannot represent multi-modal distributions, but in practice many state evolution and measurement distributions are naturally multi-model, especially in computer vision and image analysis. For example, when searching an image for an object of interest, typically a detector will assign a low score to most of the image but a high score to a small number of image locations.

6.2.2 Approximate Inference

In many cases, exact inference for sequential Bayesian filtering is not possible or prohibitively expensive in terms of computational demands. Therefore a range of approximate inference methods are used in practice.

The restrictive assumptions of the Kalman filter have lead to generalisations of the Kalman filter that relax some of these assumptions. The *Extended Kalman Filter* and *Unscented Kalman Filter* are popular models that approximate the Kalman Filter for systems whose state evolution models are non-linear [170, 172]. However the assumptions of Gaussian filtering distributions and measurement distributions remain.

The *Multiple Hypothesis Tracker* assumes that the filtering distribution can be represented by a mixture-of-Gaussians distribution in which each mixture component represents one 'hypothesis' about the state [173]. Each component functions in a similar way to an individual Kalman filter [170].

There are also approximate algorithms based on the Kalman filter for tracking random variables defined on a circular manifold, for example that of Kurz et al. [174], which is similar to the unscented Kalman Filter.

In addition to the above methods of approximate inference there are stochastic Monte Carlo methods, which will be the subject of the next section.

6.3 Monte Carlo Inference and Particle Filters

6.3.1 Basic Particle Filters

Monte Carlo methods form a set of widely-used tools for approximate inference in Bayesian filtering problems [170, 171] due to their high versatility and simplicity to implement. Such methods do not assume a specific parametrised form for the filtering distribution but rather represent it as a finite set of weighted samples (known as *particles*) drawn from it. Specifically, the filtering distribution is approximated by the sum of Dirac delta-distributions, $\delta(\cdot)$, at locations defined by a set of P particles, $\{\mathbf{s}_t^{(i)}\}_{i=1}^P$, $\mathbf{s}_t^{(i)} \in \mathbb{S}$, and their associated *importance weights* $\{w_t^{(i)}\}_{i=1}^P$.

$$p(\mathbf{s}_t \mid \mathbf{z}_{0:t}) \approx \sum_{i=1}^{P} w_t^{(i)} \delta(\mathbf{s}_t - \mathbf{s}_t^{(i)}) \tag{6.4}$$

This represents a valid probability distribution provided that $w_t^{(i)} \in [0, 1]$ and $\sum_{i=1}^P w_t^{(i)} = 1$. As the number of samples, P, increases the particle set more closely approximates the true filtering distribution.

The basic particle filtering algorithm is the *Sequential Importance Sampling* (SIS) algorithm, outlined in Algorithm 6.1. The prediction step (Equation 6.1) proceeds by sampling a new value for each particle from the state evolution distribution. This requires that a suitable sampling algorithm exists for sampling from the state evolution model.

Input: a particle set $\{\mathbf{s}_{t-1}^{(i)}\}_{i=1}^{P}$ and importance weights $\{w_{t-1}^{(i)}\}_{i=1}^{P}$ at time $t-1$
Output: an updated particle set $\{\mathbf{s}_{t}^{(i)}\}_{i=1}^{P}$ and importance weights $\{w_{t}^{(i)}\}_{i=1}^{P}$ at time t

{*For all particles*}
for $i \in \mathbb{N}_{1,P}$ **do**
{*Sample from the state evolution distribution*}
$\mathbf{s}_t^{(i)} \leftarrow \mathbf{s}_t^{(i)} \sim p(\mathbf{s}_t \mid \mathbf{s}_{t-1}^{(i)})$
{*Update weight according to measurement distribution*}
$w_t^{(i)} \leftarrow w_{t-1}^{(i)}\, p(\mathbf{z}_t \mid \mathbf{s}_t^{(i)})$
end for

{*Find the sum of the updated weights*}
$W_t \leftarrow \sum_{i=1}^{P} w_t^{(i)}$

{*Re-normalise the importance weights*}
for $i \in \mathbb{N}_{1,P}$ **do**
$w_t^{(i)} \leftarrow \frac{w_t^{(i)}}{W_t}$
end for

Algorithm 6.1: The Sequential Importance Sampling particle filtering algorithm

The update step (Equation 6.2) proceeds by updating the sample weights according to the measurement distribution. This does not take into account the denominator of Equation 6.2, so in order to ensure that the new filtering distribution represents a valid probability distribution the weights must be renormalised afterwards to complete the update step.

The performance of Sequential Importance Sampling deteriorates after a few steps because most of the importance weights become close to zero, and the majority of the probability mass becomes concentrated on a small number of particles with large weights. This is known as the *degeneracy problem*, and is particularly severe if there is large uncertainty in the state evolution model, because only a few particles will be close to the peaks of the measurement distribution. The problem is that although the particle set still represents the correct filtering distribution (in the limit of infinite particles), it does so with decreasing *efficiency*. The efficiency of a particle set is a measure of how many particles are needed to approximate the distribution [175]. An efficient particle set has many particles concentrated at areas

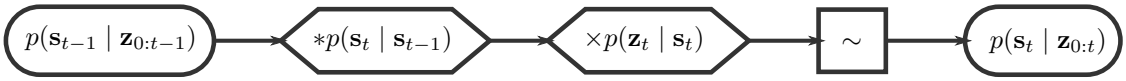


Figure 6.1: Filtering diagram for the basic particle filter following the convention of MacCormick et al. [175, 176]. The round edged boxes represent distributions in the form of particle sets. The hexagonal boxes represent operations on each particle in the particle set: the '$*$' represents convolving the particle set with the state evolution distribution and the '$\times$' represents multiplying the particle weights by the measurement distribution. The square box containing the '$\sim$' represents the resampling operation across the entire particle set.

of high probability and few particles at areas of low probability, or equivalently has a more equal distribution of weights among the particle set.

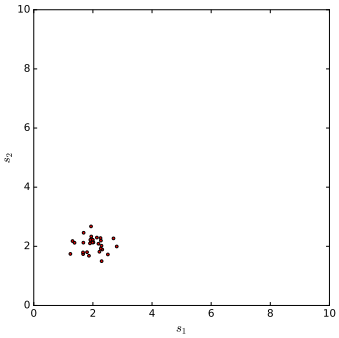
This is overcome by adding a *resampling* step to the SIS algorithm in which a new particle set is created from the old particle set by resampling particles according to their importance weights, and then resetting the weights to a uniform value. This does not change the distribution that the particle set approximates, but by concentrating the particles at areas of high probability, it increases the efficiency with which it approximates it. The resulting algorithm is called the Sequential Importance Resampling (SIR) algorithm, and is outlined in Algorithm 6.2 and illustrated in Figure 6.2.

Following MacCormick et al. [175], Figure 6.1 shows a schematic representation of the steps in each iteration of the SIR algorithm.

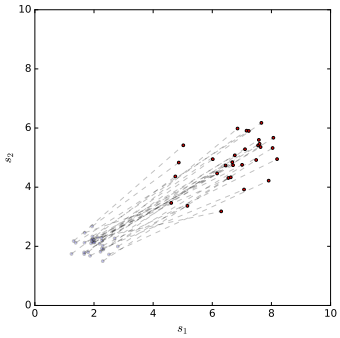
The disadvantage of the SIR algorithm is that each resampling step reduces the *diversity* of the particle set, as many particles are identical immediately after the resampling step (Figure 6.2). Some formulations do not resample on every step, but resample only when the variance of the particle weights exceeds some threshold [170] in order to avoid reducing the diversity of particles unnecessarily.

6.3.2 Conditional Random Field Filters (CRF-Filters)

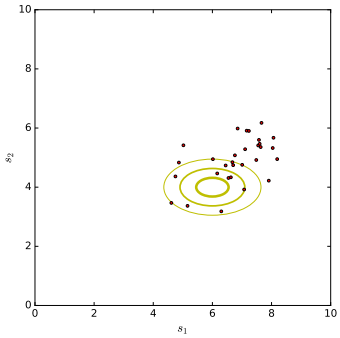
Notice that in §6.3.1 the measurement distribution $p(\mathbf{z}_t \mid \mathbf{s}_t)$ has the form of a generative model, i.e. one that models the likelihood of the observed measurement given a certain value of the state. This is a natural formulation for many simple



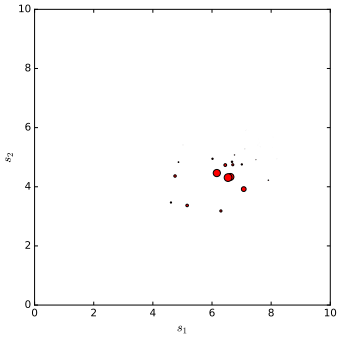
(a) The initial distribution of particles, with uniform weights.



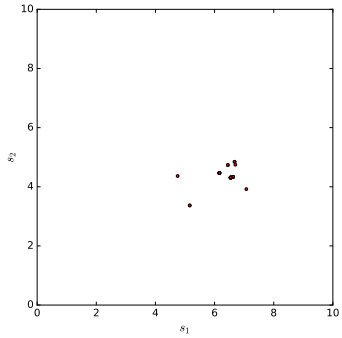
(b) A new state value for each particle is sampled from the state evolution distribution.



(c) The Gaussian measurement distribution is shown by its contours (yellow) at 1, 2, and 3 standard deviations from the mean.



(d) The importance weights are updated according to the measurement distribution.



(e) The particles are resampled to concentrate them at areas of high probability. Note that several particles appear on top of each other.

Figure 6.2: Illustration of a single time-step of the SIR filtering algorithm for particle filtering. Here there is a 2D state vector $\mathbf{s} = [s_1, s_2]$, a Gaussian state evolution distribution and an axis-aligned Gaussian measurement distribution representing a point estimate corrupted by noise. In all images, the current particle set is represented by the red particles and the blue particles (where shown) represent the previous particle set. The radius of the circle representing a particle is proportional to its importance weight.

Input: a particle set $\{\mathbf{s}_{t-1}^{(i)}\}_{i=1}^{P}$ and importance weights $\{w_{t-1}^{(i)}\}_{i=1}^{P}$ at time $t-1$
Output: an updated particle set $\{\mathbf{s}_{t}^{(i)}\}_{i=1}^{P}$ and importance weights $\{w_{t}^{(i)}\}_{i=1}^{P}$ at time t

{*For all particles*}
for $i \in \mathbb{N}_{1,P}$ **do**
  {*Sample from the state evolution distribution and store in a temporary list*}
  $\hat{\mathbf{s}}_t^{(i)} \leftarrow \hat{\mathbf{s}}_t^{(i)} \sim p(\mathbf{s}_t \mid \mathbf{s}_{t-1}^{(i)})$
  {*Update weight according to measurement distribution*}
  $w_t^{(i)} \leftarrow w_t^{(i)}\, p(\mathbf{z}_t \mid \hat{\mathbf{s}}_t^{(i)})$
end for

{*Find the sum of the updated weights*}
$W_t \leftarrow \sum_{i=1}^{P} w_t^{(i)}$

{*Re-normalise the importance weights*}
for $i \in \mathbb{N}_{1,P}$ **do**
  $w_t^{(i)} \leftarrow \frac{w_t^{(i)}}{W_t}$
end for

{*Resample the particle set*}
for $i \in \mathbb{N}_{1,P}$ **do**
  {*Draw a resampling index for this particle according to importance weights*}
  $j \leftarrow j \sim \mathcal{D}(j \mid \mathbf{w}_t)$
  {*Copy the particle with this index from the temporary list*}
  $\mathbf{s}_t^{(i)} \leftarrow \hat{\mathbf{s}}_t^{(j)}$
end for

{*Reset the weights*}
for $i \in \mathbb{N}_{1,P}$ **do**
  $w_t^{(i)} \leftarrow \frac{1}{P}$
end for

Algorithm 6.2: The Sequential Importance Resampling particle filtering algorithm

applications in which the measurement process can be modelled straightforwardly, for example when a direct point measurement of the state value is corrupted by a simple noise process.

However in many situations the relationship between the measurement information and the state value that gave rise to it is much less straightforward. For example, when tracking an object in a video stream some detection procedure

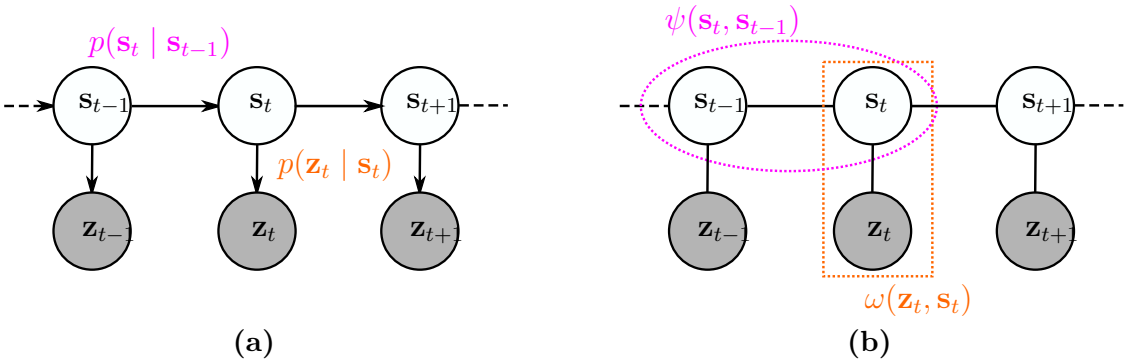


Figure 6.3: Graphical structures of (a) a sequential Bayesian filter and (b) a conditional random field filter. The former is a directed graphical model with the edges representing valid conditional probability distributions. The latter is an undirected graphical model with non-negative clique potentials defined between the graph nodes [177].

must first be applied to the image, which might result in multiple hypotheses for the object's location in the image. Furthermore there is likely to be 'clutter' or information in the image that does not depend on the state of the object of interest.

In these cases, creating a full *generative* model for the image given the state $p(\mathbf{z}_t \mid \mathbf{s}_t)$ is unnecessary and likely to be highly complex. Instead a more natural approach is to construct a *discriminative* model $p(\mathbf{s}_t \mid \mathbf{z}_t)$ that directly predicts the likelihood of a certain state value given the observed measurement. The latter approach allows the design of the model to focus on the relevant task – that of predicting the state given the image – rather than unnecessary modelling of the full imaging process.

To reflect this change of approach, Limketkai et al. [177] introduced the *Conditional Random Field Filter* (CRF-Filter), named to reflect the analogy with conditional random field models within image analysis.

The model architecture for the CRF-Filter and comparison with a standard sequential Bayesian filter is shown in Figure 6.3. The CRF Filter architecture is an undirected graphical model in which two clique potentials are defined: a *prediction* potential $\psi(\mathbf{s}_t, \mathbf{s}_{t-1})$, and an *observation potential* $\omega(\mathbf{z}_t, \mathbf{s}_t)$. The prediction potential captures the system dynamics by measuring the compatibility of the current state and the previous state and therefore acts very much its counterpart in the traditional sequential Bayesian filter, the *state evolution model*. The *observation potential*

processes the measurement information and measures the compatibility of this information with a given state estimate. It therefore is the counterpart to the *measurement distribution* in the traditional sequential Bayesian filter.

This formulation is attractive because it is more general than the original sequential Bayesian filter formulation. In order to use particle filtering to perform inference in the model, it is necessary to be able to sample from the *prediction* potential, much like in the traditional particle filtering case. For this reason the notation $\psi(\mathbf{s}_t \mid \mathbf{s}_{t-1})$ will also be used, as the expression must represent a valid probability distribution. However, in the CRF-Filter formulation, the *observation* potential can be any non-negative function of its arguments. Provided that this condition is met, the Hammersley-Clifford theorem states that the graphical model will still represent a valid probability distribution [172]. This affords far greater modelling flexibility to design advanced, discriminative measurement models without being concerned about the function being a valid generative probability distribution as required in traditional sequential Bayesian filter.

Monte Carlo inference in the CRF Filter model using SIS or SIR particle filtering proceeds exactly as in Algorithms 6.1 and 6.2 respectively, with the state evolution distribution $p(\mathbf{s}_t \mid \mathbf{s}_{t-1})$ replaced with the prediction potential $\psi(\mathbf{s}_t, \mathbf{s}_{t-1})$ and the measurement distribution $p(\mathbf{z}_t \mid \mathbf{s}_t)$ replaced with the observation potential $\omega(\mathbf{z}_t, \mathbf{s}_t)$.

Consequently, the differences between the two approaches are mostly theoretical insofar as they wil be used in this thesis. In particular, one could adapt the sequential Bayesian filter approach to use a discriminative model by relating $p(\mathbf{z}_t \mid \mathbf{s}_t)$ and $p(\mathbf{s}_t \mid \mathbf{z}_t)$ via Bayes rule, and assuming a constant prior $p(\mathbf{s}_t)$. Doing so would arrive at an identical filtering procedure. However in this thesis the conditional random field presentation will be preferred as it obviates the need to justify the observation models as reflecting valid probability distributions.

6.3.3 Partitioned Particle Filters

The concept of *survival rate* [175] is key to analysing the performance of particle filters. The survival rate is defined as the fraction of particles that are expected to

survive each resampling stage. A filter has a high survival rate if the probability mass in the state evolution distribution is highly concentrated and/or the probability mass in the measurement distribution is sparsely distributed such that after the prediction step the majority of particles are concentrated in the area of high probability according to the measurement distribution and therefore receive high weights.

The concept is formalised by MacCormick et al. [175], who define the survival rate, α, for a filter as follows. Given the *predicted distribution* $p(\mathbf{s}_t \mid \mathbf{z}_{0:t-1})$ (after the prediction step in Equation 6.1) and the *filtering distribution* $p(\mathbf{s}_t \mid \mathbf{z}_{0:t})$ (after the update step in Equation 6.2) can be found as follows:

$$\alpha = \left(\int_{\mathbb{S}} \frac{p(\mathbf{s}_t \mid \mathbf{z}_{0:t})^2}{p(\mathbf{s}_t \mid \mathbf{z}_{0:t-1})} \, \mathrm{d}\mathbf{s} \right)^{-1} \tag{6.5}$$

Maintaining a high survival rate is important in order to maintain a diverse set of particles, which in turn reduces the probability of the particle filter 'missing' areas of high probability in the filtering distribution. If a particle filter has a low survival rate, a large number of particles must be used in order to have an effective filter. This is a particularly acute problem when the state dimension is large because the areas of high measurement likelihood tend to represent a smaller proportion of the state space.

In order to reduce this problem for high-dimensional filters, MacCormick et al. [175, 176] introduced the *partitioned particle filter*[2]. The fundamental idea here is to group the variables that form the state vector into *partitions* that can be considered in sequence, and then performing the particle filtering algorithm on each partition in turn. This technique works well in situations where certain independence assumptions can be made about the state evolution and measurement distributions in each partition, and in such situations can improve the survival rate such that fewer particles need to be used.

Consider a state vector $\mathbf{s}_t = \mathbf{s}_{1:Q,t} = [\mathbf{s}_{1,t}^T, \mathbf{s}_{2,t}^T, \ldots, \mathbf{s}_{Q,t}^T]^T$ that has been split into $Q \in \mathbb{N}$ partitions (note that here the first subscript denotes the partition index

[2]The presentation of partitioned particle filters by MacCormick et al. [175] is slightly more general than that presented in this thesis. For reasons of conciseness, the version presented here corresponds to the *articulated model* special case in §3.3 of that paper.

and the second denotes the time-step). The colon notation in the subscript $\mathbf{s}_{n:m,t}$ denotes the concatenation of the variables in all partitions between indices n and m inclusive (at time-step t). A partitioned particle filter can be constructed that updates the particle set one partition at a time, with partition 1 first and partition Q last, provided that the following conditions are satisfied:

- The state evolution model $p(\mathbf{s}_{1:Q,t} \mid \mathbf{s}_{1:Q,t-1})$ can be rewritten using the probability chain rule as the product of a state evolution model for each partition, where the state evolution model for a given partition is independent of the values of later partitions (but not necessarily independent of earlier partitions) at the same time-step, i.e.

$$p(\mathbf{s}_{1:Q,t} \mid \mathbf{s}_{1:Q,t-1}) = p(\mathbf{s}_{1,t} \mid \mathbf{s}_{1:Q,t-1}) \prod_{q=2}^{Q} p(\mathbf{s}_{q,t} \mid \mathbf{s}_{1:q-1,t}\,, \mathbf{s}_{1:Q,t-1}) \tag{6.6}$$

  This factorisation allows the update step for the first partition to occur independently of the second partition and so on. Note that in the Equation 6.6, the first term in the product (corresponding to the first partition) has been written separately as it does not depend on the value of any other partition at time t.

- The measurement distribution may be factorised into a form that can be evaluated for each partition independently of the value of later partitions, i.e.

$$p(\mathbf{z}_t \mid \mathbf{s}_{1:Q,t}) = \prod_{q=1}^{Q} p(\mathbf{z}_t \mid \mathbf{s}_{1:q,t}) \tag{6.7}$$

If these conditions (Equations 6.6 and 6.7) are satisfied, then the SIR algorithm for partitioned particle filtering proceeds by applying the state evolution model, measurement model and resampling step to each partition in turn. The procedure is given in Algorithm 6.3 and a schematic representation is shown in Figure 6.4.

The key to the performance of partitioned particle filters is the improved survival rate of the particles. Suppose a partitioned filter is factorised into Q partitions each with survival rate α_q. In this case, if the standard SIR algorithm were used,

Input: a particle set $\{\mathbf{s}_{t-1}^{(i)}\}_{i=1}^{P}$ and importance weights $\{w_{t-1}^{(i)}\}_{i=1}^{P}$ at time $t-1$
Output: an updated particle set $\{\mathbf{s}_{t}^{(i)}\}_{i=1}^{P}$ and importance weights $\{w_{t}^{(i)}\}_{i=1}^{P}$ at time t

$\{$*Create a list to track the index of 'ancestor' of each particle from time* $t-1\}$
for $i \in \mathbb{N}_{1,P}$ **do**
  $k_{0,i} \leftarrow i$
end for
$\{$*For all partitions*$\}$
for $q \in \mathbb{N}_{1,Q}$ **do**
  $\{$*For all particles*$\}$
  for $i \in \mathbb{N}_{1,P}$ **do**
    $\{$*Create a copy of all partitions*$\}$
    $\hat{\mathbf{s}}_{t}^{(i)} \leftarrow \mathbf{s}_{t}^{(i)}$
    $\{$*Sample from the state evolution distribution for current partition*$\}$
    $\hat{\mathbf{s}}_{q,t}^{(i)} \leftarrow \hat{\mathbf{s}}_{q,t}^{(i)} \sim p(\mathbf{s}_{q,t} \mid \mathbf{s}_{1:q-1,t}^{(i)}, \mathbf{s}_{1:Q,t-1}^{(k_{q-1,i})})$
    $\{$*Update weight according to the measurement distribution for this partition*$\}$
    $w_{t}^{(i)} \leftarrow w_{t}^{(i)}\, p(\mathbf{z}_t \mid \hat{\mathbf{s}}_{1:q,t}^{(i)})$
  end for
  $\{$*Find the sum of the updated weights*$\}$
  $W_t \leftarrow \sum_{i=1}^{P} w_{t}^{(i)}$
  $\{$*Re-normalise the importance weights*$\}$
  for $i \in \mathbb{N}_{1,P}$ **do**
    $w_{t}^{(i)} \leftarrow \frac{w_{t}^{(i)}}{W_t}$
  end for
  $\{$*Resample*$\}$
  for $i \in \mathbb{N}_{1,P}$ **do**
    $\{$*Draw a resampling index for this particle according to importance weights*$\}$
    $j \leftarrow j \sim \mathcal{D}(j \mid \mathbf{w}_t)$
    $\{$*Resample all partitions according to this index*$\}$
    $\mathbf{s}_{t}^{(i)} \leftarrow \hat{\mathbf{s}}_{t}^{(j)}$
    $\{$*Update the ancestor index for this particle*$\}$
    $k_{q,i} \leftarrow k_{q-1,j}$
  end for
  $\{$*Reset the weights*$\}$
  for $i \in \mathbb{N}_{1,P}$ **do**
    $w_{t}^{(i)} \leftarrow \frac{1}{P}$
  end for
end for

Algorithm 6.3: The Sequential Importance Resampling particle filtering algorithm for a partitioned particle filter. The algorithm is similar to Algorithm 6.2, but more 'book-keeping' is necessary due to multiple resampling stages per time-step.

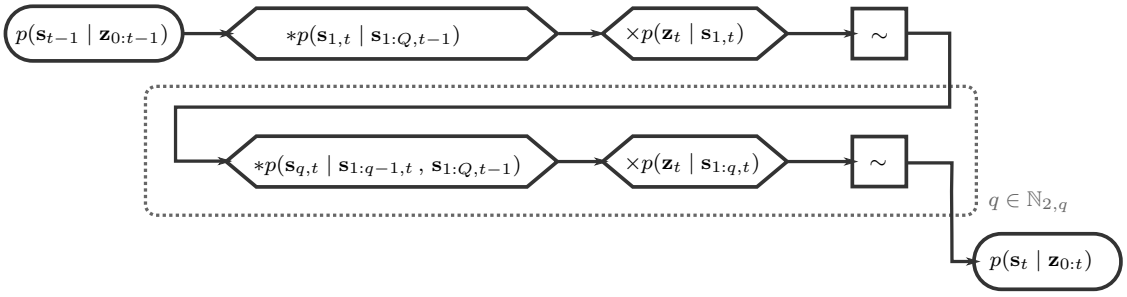


Figure 6.4: Filtering diagram for the partitioned particle filter following the convention of Figure 6.1. There is now one state evolution, re-weighting and resampling step for each of the Q partitions.

the overall survival rate would be $\prod_{q=1}^{Q} \alpha_q$. However, when the partitioned filtering algorithm is used, this is replaced by Q resampling processes, each of survival rate α_q, with the dynamic models being applied in between. Consequently, the particle set is more concentrated at areas of high likelihood throughout the process, see Figure 6.5 for an example of this.

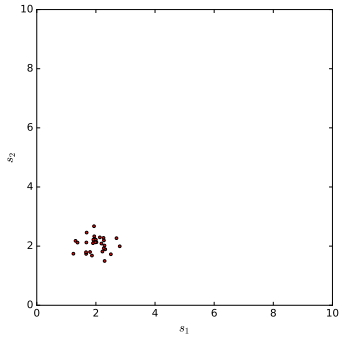
A CRF-Filter may be partitioned in exactly the same way. In this case, the two conditions on the state evolution and measurement distributions (Equations 6.6 and 6.7) are replaced with the conditions on the prediction and observation potentials in Equations 6.8 and 6.9 respectively.

$$\psi(\mathbf{s}_{1:Q,t} \mid \mathbf{s}_{1:Q,t-1}) = \prod_{q=1}^{Q} \psi(\mathbf{s}_{1:q,t} \mid \mathbf{s}_{1:Q,t-1}) \tag{6.8}$$

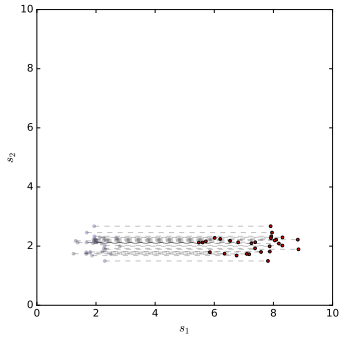
$$\omega(\mathbf{z}_t\,,\, \mathbf{s}_{1:Q,t}) = \prod_{q=1}^{Q} \omega(\mathbf{z}_t\,,\, \mathbf{s}_{1:q,t}) \tag{6.9}$$

6.4 Filtering Architectures for Heart Tracking

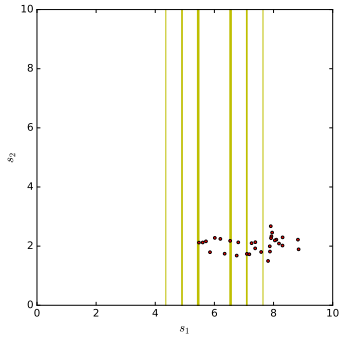
Having covered the relevant background material in §6.2 and §6.3, the rest of this chapter will develop a particle-filtering based model for tracking the global state variables in fetal heart ultrasound videos. In this section, an overview of two model architectures for this purpose is presented. The state in both cases is represented by the tuple containing the heart's visibility, position, view, orientation, cardiac phase,



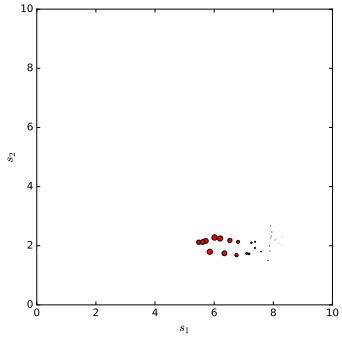
(a) The initial distribution of particles, with uniform weights.



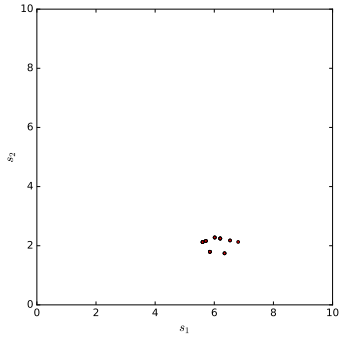
(b) A new s_1 value for each particle is sampled from the state evolution distribution for s_1.



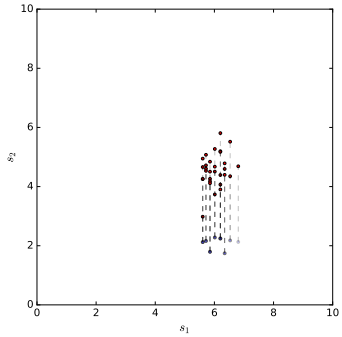
(c) The Gaussian measurement distribution for s_1 is shown by its contours (yellow) at 1, 2, and 3 standard deviations from the mean.



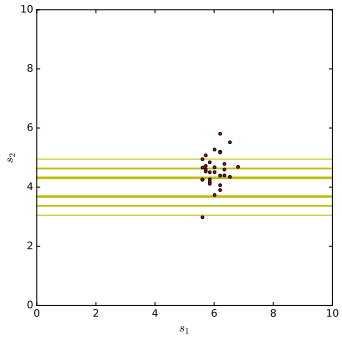
(d) The importance weights are updated according to the measurement distribution for s_1.



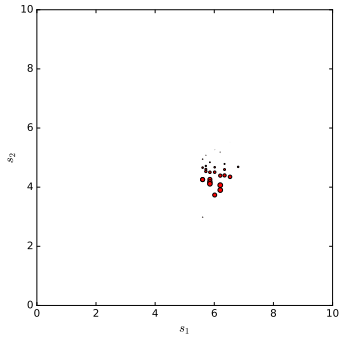
(e) The particles are resampled to concentrate them at areas of high probability. Note that several particles appear on top of each other.



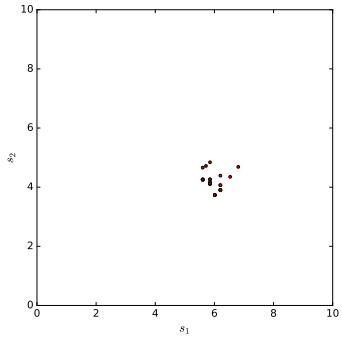
(f) A new state value for s_2 for each particle is sampled from the state evolution distribution for s_2.



(g) The Gaussian measurement distribution for s_2 is shown by its contours (yellow) at 1, 2, and 3 standard deviations from the mean.



(h) The importance weights are updated according to the measurement distribution for s_2.



(i) The particles are resampled to concentrate them at areas of high probability. Note that several particles appear on top of each other.

Figure 6.5: Illustration of a single time-step of the SIR filtering algorithm for partitioned filtering. The model is the same as that in Figure 6.2 except that now the each of the state variables, s_1 and s_2 occupies its own partition. Note how resampling between the partitions concentrates the particles in high probability areas throughout the process.

and cardiac phase rate, as defined in Equation 1.1, and the aim is to estimate these state variables from the video sequence as each frame is processed.

The filters are CRF-Filters that use the random forest models described in Chapter 4 as the observation potentials as they represent a powerful and efficient way to assess different hypotheses about the state without a full generative model of the imaging process. The prediction potentials in the filter are chosen to capture prior anatomical knowledge about the fetal heart and its appearance in ultrasound videos.

The factorisation of the filter into partitions is very natural in this case. The observation potentials described in previous chapters make predictions about the state variables independently. Moreover, it is natural to assume that changes in heart position and view are independent from changes in orientation and cardiac phase. Consequently, a partitioned filter is used to exploit this natural partitioning of the state in order to improve the filter efficiency.

The two different architectures presented below (and shown schematically in Figure 6.6) differ in their treatment of the orientation state variable. The first, `RIFFilter`, uses rotation-invariant features (Chapter 3), and is therefore able to assess position, view class and cardiac phase independently of orientation. This allows the orientation state variable to be placed in a separate partition after the other variables, giving a total of three partitions. The second architecture, `RECFilter`, uses rectangular filters, and therefore must assume a certain orientation in order to re-weight the particles according to their position, view class, and cardiac phase state variables. The orientation state variable must therefore appear in an earlier partition, giving an architecture with two partitions. The two filter architectures are defined in §6.4.2 and §6.4.3.

Both architectures use a common factorisation of the prediction potential, as outlined in §6.4.1, but place them in different orders within the filtering procedure.

6.4.1 Prediction Potential Factorisation

The factorisation of the prediction potential function used by both architectures breaks the potential function down as the product of terms corresponding to each

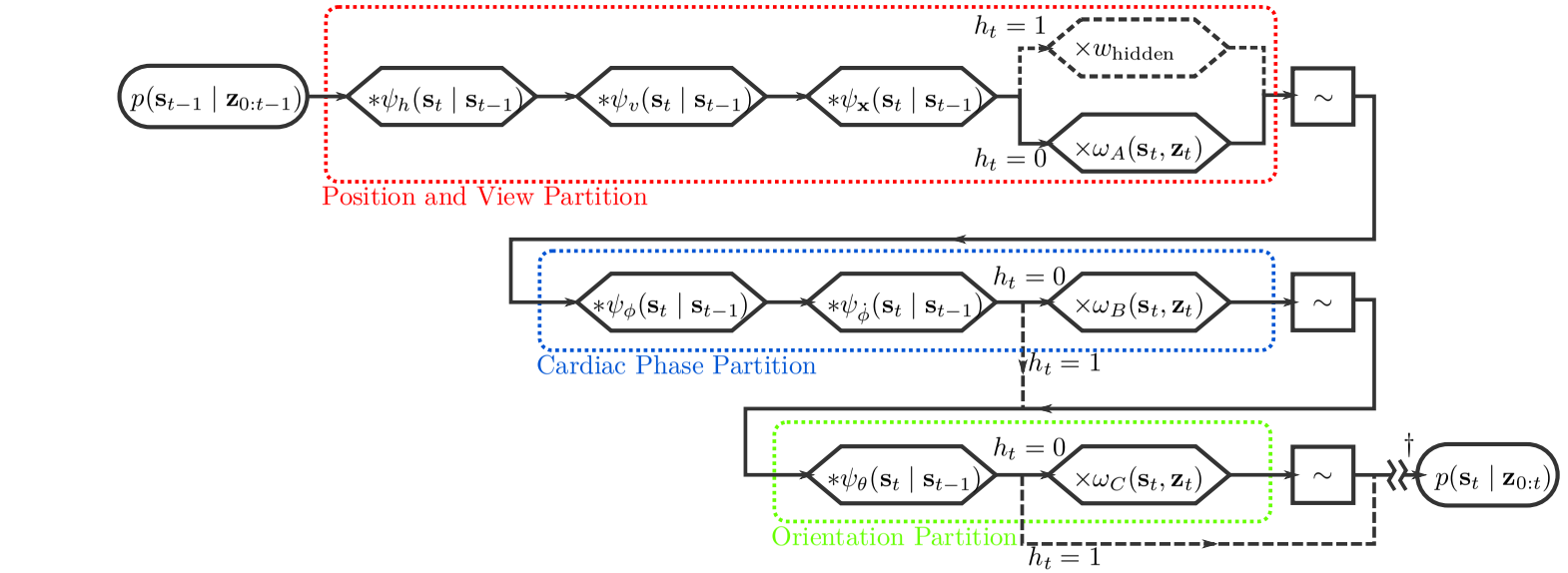
of the six state variables as follows (c.f. Equation 6.8):

$$\begin{aligned}\psi(\mathbf{s}_t \mid \mathbf{s}_{t-1}) = &\psi(h_t \mid h_{t-1}) \times \\ &\psi(v_t \mid v_{t-1}) \times \\ &\psi(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \theta_{t-1}, v_t, v_{t-1}) \times \\ &\psi(\theta_t \mid \theta_{t-1}, v_t, v_{t-1}) \times \\ &\psi(\phi_t \mid \phi_{t-1}, \dot{\phi}_{t-1}) \times \\ &\psi(\dot{\phi}_t \mid \dot{\phi}_{t-1}) \end{aligned} \tag{6.10}$$

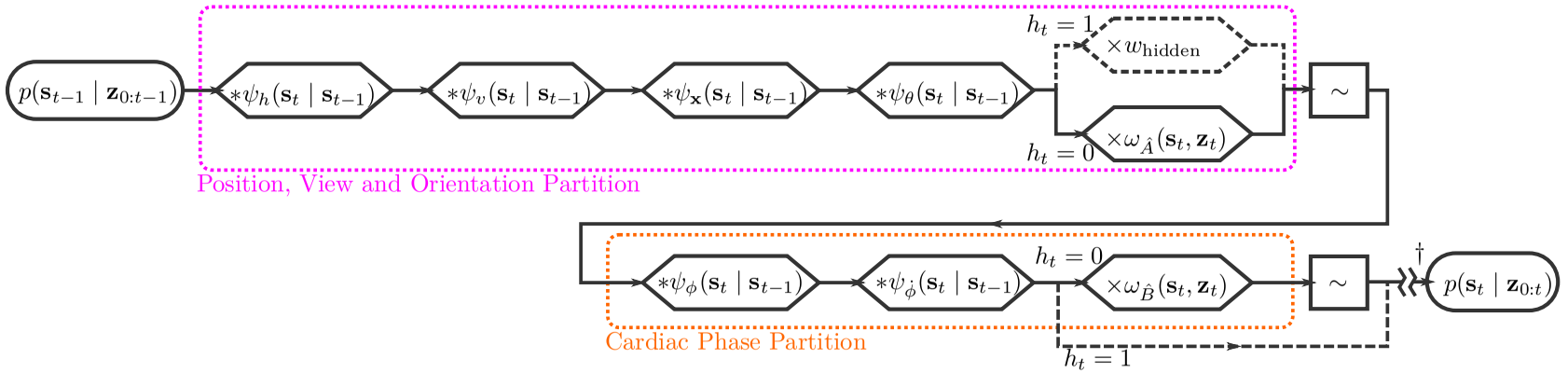
The details of each of these six prediction potentials are given in §6.5. Note that there is not a one-to-one correspondence between the terms in this factorisation and the partitions of the filtering architectures in the following sections. This reflects the fact that there are groups of variables in the state for which the prediction potential functions may be assumed to be independent, but that cannot be re-weighted independently.

For readability, a subscript-based shorthand notation for these terms will be used as follows:

$$\begin{aligned}\psi_h(\mathbf{s}_t \mid \mathbf{s}_{t-1}) &= \psi(h_t \mid h_{t-1}) \\ \psi_v(\mathbf{s}_t \mid \mathbf{s}_{t-1}) &= \psi(v_t \mid v_{t-1}) \\ \psi_{\mathbf{x}}(\mathbf{s}_t \mid \mathbf{s}_{t-1}) &= \psi(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \theta_{t-1}, v_t, v_{t-1}) \\ \psi_\theta(\mathbf{s}_t \mid \mathbf{s}_{t-1}) &= \psi(\theta_t \mid \theta_{t-1}, v_t, v_{t-1}) \\ \psi_\phi(\mathbf{s}_t \mid \mathbf{s}_{t-1}) &= \psi(\phi_t \mid \phi_{t-1}, \dot{\phi}_{t-1}) \\ \psi_{\dot{\phi}}(\mathbf{s}_t \mid \mathbf{s}_{t-1}) &= \psi(\dot{\phi}_t \mid \dot{\phi}_{t-1}) \end{aligned}$$



(a) Schematic of the `RIFFilter` architecture.



(b) Schematic of the `RECFilter` architecture.

Figure 6.6: Filter schematics for the `RIFFilter` and `RECFilter` architectures. The black dotted lines indicate routes taken only by 'hidden' particles (those with $h = 1$) and the coloured dotted boxes contain the operations within a single partition (one colour per partition). The break at the † symbol marks the location where extra stages are added for structure tracking in Chapter 8.

6.4.2 The `RIFFilter` Architecture

The schematic representation of the `RIFFilter` architecture is shown in Figure 6.6a. The filter uses three partitions. The first contains the visibility, h_t, heart position, $\mathbf{x}_t$, and view class v_t state variables and is re-weighted using the `RIFDetection` forest, which is invariant to orientation (due to its use of RIFs) and phase (by training). The second contains the cardiac phase, ϕ_t, and cardiac phase rate, $\dot{\phi}_t$, state variables, and is re-weighted using the `RIFPhase` forest models, which are invariant to orientation (due to their use of RIFs). The third and final partition contains the orientation, θ_t, state variable and is re-weighted using the `RIFOri` models.

6.4.3 The `RECFilter` Architecture

The schematic representation of the `RECFilter` architecture is shown in Figure 6.6b and uses two partitions. The first partition contains the visibility, h_t, heart position, $\mathbf{x}_t$, view class v_t, and orientation, θ_t, state variables and is re-weighted using the `RECDetection` forests at the relevant orientation, which are invariant to phase by training. The second contains the cardiac phase, ϕ_t, and cardiac phase rate, $\dot{\phi}_t$, state variables, and is re-weighted using the `RECPhase` forest models,

6.4.4 Filtering Using Hidden Particles

The standard particle filtering model assumes that the measurement information (in this case, the image) is always informative about the value of the state, as this is a natural assumption in most cases. However the fetal heart in ultrasound videos is not always visible in the image. Without the visibility state variable, h, the filter would always lock onto the particles that received the highest weight from the observation potentials, regardless of how high those weights were in absolute terms. This is a consequence of the normalisation of the weights that occurs in the SIR algorithm. Therefore when the heart disappeared from the image, the filter would lock onto some other part of the image and reflect any small variations in the observation potential function.

The purpose of the hidden state variable is both to prevent this undesirable behaviour and explicitly detect when the heart is no longer visible in the image. In both filter architectures, the re-weighting scheme for the hidden particles (those with $h_t = 1$) is different from that for visible particles (with $h_t = 0$), and uses a mechanism that is shown by the black dotted lines in the schematics in Figure 6.6. At the detection and view classification stage, the hidden particles are not re-weighted by the observation potential but are instead given a small, constant weight w_{hidden} that does not depend on the other state variables or the image information. This creates branches in the filtering schematic, similar to those used by MacCormick [176] for other purposes.

The rationale for this design is as follows. If many visible particles are assigned a high weight value ($\gg w_{\text{hidden}}$) by the observation potentials (indicating a high probability of a heart being present at their position), then the hidden particles will have small normalised weights and most will be removed at the following resampling step. However, if most visible particles are assigned a small weight ($\approx w_{\text{hidden}}$ or $< w_{\text{hidden}}$), then the *normalised* weights of the hidden particles will be large and many will survive the following resampling step, reflecting increased uncertainty about the presence of the heart.

The hidden particles then move through the rest of the prediction potentials in the same way as the visible particles. This means that the hidden particles still 'remember' the likely state that they will be in if they re-appear. This reflects the fact that if the heart disappears from the image due to obscuring artefacts or out-of-plane motion, it is most likely to be in a similar position and orientation when it re-appears but the uncertainty grows with the length of time for which the heart was not visible. Furthermore, the cardiac phase cycle should continue regardless of whether the heart is visible in the image or not. However, the hidden particles do not participate in any of the subsequent observation potentials or resampling steps, as it is not possible to assess the heart orientation or cardiac phase when the heart cannot be seen in the image.

6.5 Definition of Prediction Potential Terms

This section gives the details of the terms in the factorisation of the prediction potential in Equation 6.8. The same terms are used (it in a different order) in the both the filtering architectures presented in §6.4.

Recall that each term is a probabilistic model of the evolution of one state variable, and may depend on (i.e. be conditioned upon) other state variables in earlier partitions. The design of the terms aims to incorporate prior knowledge about the appearance of the heart in the video, including changes due to both fetal motion (including the fetal heartbeat) and probe motion during the scan. Another important consideration is that the terms must be easy to sample from in a computationally efficient manner.

The rest of this section considers each term in turn.

6.5.1 Visibility Prediction Potential, $\psi_h(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

This term models the probability of the heart moving between the hidden and visible states, represented by the Boolean state variable $h_t \in \{0, 1\}$ where 0 represents visible and 1 represents hidden. The term is independent of the other state variables:

$$\psi_h(\mathbf{s}_t \mid \mathbf{s}_{t-1}) = \psi(h_t \mid h_{t-1}) \tag{6.11}$$

The model captures the fact that the heart may become hidden due to relative motion between the probe and the fetus. This includes motion of the fetus and motion of the probe (voluntary and involuntary) in the sonographer's hand. The heart can disappear due to relative motion within the imaging plane (i.e. the heart disappears off the side of image) and/or perpendicular to the imaging plane. There are also situations where the heart is within the visible area but very strong artefacts (e.g. shadowing effects) obscure it.

To model these effects, a hidden particle becomes visible with a fixed probability $P_{\mathrm{h}\to\mathrm{v}} \in [0, 1]$, and a visible particle becomes hidden with a (generally different) fixed probability $P_{\mathrm{v}\to\mathrm{h}} \in [0, 1]$, i.e.

$$\psi(h_t \mid h_{t-1}) = \begin{cases} P_{\mathrm{h}\to\mathrm{v}}, & h_t = 0, h_{t-1} = 1 \\ 1 - P_{\mathrm{h}\to\mathrm{v}}, & h_t = 1, h_{t-1} = 1 \\ P_{\mathrm{v}\to\mathrm{h}}, & h_t = 1, h_{t-1} = 0 \\ 1 - P_{\mathrm{v}\to\mathrm{h}}, & h_t = 0, h_{t-1} = 0 \end{cases} \tag{6.12}$$

These parameters should not be chosen without a careful analysis of the statistics of the resulting Markov chain. Consider a single particle evolving according to this transition probability model in the absence of particle re-weighting and resampling. Let the probability of the particle being hidden at time t be $\lambda_t \in [0, 1]$. The probability that the particle is visible is accordingly $1 - \lambda_t$, and so we can write the distribution over the two states at time t as a vector

$$\boldsymbol{\lambda}_t = \begin{bmatrix} 1 - \lambda_t \\ \lambda_t \end{bmatrix} \tag{6.13}$$

Over time, this distribution will evolve as the particle randomly transitions between the two states. The evolution of this distribution is described by the *transition matrix* $\mathbf{M}$, as follows:

$$\begin{bmatrix} 1 - \lambda_t \\ \lambda_t \end{bmatrix} = \begin{bmatrix} 1 - P_{\mathrm{v}\to\mathrm{h}} & P_{\mathrm{h}\to\mathrm{v}} \\ P_{\mathrm{v}\to\mathrm{h}} & 1 - P_{\mathrm{h}\to\mathrm{v}} \end{bmatrix} \begin{bmatrix} 1 - \lambda_{t-1} \\ \lambda_{t-1} \end{bmatrix} \tag{6.14}$$

or, in matrix notation,

$$\boldsymbol{\lambda}_t = \mathbf{M}\boldsymbol{\lambda}_{t-1} \tag{6.15}$$

The stationary distribution of this Markov chain is the distribution to which the Markov chain converges as $t \to \infty$. The stationary distribution occurs at the eigenvector of $\mathbf{M}$ with eigenvalue 1 (there will always be one such eigenvalue by construction of $\mathbf{M}$), such that $\boldsymbol{\lambda}_t = \boldsymbol{\lambda}_{t-1}$ (see Chapter 17 of Murphy [172]). At this distribution, the probability that the particle is hidden is λ^*, where

$$\lambda^* = \frac{P_{\mathrm{v}\to\mathrm{h}}}{P_{\mathrm{v}\to\mathrm{h}} + P_{\mathrm{h}\to\mathrm{v}}} \tag{6.16}$$

and accordingly the probability of the particle being visible is $1 - \lambda^*$. Given sufficient time for convergence (assuming that none of the transition probabilities are zero),

the distribution over the two states for each particle will converge to this stationary distribution regardless of the initial state. Therefore, over time, the expected proportion of hidden particles in the particle set will also converge to λ^*.

Tuning this stationary distribution can be considered as a method for adjusting the sensitivity of the filter. If λ^* is large, the equilibrium state of the Markov chain contains mostly hidden particles, and so the weight of the visible particles (from the observation potentials) must be much larger than w_{hidden} to overcome this imbalance and bring a majority of visible particles at the resampling step. However, if λ^* is approximately 0.5 then the equilibrium state of the Markov chain contains a roughly equal number of hidden and visible particles, and consequently the weight of the visible particles need only be slightly larger than w_{hidden} to create a majority of visible particles.

If a value for λ^* is chosen, then the transition probabilities must be chosen such that

$$P_{\text{v}\to\text{h}} = P_{\text{h}\to\text{v}}\frac{\lambda^*}{1-\lambda^*} \tag{6.17}$$

which, along with the constraint that both are valid probabilities between 0 and 1, excludes certain values of each transition probability.

However, the stationary distribution is not the only important consideration when tuning these parameters because the transitory behaviour, or how the stationary distribution is reached, is also important. This is governed by the second eigenvalue of $\mathbf{M}$, which governs the decay of the component of the particle's initial state along the second eigenvector. The value of this eigenvalue is

$$\epsilon_2 = 1 - (P_{\text{v}\to\text{h}} + P_{\text{h}\to\text{v}}) \tag{6.18}$$

If a desired second eigenvalue ϵ_2 is specified along with a desired distribution, then solving Equations 6.17 and 6.18 gives values for the two transition probabilities required to give these desired properties:

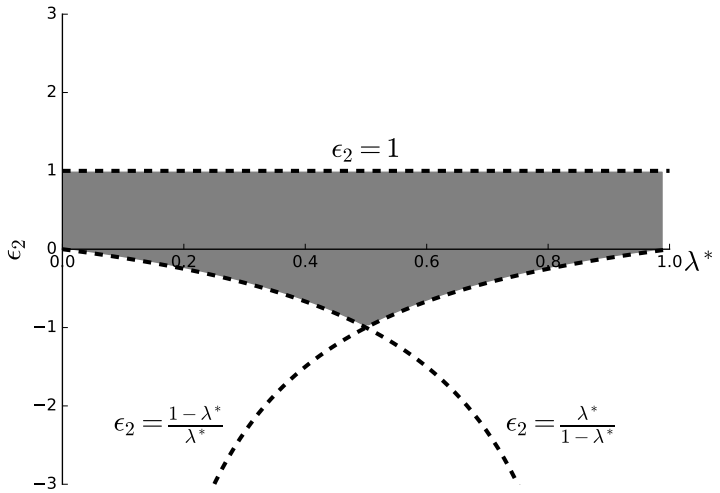


Figure 6.7: The valid region for ϵ_2 given λ^*. The dotted lines represent the three constraints and the grey shaded area represents the valid region.

$$P_{\mathrm{h}\to\mathrm{v}} = (1-\epsilon_2)(1-\lambda^*) \tag{6.19}$$

$$P_{\mathrm{v}\to\mathrm{h}} = (1-\epsilon_2)\lambda^* \tag{6.20}$$

Due to the constraints that transition probabilities $P_{\mathrm{h}\to\mathrm{v}}$ and $P_{\mathrm{v}\to\mathrm{h}}$ are both between 0 and 1, only certain values of ϵ_2 are achievable. This puts the following constraints on ϵ_2:

- $\epsilon_2 \leq 1$. This follows straightforwardly from Equation 6.18 and $P_{\mathrm{h}\to\mathrm{v}} \geq 0$ and $P_{\mathrm{v}\to\mathrm{h}} \geq 0$.
- $\epsilon_2 \geq \frac{\lambda^*-1}{\lambda^*}$ and $\epsilon_2 \geq \frac{\lambda^*}{\lambda^*-1}$. These can be derived from Equations 6.19 and 6.20 and $P_{\mathrm{h}\to\mathrm{v}} \leq 1$ and $P_{\mathrm{v}\to\mathrm{h}} \leq 1$. Note that the first of these two constraints is active when $\lambda^* \geq 0.5$ and the second is active when $\lambda^* \leq 0.5$ (and both when $\lambda^* = 0.5$). In all cases this means that $\epsilon_2 \geq -1$.

These constraints lead to a permissible region for ϵ_2 as shown in Figure 6.7.

The effect of altering ϵ_2 is described in Table 6.1 and simulated in Figure 6.8. Desirable behaviour for the purposes of the filter is that the distribution over the

Region	Behaviour	Description
$\epsilon_2 = 1$	Marginally Stable	The particle's state is fixed over time.
$0 < \epsilon_2 < 1$	Overdamped	The distribution over the two states decays to the stationary distribution without oscillation. The lower ϵ_2 is, the faster the decay.
$\epsilon_2 = 0$	Critically Damped	The distribution over the two states moves straight to the stationary distribution in a single time-step. This occurs when the probability of the heart being hidden in a given time-step does not depend on previous time-steps.
$-1 < \epsilon_2 < 0$	Underdamped	The distribution over the two states oscillates around the stationary distribution with decaying amplitude. The smaller $\lvert\epsilon_2\rvert$, the faster the decay.
$\epsilon_2 = -1$	Undamped	This is only possible when $\lambda^* = 0.5$ and implies that the particle will continuously change back and forth between the two states on each time-step.

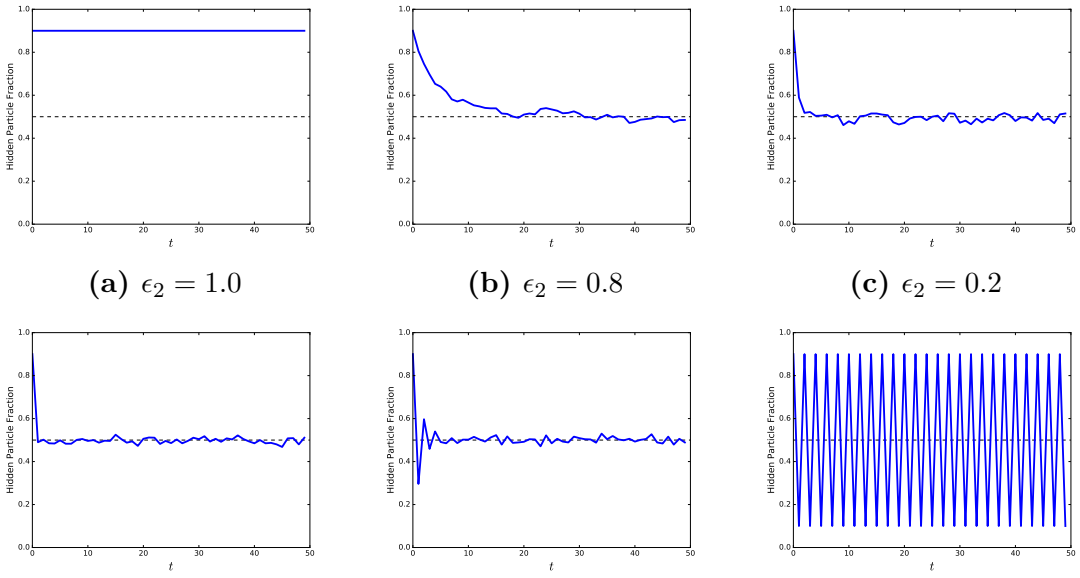
Table 6.1: Behaviour of the distribution over the two states for a single particle over time for different values of ϵ_2. This only considers the behaviour in the absence of the re-weighting and resampling steps.

two states should converge to the stationary distribution relatively slowly in order to model the fact the heart changes from hidden to visible infrequently. This suggests that the value of ϵ_2 should be set to be a positive number below but fairly close to 1, however there is also an interaction with the λ^* parameter. Rather than choose ϵ_2 directly, it is more intuitive to instead choose a time constant parameter τ, given by

$$\tau = \frac{1}{1 - |\epsilon_2|} \tag{6.21}$$

which determines the number of frames taken to reach the stationary distribution (the distribution gets closer to the equilibrium fraction by a factor of e^{-1} every τ frames).

The effect of varying these parameters on the detection performance in investigated experimentally in Chapter 7.



(a) $\epsilon_2 = 1.0$ **(b)** $\epsilon_2 = 0.8$ **(c)** $\epsilon_2 = 0.2$

(d) $\epsilon_2 = 0.0$ **(e)** $\epsilon_2 = -0.5$ **(f)** $\epsilon_2 = -1.0$

Figure 6.8: Simulations of the evolution of the fraction of hidden particles in a particle set in the absence of re-weighting and re-sampling steps. Different values of ϵ_2 are shown, exhibiting the behaviour described in Table 6.1. In all cases $\lambda^* = 0.5$ (shown as a horizontal dotted line) and the initial distribution contained 1000 particles with 900 hidden particles.

6.5.2 View Prediction Potential, $\psi_v(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

This term models the view transitions made between the three view categories represented by the discrete state variable $v_t \in \{4\text{C,LVOT,3V}\}$,

$$\psi_v(\mathbf{s}_t \mid \mathbf{s}_{t-1}) = \psi(v_t \mid v_{t-1}) \tag{6.22}$$

A view transition occurs due to (usually deliberate) relative motion of the probe and the fetus in the direction perpendicular to the imaging plane. The probability of a transition between the different viewing planes is implemented simply as a discrete distribution with a constant probability of moving to each new state:

$$\psi(v_t \mid v_{t-1}) = \begin{cases} p_{\text{same}}, & v_t = v_{t-1} \\ p_{\text{change}}, & v_t \neq v_{t-1} \end{cases} \tag{6.23}$$

Typically a sonographer will move relatively slowly between the different views and therefore $p_{\text{same}} \gg p_{\text{change}}$. However, it is often helpful to slightly overestimate the probability of transition to allow the filter to recover from mistakes. The

three viewing planes have a natural order to them (4C, LVOT and 3V moving in a cephalad direction, i.e. towards the fetal head) that could be modelled at this stage by, for example, making it more likely for a 4C view to transition to an LVOT view than to a 3V view. However observations of the dataset have suggested that in practice abrupt probe motions do cause transitions through multiple views like this relatively frequently.

Note also that this assumes that view transitions happen instantaneously between two consecutive frames, whereas in practice they occur over a number of frames with a number of ambiguous frames occurring in between. However, modelling this transitional period would dramatically increase the complexity of the model with little gain.

6.5.3 Position Prediction Potential, $\psi_{\mathbf{x}}(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

This term models changes in the position of the heart centre in the image, $\mathbf{x}_t \in \mathbb{R}^2$, and depends upon the heart view and orientation:

$$\psi_{\mathbf{x}}(\mathbf{s}_t \mid \mathbf{s}_{t-1}) = \psi(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \theta_{t-1}, v_t, v_{t-1}) \tag{6.24}$$

When a frame contains the same view of the heart as the previous frame ($v_t = v_{t-1}$), then a change in position can occur due to probe motion and/or fetal motion within the imaging plane. When a view transition occurs ($v_t \neq v_{t-1}$), the change of position is composed of two parts. The first is in-plane motion as can occur between any two consecutive frames. The second is due to the fact that the centre position is defined differently in each view (see Figure 1.4). This *view offset* is uncertain due to anatomical variation between subjects and small differences in the location of the imaging plane within the anatomy.

A 2D Gaussian distribution is used to model the change in position between each pair of different views (v_t and v_{t-1}). The distributions are learnt at training time relative to a heart at orientation zero and with unit radius, giving *relative* offset distributions with means $\hat{\boldsymbol{\mu}}_{v_1 \to v_2} \in \mathbb{R}^2$ and covariances $\hat{\boldsymbol{\Sigma}}_{v_1 \to v_2} \in \mathbb{R}^{2\times 2}$, where the '$\hat{\cdot}$' diacritic is used to distinguish the *relative* distribution parameters. At test

time, these are then scaled by the radius R_{test} and rotated by the orientation θ_{t-1} to give the *absolute* mean and covariance of the offset.

Specifically:

$$\psi(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \theta_{t-1}, v_t, v_{t-1}) = \mathcal{N}_2(\mathbf{x}_t \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \tag{6.25}$$

where the (absolute) mean and covariance are given by:

$$\boldsymbol{\mu}_t = \mathbf{x}_{t-1} + R_{\text{test}}\mathbf{R}_{[\theta_{t-1}]}\hat{\boldsymbol{\mu}}_{v_{t-1}\to v_t} \tag{6.26}$$

$$\Sigma_t = r\mathbf{R}_{[\theta_{t-1}]}\hat{\boldsymbol{\Sigma}}_{v_{t-1}\to v_t}\mathbf{R}^T_{[\theta_{t-1}]} \tag{6.27}$$

Here, $\mathcal{N}_2(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the PDF of a 2D Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, and $\mathbf{R}_{[\theta]}$ is the 2×2 rotation matrix representing a rotation through angle θ. Note that the mean of the relative offset distribution, $\hat{\boldsymbol{\mu}}_{v_1\to v_1}$, is constrained to be zero when the view does not change. However the covariance, $\hat{\boldsymbol{\Sigma}}_{v_1\to v_1}$, is non-zero to represent in-plane motion. In practice, sampling from the 2D Gaussian is achieved using the precomputed Cholesky decomposition of the covariance matrix.

In the case where the updated position moves the heart to within the detection radius of the edge of the ultrasound fan area (i.e. off the edge of the permissible area of the image), the updated value is ignored and the previous position is maintained.

6.5.4 Orientation Prediction Potential, $\psi_\theta(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

This term models changes in the orientation of the heart, $\theta_t \in [0, 2\pi)$, and depends on the view state variable:

$$\psi_\theta(\mathbf{s}_t \mid \mathbf{s}_{t-1}) = \psi(\theta_t \mid \theta_{t-1}, v_t, v_{t-1}) \tag{6.28}$$

When no view transition has occurred between the time t and $t-1$, changes in orientation are due to relative angular motion between the fetus and the probe within the imaging plane. When there is a view transition, there is an *orientation offset* due to the different definitions of the orientation in each view (see Figure 1.4). The orientation offset accompanying each view transition is modelled by a *wrapped*

normal distribution [166]. The wrapped normal distribution with mean μ and variance σ is a distribution over the range $[0, 2\pi)$ formed by wrapping the PDF of the univariate Gaussian distribution with mean μ and variance σ around the range $[0, 2\pi)$. Consequently, unlike the similar von Mises distribution, its PDF has an inconvenient form (expressed as an infinite sum), but is straightforward to sample from. Sampling from a wrapped normal distribution involves sampling from the corresponding Gaussian distribution and then wrapping the result to lie in the range $[0, 2\pi)$.

Each view transition uses its own wrapped normal distribution with mean $\hat{\xi}_{v_1 \to v_2} \in [0, 2\pi)$ and variance $\tau_{v_1 \to v_2} \in \mathbb{R}_0^+$ for the orientation offset. The distribution parameters are learnt at training time using standard fitting routines for circular data.

$$\psi(\theta_t \mid \theta_{t-1}, v_t, v_{t-1}) = \mathcal{W}(\theta_t \mid \xi_t, \tau_{v_{t-1} \to v_t}) \tag{6.29}$$

where

$$\xi_t = \theta_{t-1} + \hat{\xi}_{v_{t-1} \to v_t} \tag{6.30}$$

and $\mathcal{W}(\cdot \mid \xi, \tau)$ is the PDF of the wrapped normal distribution. Again zero mean ($\hat{\xi}_{v_1 \to v_1} = 0$) but non-zero variance is assumed when no view transition has occurred. Note that the variance τ does not depend on any other parameter, and therefore does not have a relative and absolute form.

6.5.5 Cardiac Phase Prediction Potential, $\psi_\phi(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

This term models changes in the cardiac phase variable, $\phi \in [0, 2\pi)$. Since a second order model is used for the cardiac phase, in this case the model is a deterministic one given the current value of the phase rate variable, $\dot{\phi}$. Assuming that $\dot{\phi}$ is expressed in $\mathrm{rad\,s^{-1}}$, the update equation is

$$\phi_t = \phi_{t-1} + \frac{\dot{\phi}_{t-1}}{\Delta t} \tag{6.31}$$

where Δt is the (constant) time elapsed between video frames in seconds. The purpose of dividing by Δt here is to ensure that the state evolution model is not sensitive to the frame rate of the video being analysed.

6.5.6 Cardiac Phase Rate Prediction Potential, $\psi_{\dot{\phi}}(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

This term models changes in the cardiac phase rate variable $\dot{\phi}$, and is assumed to be independent of the other state variables.

$$\psi_{\dot{\phi}}(\mathbf{s}_t \mid \mathbf{s}_{t-1}) = \psi(\dot{\phi}_t \mid \dot{\phi}_{t-1}) \tag{6.32}$$

In the absence of cardiac defects, the cardiac phase rate (i.e. heart rate) should remain relatively constant over the course of the video. However some variation must be permitted to allow the filter to track small changes, lock on to the correct phase rate initially and recover from mistakes. Therefore small variations are modelled using a Gaussian offset, with constant variance η:

$$\psi(\dot{\phi}_t \mid \dot{\phi}_{t-1}) = \mathcal{N}_1(\dot{\phi}_t \mid \dot{\phi}_{t-1}, \eta) \tag{6.33}$$

In order to prevent unreasonable values, and avoid time aliasing effects, hard limits of $\dot{\phi}_{\min}$ and $\dot{\phi}_{\max}$ are placed on the value of the cardiac phase rate. If an update would take the value outside of the permissible region, the updated value is ignored and the previous value in maintained.

6.5.7 Particle Initialisation

The particle set must be initialised to represent the initial distribution at the start of the video. This is performed by sampling a value for each state variable of each particle independently from each other particle and state variable according to the distributions outlined in Table 6.2.

State Variable	Symbol	Initial Distribution
Visibility	h	A Bernoulli distribution with $p(h = 1) = \lambda^*$, the stationary hidden fraction.
View	v	A discrete uniform distribution over the three views.
Location	$\mathbf{x}$	A 2D continuous uniform distribution over the valid region of the image (more than the heart radius R_{test} from the edge of the ultrasound fan area).
Orientation	θ	A continuous uniform distribution over the interval $[0, 2\pi)$.
Cardiac Phase	ϕ	A continuous uniform distribution over the interval $[0, 2\pi)$.
Cardiac Phase Rate	$\dot{\phi}$	A gamma distribution with shape parameter $\alpha_0 \in \mathbb{R}^+$ and rate parameter $\beta_0 \in \mathbb{R}^+$, which are fitted at training time on the annotated training set. Values outside the range $[\dot{\phi}_{\min}, \dot{\phi}_{\max}]$ are drawn again.

Table 6.2: The initial distributions used for creating the initial particle set.

6.6 Definition of Observation Potential Terms

Table 6.3 contains the definitions of the five observation functions used in the filters (see Figure 6.6). Note that, unlike the prediction potential terms, different observation potentials are used in each of the two filtering architectures.

6.7 Extracting State Estimates from a Particle Set

Recall that the purpose of the particle filter is to maintain a probabilistic estimate of the state at each time-step by updating the filtering distribution, which is represented by a weighted a particle set. Whilst this approach has a number of advantages, especially with regards to the robustness of the algorithm, for many purposes it is necessary to extract a single point estimate of the state value at each time-step, rather than work with the full filtering distribution.

One way of doing this is to take the mean of all the particles in the set as the state estimate. This is very straightforward to do and generally results in a

Function	Definition
$\omega_A(\mathbf{s}_t, \mathbf{z}_t)$	The probability of the view v_t at image location $\mathbf{x}_t$ as calculated by the `RIFDetection` forest.
$\omega_B(\mathbf{s}_t, \mathbf{z}_t)$	The probability of the cardiac phase value ϕ_t at the location $\mathbf{x}_t$ as calculated by the `RIFPhase` forest model for view v_t.
$\omega_C(\mathbf{s}_t, \mathbf{z}_t)$	The probability of the orientation θ_t at image location $\mathbf{x}_t$ as calculated by the `RIFOri` models attached to the leaf nodes of the `RIFPhase` forest model that are reached by passing the particle into the `RIFPhase` forest model as above.
$\omega_{\hat{A}}(\mathbf{s}_t, \mathbf{z}_t)$	The probability of the view v_t at image location $\mathbf{x}_t$ as calculated by the two `RECDetection` forests models with training orientations closest to θ_t (one either side of θ_t accounting for the wrapping of the orientation). To get a single value, the weighted mean of these two values is used, where the weights reflect the distances of the training angle from θ_t.
$\omega_{\hat{B}}(\mathbf{s}_t, \mathbf{z}_t)$	The probability of the cardiac phase value ϕ_t as at the location $\mathbf{x}_t$ as calculated by the two `RECPhase` forests models with training orientations closest to θ_t and averaged as above.

Table 6.3: Definitions of the observation potential functions used by the two architectures. Details of the models are referred to are found in §5.3.1 and §5.5.1.

state estimate that varies smoothly over time. However, this does not give very meaningful results when the filtering distribution is multi-modal, as the mean might lie in an area of low probability (i.e. low particle density).

A second method is to estimate the mode of the filtering distribution by selecting the particle with the highest normalised weight before the re-sampling step. This ensures that the point estimate lies in an area of high probability density in the filtering distribution. However it is very sensitive to small variations in the observation potentials and therefore tends to vary significantly and erratically between time-steps, leading to a point estimate that behaves in an unrealistic manner over time.

To balance between this two extremes, a third option is to use the *mean-shift* algorithm [178] to find a mode of the filtering distribution. This ensures that the estimate lies in a high density area of the filtering distribution, but generally results in more smoothly varying point estimates than choosing the particle with the highest weight.

However, the mean-shift algorithm cannot be straightforwardly applied to a particle set in the heart state space used here because it contains a mixture of discrete, continuous and circular variables. This section describes a method to apply the mean shift algorithm to calculate point estimates from partitioned particle filters.

The central idea is to apply the mean-shift algorithm to each state variable in turn, in the order that they are updated in the partitioning scheme. However, only the particles that fall within the mean-shift kernel of all previous partitions are considered within a certain partition.

For the discrete state variables (h and v), the mean-shift algorithm simply involves finding the mode of the discrete distribution. Only the particles that have this the modal value of the state variable are considered in the mean-shift stage for subsequent variables. For example, if the modal value of h_t is 0 (i.e. visible) then only the visible particles are considered in the mean-shift stage for v_t.

For the real-valued state variables ($\mathbf{x}$ and $\dot{\phi}$), the mean-shift algorithm starts at the mean value of the input particle set. Mean-shift then takes place with a (flat) kernel of a specified size ($K_{\mathbf{x}}$ or $K_{\dot{\phi}}$ for the corresponding state variables) and continues until the updates are less than a specified tolerance ($\epsilon_{\mathbf{x}}$ and $\epsilon_{\dot{\phi}}$). Only those particles that lie within the kernel are passed down to the mean-shift for the next state variable.

For the circular state variables (θ and ϕ), mean-shift starts at the circular mean (Equation 4.8) of the input particle set and uses a kernel based on the circular distance (Equation 4.9) with a fixed angular width (K_θ or K_ϕ for the corresponding state variables). It continues until the updates fall below a fixed tolerance (ϵ_θ and ϵ_ϕ), and only particles whose values fall within this kernel are passed down to the mean-shift algorithm for the next state variable.

7

Experimental Validation of Filtering Architectures

Contents

This chapter experimentally evaluates the filtering framework described in Chapter 6. Related results were presented in an article in *Medical Image Analysis* [163], but those experiments did not use partitioning of the particle filter framework and were conducted with an older, less computationally efficient version of the implementation. The results here also expand upon those in that article by considering the effect of further parameters (such as the composition of the random forest models), and performing a comparison of RIFs with more standard rectangular features.

Parameter	Description
$\hat{\boldsymbol{\mu}}_{v_1 \to v_2}$, $\hat{\boldsymbol{\Sigma}}_{v_1 \to v_2}$	Mean and covariance of the relative distribution for the position prediction potential when a view transition between two *different* views v_1 and v_2 occurs. There are six such distributions, one for each combination of different views.
$\hat{\xi}_{v_1 \to v_2}$, $\tau_{v_1 \to v_2}$	Circular mean and circular variance of the orientation prediction when a view transition between two *different* views v_1 and v_2 occurs. There are six such distributions, one for each combination of different views.
α_0, β_0	Shape and rate parameters of the gamma distribution for the initialisation of cardiac phase rate particles.

Table 7.1: List of filter prediction parameters fitted to training data.

7.1 Fitting Prediction Potential Models

There are a number of parameters of the prediction potentials (§6.5) that must be fitted to the training data, which in this case are the ground truth annotations. These parameters are listed in Table 7.1.

The α_0, β_0 parameters are straightforward to fit to the set of annotations. The cardiac phase rate is calculated at each frame in the dataset that contains a heart by referring to the cardiac phase value in that frame and the previous frame. Then, the α_0, β_0 can be fitted to these values using a standard maximum likelihood approach for a gamma distribution.

The remaining parameters from Table 7.1 ($\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\Sigma}}$, $\hat{\xi}$, and τ) are more complicated to fit because they relate to transitions between the different views. The first stage is therefore to find suitable transitions – i.e. sequences of video where the view label changes from one of the heart views to another heart view in consecutive frames. The accompanying change in position and orientation then forms one data point for the transition parameters in question. Given these data points, the parameters can be fitted using standard maximum likelihood parameters.

A number of other parameters were fixed in value for all experiments using values that were found to give reasonably good results. These are listed in Table 7.2.

Parameter	Value	Description
p_{same}	0.90	Discrete probability of a particle keeping its view label between frames.
p_{change}	0.05	Discrete probability of a particle moving to each of the other views between frames.
$\mathbf{\Sigma}_{v_1 \to v_1}$	$1.0\mathbf{I}$ pixels^2	Covariance of the position prediction potential when the view does not change between two consecutive frames.
$\tau_{v_1 \to v_2}$	$0.0025\,\text{rad}^2$	Circular variance of the orientation prediction potential when the view does not change between two consecutive frames.
η	$0.04\,\text{rad}^2\text{s}^{-2}$	Variance of the prediction potential for the cardiac phase rate variable.
$\dot{\phi}_{\min}$, $\dot{\phi}_{\max}$	100, 200 bpm	The minimum and maximum permissible values for the cardiac phase rate variable $\dot{\phi}$.

Table 7.2: List of filter prediction parameters taking fixed values.

Parameter	Value	Description
$K_{\mathbf{x}}$	5.0 pixels	Kernel size for the position mean shift.
K_θ	0.25 rad	Kernel size for the orientation mean shift.
K_ϕ	0.25 rad	Kernel size for the cardiac phase mean shift.
$\epsilon_{\mathbf{x}}$	1.0 pixels	Tolerance for the position mean shift.
ϵ_θ	0.05 rad	Tolerance for the orientation mean shift.
ϵ_ϕ	0.05 rad	Tolerance for the cardiac phase mean shift.

Table 7.3: Parameters of the mean-shift algorithm used in experiments.

The effect of varying the parameters of the visibility prediction potential (the equilibrium fraction of hidden particles λ^*, time constant τ and hidden weight w_{hidden}) is investigated in §7.3.3.

7.2 Mean Shift Parameters

The parameters of the mean-shift algorithm used to extract point estimates for the state values from the particle sets were not found to have a significant impact on the results. Reasonable values, shown in Table 7.3, were chosen empirically and used for all experiments.

Methodology	Average Calculation Time (ms)
`spatial`	1256.7
`spatial + memo`	37.6
`freq`	18.6
`auto`	18.6
`auto + memo`	18.5

Table 7.4: Table of values in Figure 7.1.

7.3 Results

Within the particle filtering framework, the estimates of the different variables (location, view, orientation, phase, visibility) become interdependent because they are all based on a single set of particles that is affected by the reweighting stages of all the partitions. It is therefore not possible to consider the different variables in complete isolation. However in the following section, the variables will be discussed separately as far as is possible.

7.3.1 RIF Calculation Methodologies

The relative speeds of the different calculation methodologies was investigated previously in §5.6.1. However, the speed of the calculations is likely to be highly dependent upon the way in which the forest models are queried. The experiments in Chapter 5 considered every valid image location in each frame as a possible location for the heart, and therefore a very large number of queries of the random forest model are made for each frame. By contrast, in the particle filtering framework, the observations potentials need only be evaluated at the positions of the relatively small number of particles. Furthermore, since the particles cluster around areas of high probability, and the observation potentials necessarily round the image location to the nearest pixel, it is likely that many of these image locations will be duplicates of each other.

Consequently, the random forests models are used in a rather different way when a particle filter is used compared to the case investigated in 5.6.1. This motivates an investigation of the speeds of the RIF calculation methodologies when particle filtering is used.

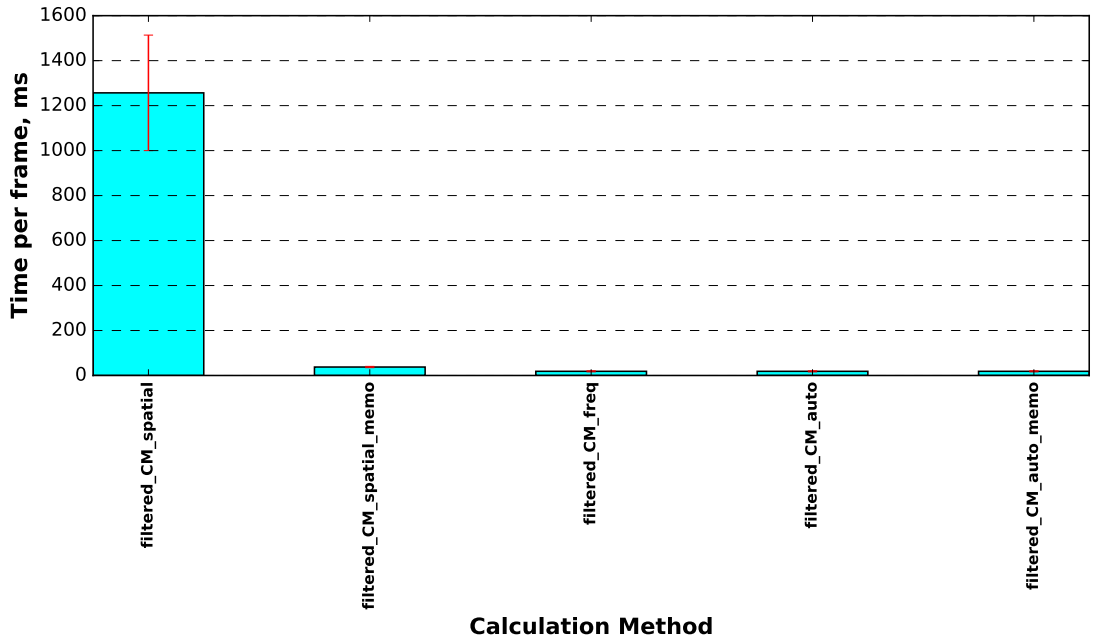


Figure 7.1: Average per-frame calculation times for different calculation methodologies with particle filtering. The models used were a `RIFDetection` forest with $N_{\text{trees}} = 16$ trees and $D_{\max} = 10$ and a `RIFPhase` forest with $N_{\text{trees}} = 32$ trees and $D_{\max} = 8$, both using `grad322motion322_extra`, and 1000 particles.

Figure 7.1 and Table 7.4 show the results of an experiment conducted to assess the different calculation methods when using a particle filter. Note when comparing to the results in §5.6.1 that this experiment also estimated the orientation and phase variables whereas the previous experiment just estimated location and view.

The results show that despite using a smaller number of points, the frequency domain calculations still provide a very substantial speed up compared to spatial domain calculations. However, memo-isation of features with the spatial results in an even more dramatic speed up than in the previous experiment. This is because once calculated, features are more likely to be re-used later by the `RIFPhase` forest or `RIFOri` models.

The average calculation time per frame for the best method is 18.5 ms, which equates to around 54 frames per second. Whilst the exact value will depend on other parameters, such as number of trees and particles used, this is a high frame rate that is well-suited to real-time performance.

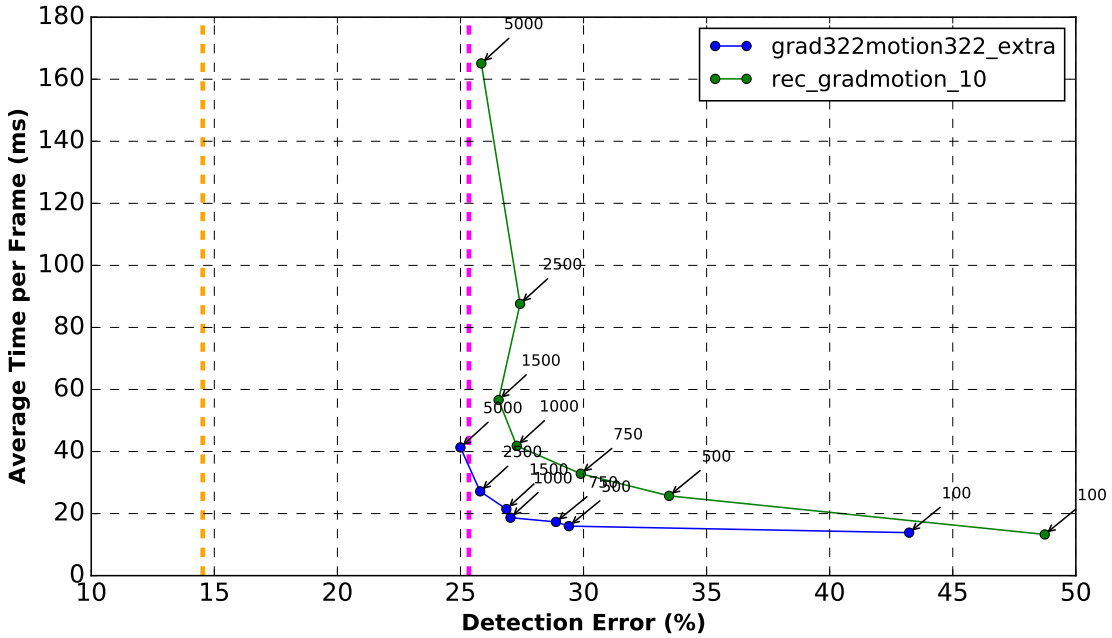


Figure 7.2: Detection error (§5.1) and calculation time with different numbers of particles in the `RIFFilter` and `RECFilter` architectures. The feature sets were `grad322motion322_extra` using the `freq` calculation method for the `RIFFilter` model, and `rec_gradmotion_10` for the `RECFilter` model. The `RIFDetection`/`RECDetection` models used $N_{\text{trees}} = 32$ and $D_{\max} = 10$, the `RIFPhase`/`RECPhase` models used $N_{\text{trees}} = 32$ and $D_{\max} = 8$. The filter parameters were $\lambda^* = 0.4$, $\tau = 2$ frames and $w_{\text{hidden}} = 0.3$ for the `RIFFilter` and $\lambda^* = 0.3$, $\tau = 2$ frames and $w_{\text{hidden}} = 0.2$ for the `RECFilter`. Note that the calculation time given is for the entire filtering framework, also involving orientation and cardiac phase estimation. The orange line shows the estimate of intra-observer variation, the magenta line shows the estimate of inter-observer variation.

7.3.2 Detection Performance

Figure 7.2 shows the detection performance as the number of particles in the particle set varies. As the number of particles increases, the detection accuracy generally improves as the particle set better approximates the true posterior, however this benefit saturates at high numbers of particles. However, there is a clear trade off with calculation time, as a larger number of particles means more evaluations of the observation potentials and a greater cost incurred in applying the prediction potential and resampling operations. On balance, a particle set of approximately 1000 particles gives good detection accuracy at a reasonably high speed.

Note that the filtering parameters used (particularly the hidden weight w_{hidden}) have a significant effect on the detection error, and different filtering parameters are

better suited to different models. Consequently, different filtering parameters (λ^*, τ frames and w_{hidden}) have been used in Figure 7.2 in order to give a similar trade off between true positive and false positive detection rates. This is discussed in more detail in §7.3.3.

One significant difference between the `RIFFilter` and `RECFilter` architectures is that the `RIFFilter` architecture is not slowed down so significantly by adding extra particles. A possible explanation for this is that the frequency-domain calculations for the raw RIFs results in values for the entire image at once. The speed of computationally intensive step is therefore not dependent on the size of the particle set. The coupling calculations are however performed on a per-particle basis. All feature calculations for the `RECFilter` are by contrast on a per-particle basis, explaining why the `RECFilter` line in Figure 7.2 rises more steeply as the number of particles increases. Furthermore, the evaluation of each particle's observation potential requires querying four different forest models corresponding to the `RECDetection` and `RECPhase` models for the two orientation bins on either side of the particle's orientation, whereas the `RIFFilter` only requires querying two forests, the `RIFDetection` and `RIFPhase` models.

However given the relative simplicity of the calculating each feature value in the `RECFilter` compared to the `RIFFilter`, it is surprising that the former appears to be significantly slower. Whilst care has been taken to create an efficient implementation of both algorithms, it is possible that the performance of the `RECFilter` could be improved with respect to the `RIFFilter` by careful code profiling.

Estimated values for the inter- and intra-observer variation are also shown in Figure 7.2. These are calculated by evaluating the additional sets of annotations (§1.5) against the ground truth annotation set in exactly the same way as the automatic estimates. It is clear that there is significant inter- and intra-observer variation between the annotation sets, which reflects the degree of ambiguity in the annotation task and provide a realistic target for the algorithm. In terms of detection performance, the automatic algorithm performs on a par with a second annotator, but falls somewhat short of the first human annotator in Figure 7.2.

However it should be noted that the filter parameters used in this experiment were chosen to emphasise a reasonable false positive rate over a high true positive rate, and much higher true positive rates can achieved at the cost of an increased false positive rate if different parameters are chosen (see §7.3.3).

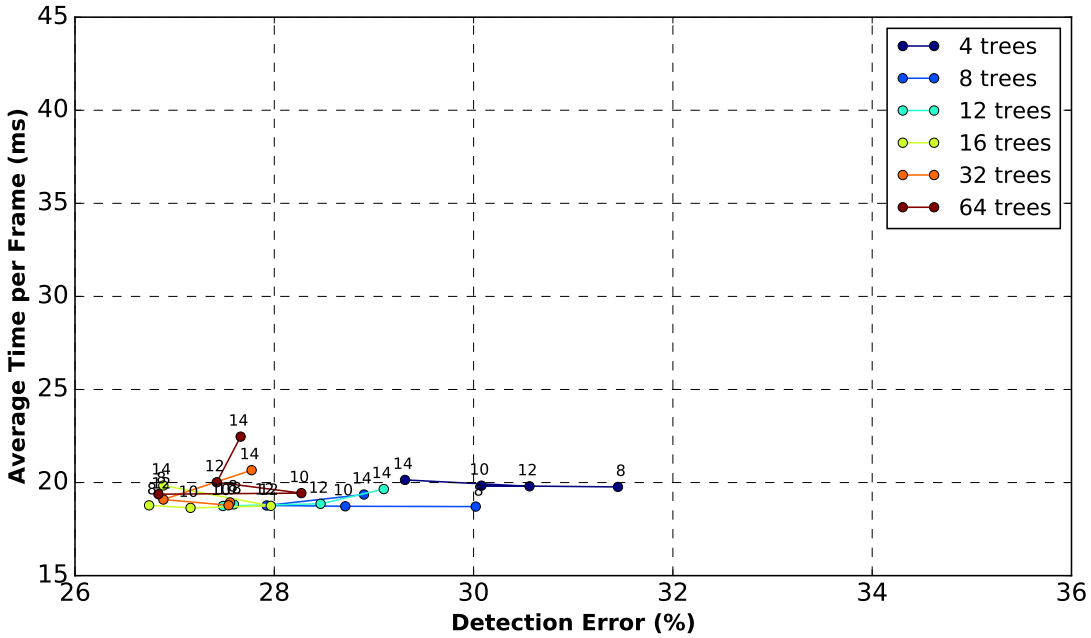
Figure 7.3 shows the performance as the number of trees and levels in the detection forest varies. In these figures (unlike Figure 7.2), the filtering parameters (λ^*, τ, and w_{hidden}) were chosen separately for the `RIFFilter` and the `RECFilter` to give approximately the same trade off between true positive rate and false positive rate using the value in §7.3.3. Comparing these plots to those in Figure 5.5 shows that the use of the particle filtering is able to significantly boost the detection performance.

It is also clear that the effect of the number of trees and maximum number of levels on accuracy is much less significant than in the unfiltered case. This suggests that, by combining information from multiple frames with a strong prediction model, the filtering architectures are able to make effective use of only reasonably accurate observation potentials, and so can still achieve good results with only a few trees.

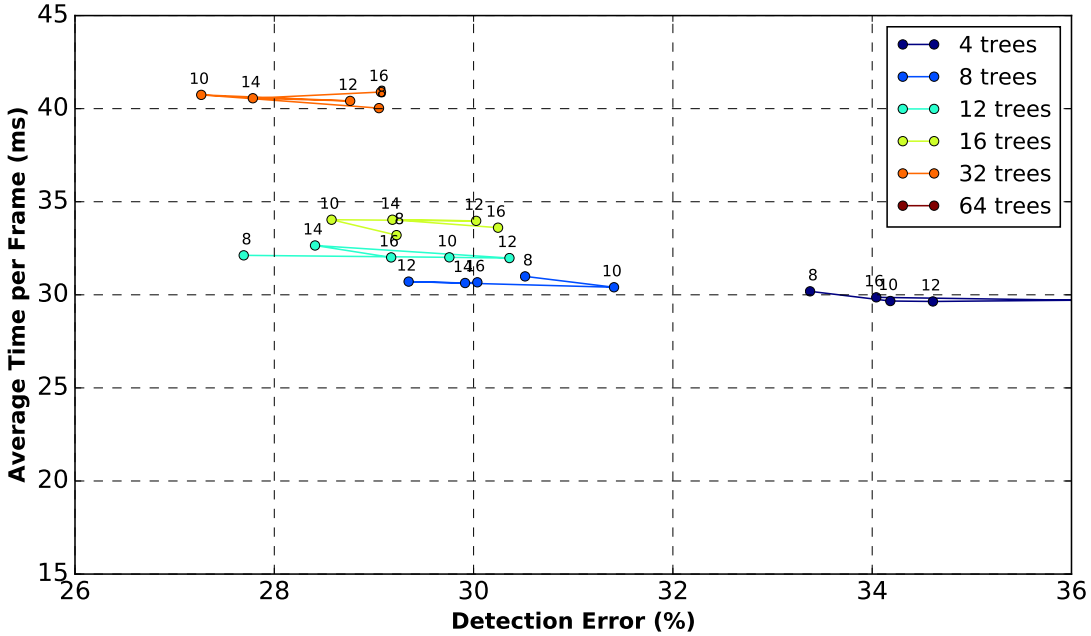
Increasing the number of trees and/or levels in the forest models increases the total number of feature evaluations needed. In Figure 7.3 the fact that increasing the number of feature evaluations slows down the `RECFilter` model more than the `RIFFilter` is observed, as it was in Figure 7.2.

The nature of the particle filtering algorithm is such that the estimates of the different state variables become intricately interconnected. Consequently, the performance of the phase regression forests (the `RIFPhase` and `RECPhase` models) can also have a significant effect on the detection performance. This effect is demonstrated in Figure 7.4, which shows the detection error and calculation time as the composition of the phase regression forests is varied, and the composition of the classification/detection forests is held constant.

The results show that the effect of the phase regression forests composition is approximately as significant that of the classification/detection forest composition. This demonstrates how linking the variables together via the particle filter is

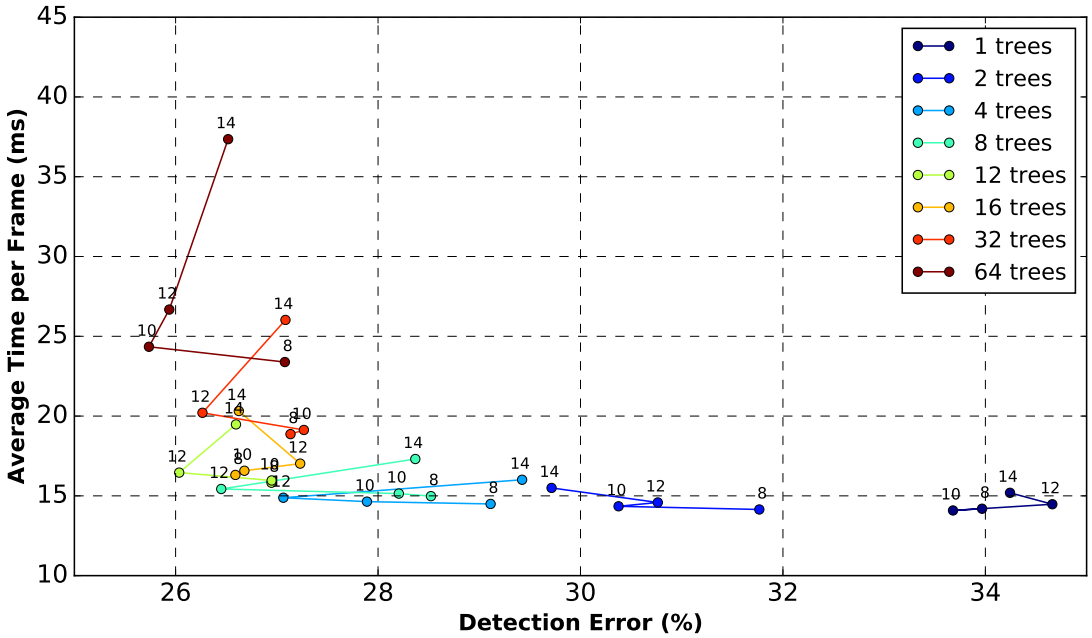


(a) `RIFFIlter` with `grad322motion322_extra` feature set using the `freq` calculation method. Filter parameters: $\lambda^* = 0.4$, $\tau = 2.0$ frames, and $w_{\text{hidden}} = 0.3$.

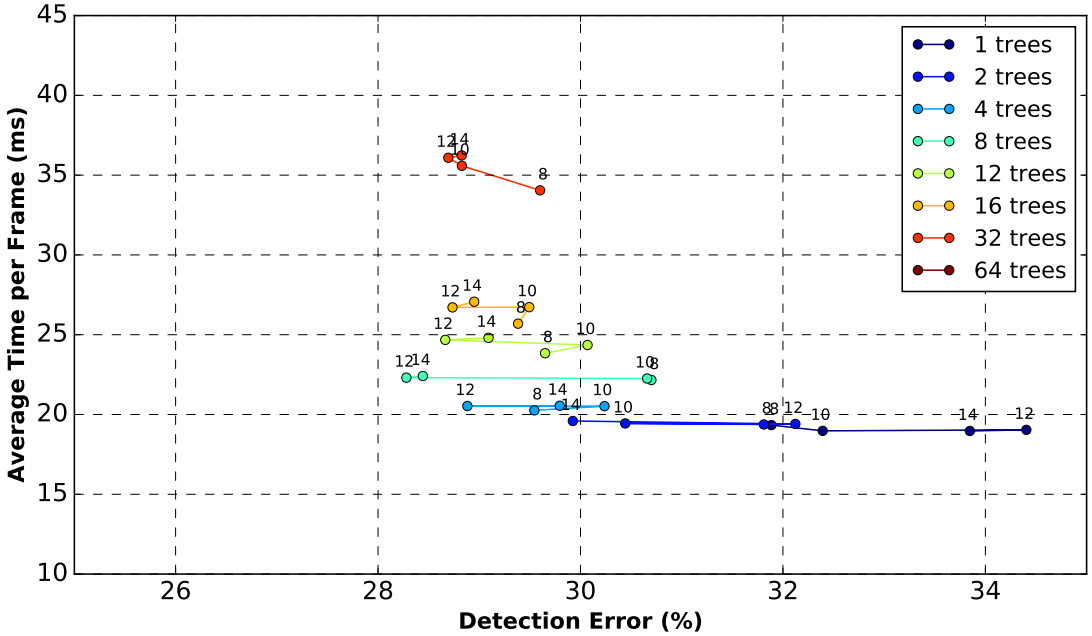


(b) `RECFilter` with `rec_gradmotion_10` feature set. Filter parameters: $\lambda^* = 0.3$, $\tau = 2.0$ frames, and $w_{\text{hidden}} = 0.2$. (Results for 64 are too slow to appear on these axes.)

Figure 7.3: Detection error (§5.1) and calculation time for the `RIFFilter` (7.3a) and `RECFilter` (7.3b) architectures as the composition of the `RIFDetection`/`RECDetection` model varies. In both cases the composition of the phase regression forest (`RIFPhase`/`RECPhase`) was fixed at $N_{\text{trees}} = 32$ and $D_{\max} = 8$, and 1000 particles were used. Annotations represent the maximum number of levels. Results are averaged over three trials on each test video due to the stochastic nature of the filter.



(a) `RIFFIlter` with `grad322motion322_extra` feature set using the `freq` calculation method. Filter parameters: $\lambda^* = 0.4$, $\tau = 2.0$ frames, and $w_{\text{hidden}} = 0.3$.



(b) `RECFilter` with `rec_gradmotion_10` feature set. Filter parameters: $\lambda^* = 0.3$, $\tau = 2.0$ frames, and $w_{\text{hidden}} = 0.2$. (Results for 64 are too slow to appear on these axes.)

Figure 7.4: Detection error (§5.1) and calculation time for the `RIFFilter` (7.3a) and `RECFilter` (7.3b) architectures as the composition of the phase regression forest (`RIFPhase`/`RECPhase`) model varies. In both cases the composition of the classification/detection forest (`RIFDetection`/`RECDetection`) was fixed at $N_{\text{trees}} = 16$ and $D_{\max} = 12$ and 1000 particles were used. Annotations represent the maximum number of levels. Results are averaged over three trials on each test video due to the stochastic nature of the filter.

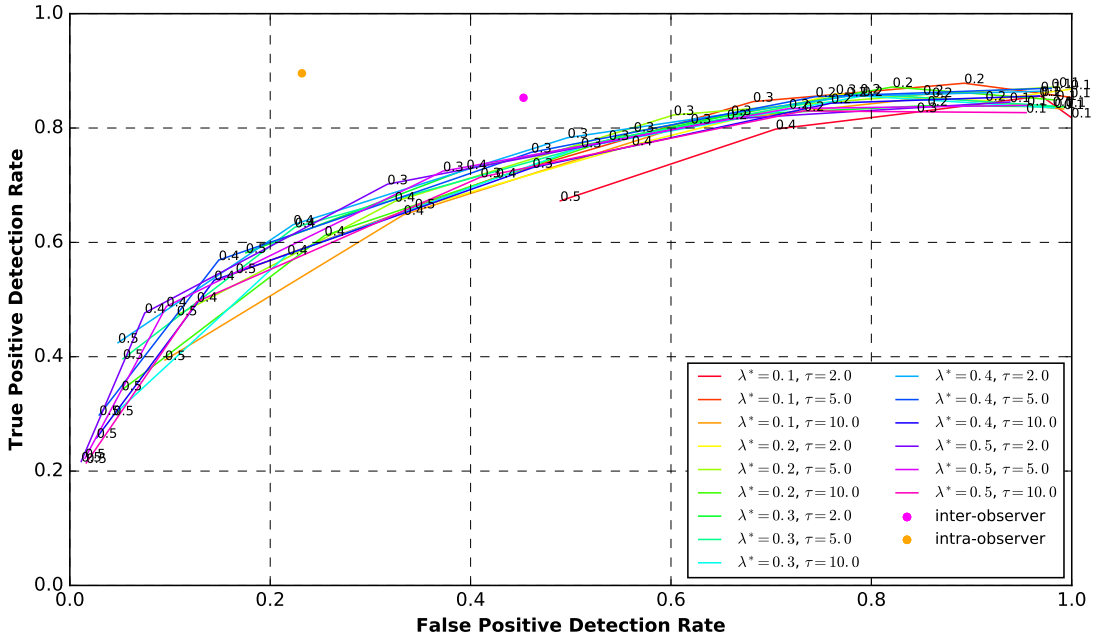
beneficial: in order to have high weights over a sequence of frames a particle must not only get a high classification score in each frame, but must also receive high scores from the cardiac phase observation potential over a sequence of frames. This means that the relevant image patch must change in appearance over the sequence in a way that is compatible with the cardiac phase prediction potential, i.e. must not only look like a heart but beat like one too. The effect of this can be the elimination of other structures that might appear like a heart view in a single frame but do not 'beat' like one, such as acoustic shadowing artefacts (see for example Figure 1.2c or the start of Figure 7.12).

One significant disadvantage of the filtering method is that the estimates it produces are stochastic in nature, and will be different each time the algorithm is run. For this reason, the results shown in Figure 7.3 are averaged over three trials of the entire framework on each video in the dataset. However, there is still significant random variation in the results, which may explain why some of the datapoints in these figures lie away from the trendline.

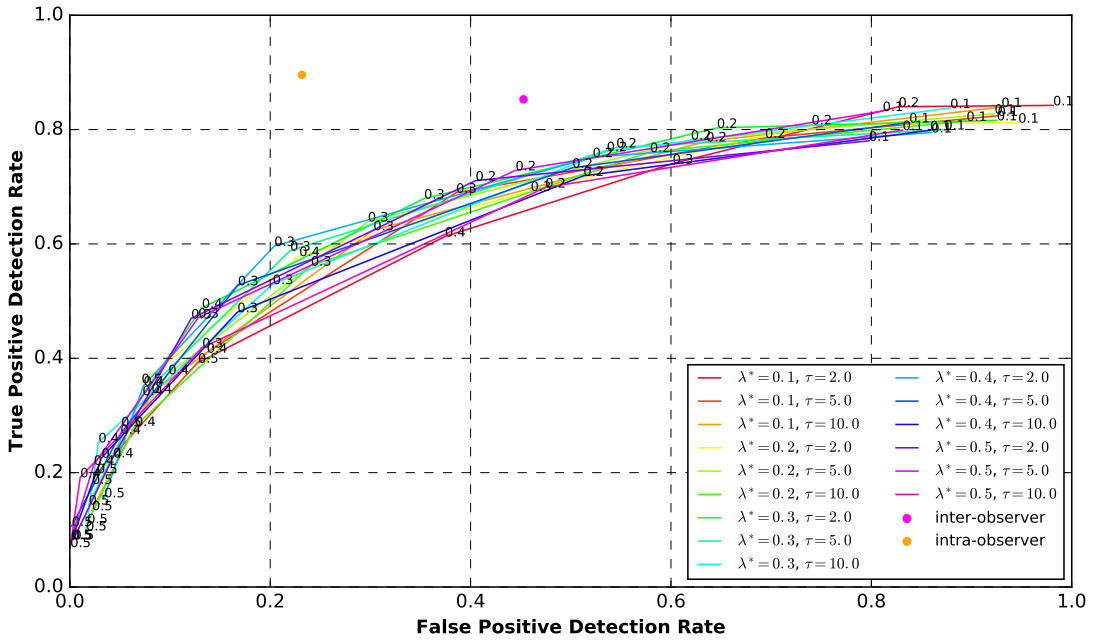
7.3.3 Heart Visibility

The behaviour of a filter with regards to rejecting negative frames (i.e. 'hidden' hearts) is controlled by three parameters: the equilibrium hidden fraction, λ^*, the hidden transition time constant τ, and the hidden weight w_{hidden} (§6.5.2). There is inevitably a trade-off between achieving a high true positive rate, meaning the frames containing a heart are correctly identified as such, and a low false positive rate, meaning that the frames in which the heart is hidden are correctly rejected (see §5.1 for full definitions).

Figures 7.5, and 7.6 illustrate this trade-off by plotting the true positive rate and false positive rate for different values of λ^*, τ, and w_{hidden} for both the `RIFFilter` and `RECFilter` architecture. The results show that there is a fairly wide range of combinations of λ^* and τ that give similar results, and that these results are superior to those for the observation potentials alone (Figure 5.7). In each case, tuning the w_{hidden} parameter gives an effective way to tune the sensitivity of the

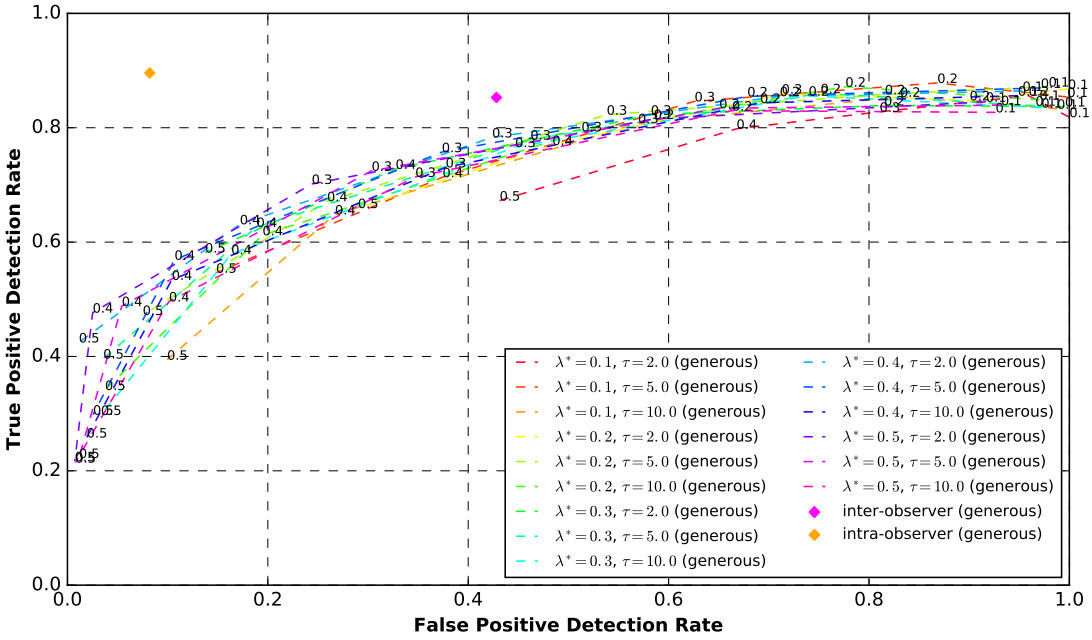


(a) The `RIFilter` architecture with the `grad322motion322_extra` feature set using the `freq` calculation method. The `RIFDetection` model had $N_{\text{trees}} = 16$ and $D_{\max} = 10$ and the `RIFPhase` model had $N_{\text{trees}} = 32$ and $D_{\max} = 8$.

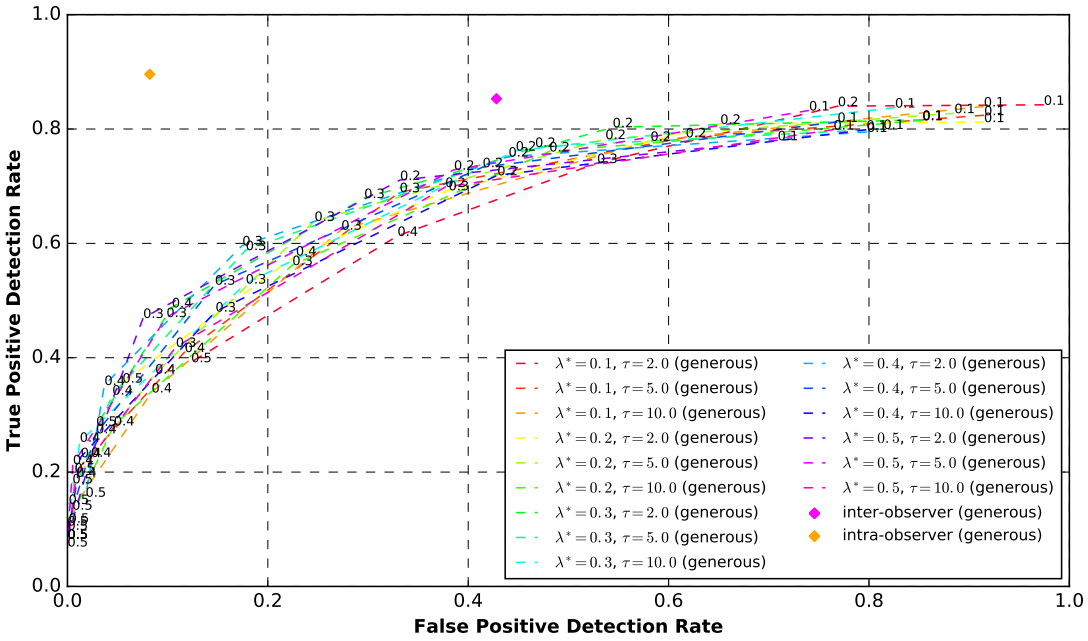


(b) The `RECFilter` architecture with the `rec_gradmotion_10` feature set. The `RECDetection` models had 32 trees and 10 levels and the `RECPhase` models had $N_{\text{trees}} = 64$ and $D_{\max} = 8$.

Figure 7.5: True positive rate and false positive rate (§5.1) for various combinations of the λ^*, τ, and w_{hidden} parameters. Each line shows one combination of λ^* and τ for several values of w_{hidden}. The w_{hidden} values are annotated on the diagram. 1000 particles were used in each case.



(a) `RIFFilter` with parameters as in Figure 7.5a.



(b) `RECFilter` with parameters as in Figure 7.5b.

Figure 7.6: Results for the experiments in Figure 7.5 using the *generous* false positive rate metric (which does not penalise detection of 'obscured' cases, §5.1) rather than the false positive rate metric.

filter, with a higher value rejecting more frames and therefore reducing the number of false positives but also reducing the number of true positives. The choice of the w_{hidden} parameter will therefore depend upon the particular application. For some applications it may be acceptable to miss some of the frames containing the heart provided that those where positive detections are made are correct. For other applications it may be more important to make sure that as many heart frames as possible are identified.

Furthermore, the results suggest that there is little difference between the performance of the `RIFFilter` and `RECFilter` in this regard, except that the filtering parameters must be tuned slightly differently for the two cases.

As in Figure 5.7, there is a slight improvement in the false positive rate when the 'generous' metric, which does not penalise the detection of frames labelled as 'obscured', is used, and this results is a horizontal left shift of the curve between Figures 7.5, and 7.6. This effect is slightly more pronounced in the particle filter case than in the framewise case, suggesting that a greater proportion of the particle filter's mistakes are less problematic detections of borderline cases rather than random errors.

Estimates of the inter- and intra-observer agreement on detection and rejection of frames are shown as dots on Figures 7.5, and 7.6. These show that there is room for improvement in the sensitivity curves. Partly this is simply a result of the fact that it is a highly subjective task to decide whether a heart is 'hidden' or 'visible' when it is obscured by a shadowing artefact, or the probe positioned such that the view of the heart diverges from the three views used. This is illustrated by significant difference between the inter- and intra-observer false positive rates, suggesting that the different annotators used different criteria to determine when to label a heart as 'hidden'.

One approach to tackling the high false positive rate would be to alter the training routine for the detection forest models. In the reported experiments, the models were trained on positive examples and random background patches, and the obscured annotation were excluded from the training dataset. Consequently, the models were not trained to make the fine-grained distinction in borderline cases. It

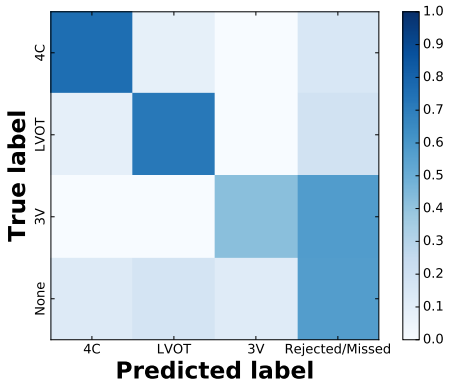
is therefore likely that a training procedure that emphasised using hard negative frames in the training would improve these results.

Another principled approach to dealing with obscured heart cases would be by explicitly modelling the imaging shadowing and drop-out artefacts that are the cause of many of the ambiguous cases. For example, Karamalis et al. [179] proposed an ultrasound 'confidence map' that models the ultrasound physics behind the image acquisition process in order to assess the quality of the image at each pixel. Such as confidence map could be used to determine when the heart is obscured in the image by an imaging artefact.

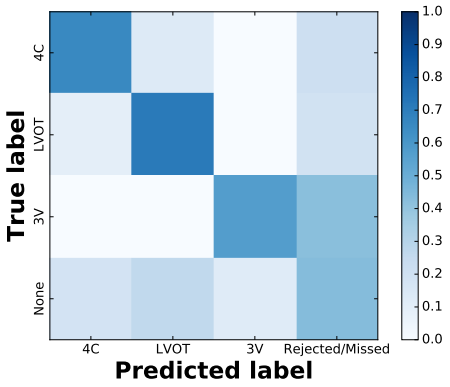
7.3.4 View Confusion Performance

This section analyses the ability of the particle filter to distinguish between the different view labels. Figure 7.7 shows example confusion matrices for the filtering architectures, and compares them with estimates of the intra- and inter-observer confusion calculated from the extra sets of annotations. Again, values of Cohen's kappa statistic are shown for the agreement between the three view classes only. Many of the trends in these figures are the same as those discussed in §5.6.3. The results again show that the 4C and LVOT views are commonly confused, even by the human annotators. The 3V is again by far the most likely view to be missed, it is likely that this is primarily the fault of the observation potentials since the same problem occurs in §5.6.3. However, Figure 7.7d shows that this is also an area where the two human annotators cannot agree. This would possibly improve if the training dataset had a larger number of such images and they could be considered as separate, more clearly defined classes, as in most clinical guidelines.

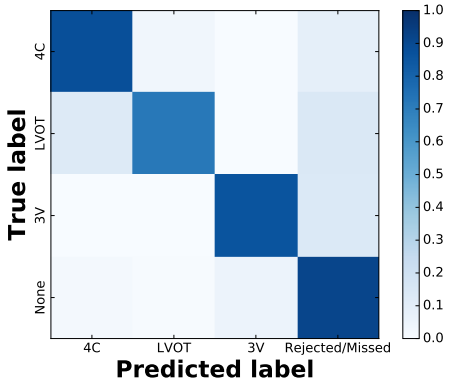
It can also be seen that the majority of the false positives discussed in §7.3.3 are detected as the 4C or LVOT view.



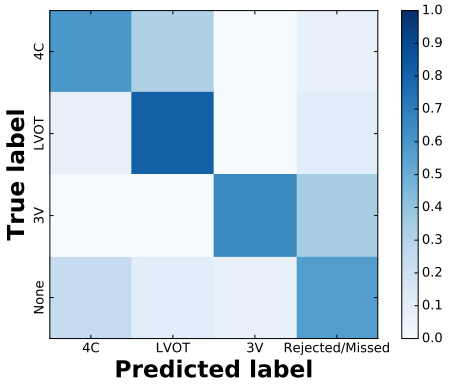
(a) `grad322motion322_extra` with $\lambda^* = 0.4$, $\tau = 2.0$, and $w_{\text{hidden}} = 0.3$, $\kappa = 0.91$.



(b) `rec_gradmotion_10` with $\lambda^* = 0.3$, $\tau = 2.0$, and $w_{\text{hidden}} = 0.2$, $\kappa = 0.83$.



(c) intra-observer, $\kappa = 0.91$.



(d) inter-observer, $\kappa = 0.81$.

Figure 7.7: Typical confusion matrices for `RIFFilter` (7.7a) and `RECFilter` (7.7b) architectures, and estimates of inter-observer and intra-observer confusion. For the automatic predictions, the `RIFDetection`/`RECDetection` forests had $N_{\text{trees}} = 32$ and $D_{\max} = 10$, and the `RIFPhase`/`RECPhase` forest had $N_{\text{trees}} = 32$ and $D_{\max} = 8$. The filter parameters were chosen to give a reasonable trade-off between false positives and misses, based on results in §7.3.3.

7.3.5 Orientation and Cardiac Phase Estimation Performance

Figures 7.8 and 7.9 show how the accuracy of the estimates of orientation and cardiac phase vary with the number of particles. Comparing these results with those in §5.6.5 shows that the use of the particle filter has improved the results for both orientation and cardiac phase and in both architectures, provided that a sufficiently large number of particles is used. This reflects the fact that in both cases, the relationships between the values of the variables in consecutive frames is very strong: the orientation does not tend to change dramatically within a video,

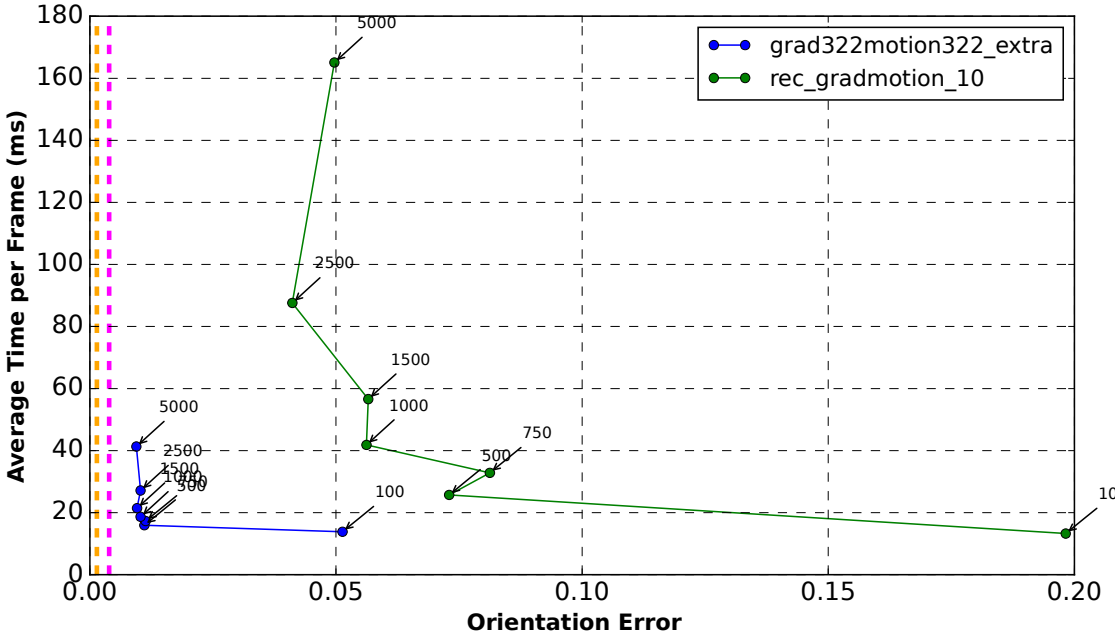


Figure 7.8: Orientation error (§5.1) and calculation time for different numbers of particles. These values were calculated from same trials as shown in Figure 7.2. The orange line shows the estimate of intra-observer variation, the magenta line shows the estimate of inter-observer variation.

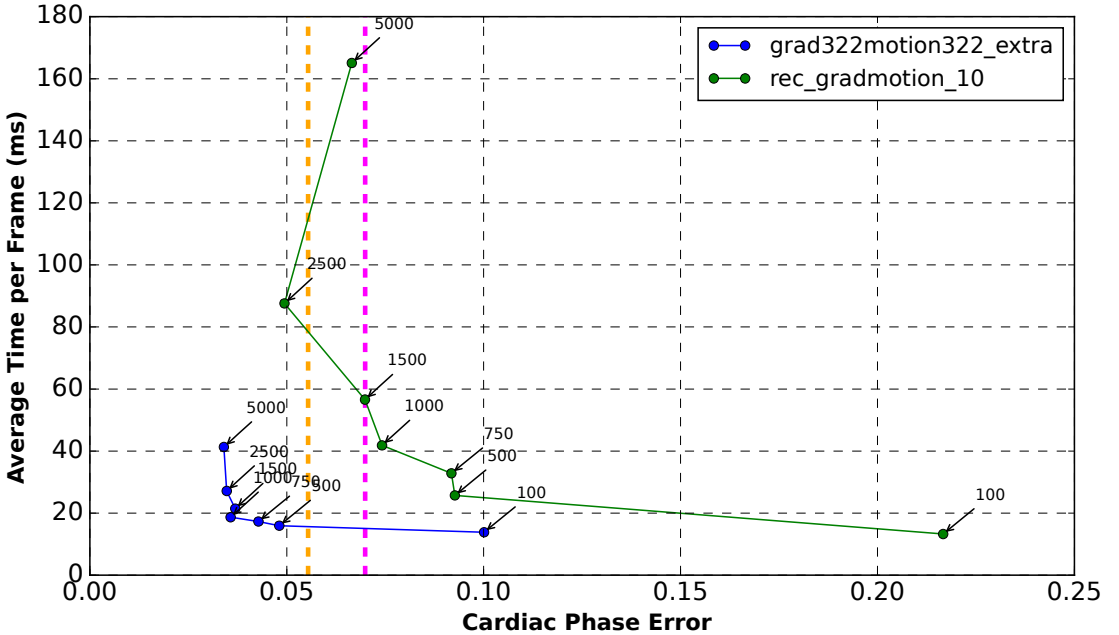


Figure 7.9: Cardiac phase error (§5.1) and calculation time for different numbers of particles. These values were calculated from same trials as shown in Figure 7.2. The orange line shows the estimate of intra-observer variation, the magenta line shows the estimate of inter-observer variation.

even when the probe position changes, and the cardiac phase value changes in a predictable way through the frames. Consequently, the particle filter is able to pool together the weak and noisy measurements in each frame to produce a high-quality, consistent estimate of these variables over the video.

Furthermore, it is clear that in the `RIFFilter` architecture a smaller number of particles is needed for cardiac phase or orientation estimation than is needed for the detection and view classification stage (Figure 7.2). This is a consequence of the fact that the detection and view classification partition has a three dimensional state space (two spatial dimensions and the discrete view category) to populate with particles, whereas the orientation and cardiac phase partition have a one-dimensional (for orientation), or two-dimensional (phase and phase rate for cardiac phase) state space. This observation may be used to speed up the filter by using different numbers of particles in different partitions, something that is suggested by MacCormick and Isard [175], but not explored in this thesis.

In contrast, the `RECFilter` architecture requires more particles to achieve the best performance. This suggests that the additional partitioning of the `RIFFilter` (using three partitions) compared to the `RECFilter` (using two partitions) is able to make better use of a small number of particles, as intended (§6.3.3).

Estimates of the inter- and intra-observer variation are also shown on Figures 7.8 and 7.9. For the cardiac phase estimates, the automatic predictions are within the range of the disagreement between the observers, suggesting that improvement in these values, without more accurate annotations, would be unachievable.

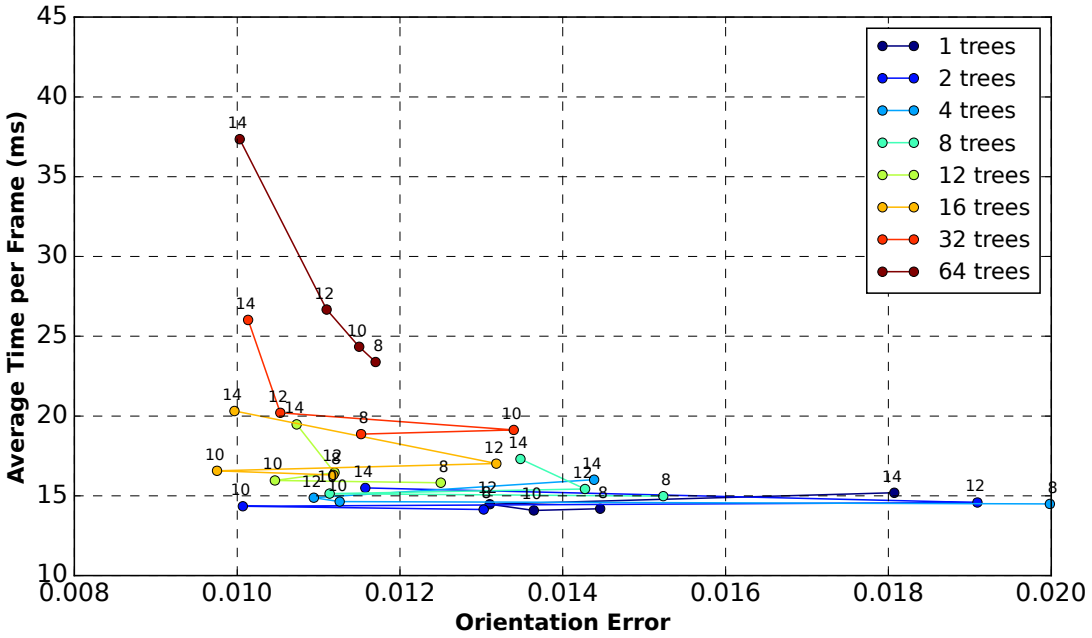
However, the orientation predictions fall short of the inter- and intra-observer variation, suggesting that there is a small amount of room for improvement here. This is primarily due to the fairly simplistic models used in both architectures. The orientation estimation in the `RIFFilter` framework is based on the complex argument of a single raw feature, and is therefore a rather noisy estimate. In the `RECFilter` case, the orientation is evaluated through the way in which the scores from the forests corresponding to the two closest training orientations are combined. However, this assumes a linear relationship, which is a fairly limited assumption.

In both cases, it is likely that the orientation could be improved through the use of more sophisticated models. However, the automatic estimates presented here are likely to be sufficiently accurate for most purposes.

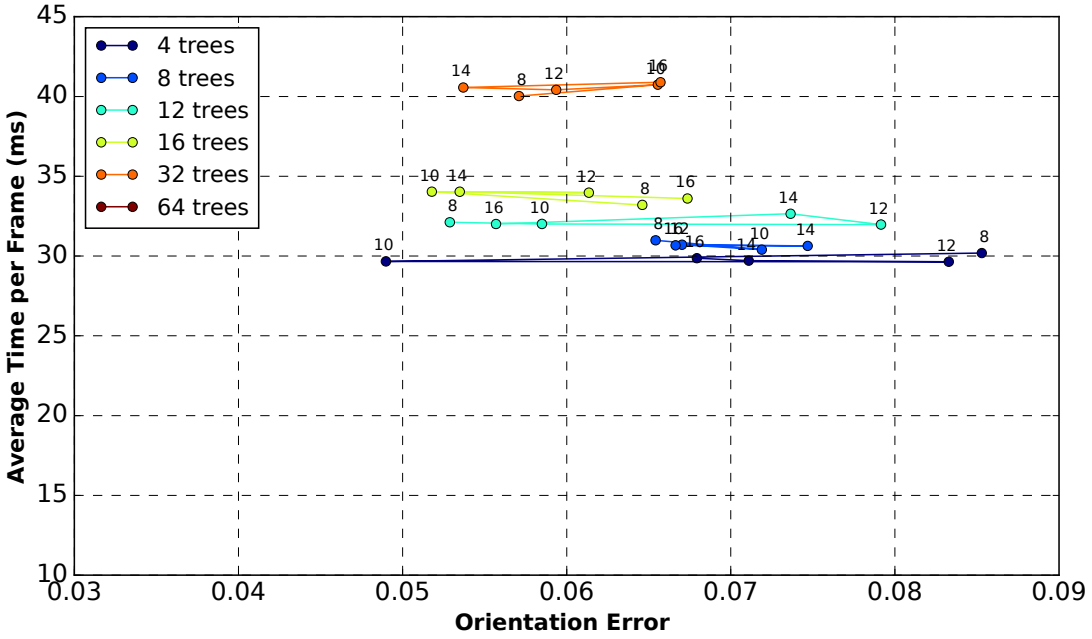
Figures 7.10 and 7.11 show the effect of varying the composition of the relevant forest models on the orientation and cardiac phase errors. The trends in the results are rather weak, which may be a consequence of the fact that altering the composition of the phase regression forests also affects the detection accuracy (Figure 7.4), and the orientation and cardiac phase accuracies are only averaged over correctly detected frames (§5.1). Generally it can be seen that increasing the number of trees does improve accuracy, but not as significantly as it did in the framewise tests (c.f. Figures 5.9, 5.10, and 5.11). The figures reflects the same two trends seen in the detection accuracy: that the particle filtering does not need the most accurate observation potentials to function well, and the calculation time of the `RECFilter` is more strongly adversely affected by adding extra trees than the `RIFFilter` model. The number of levels in the trees was seen to result in very limited effects on estimation accuracy in the framewise experiments, and in the filtering experiments is seen to have fairly unpredictable, small effects.

Generally speaking, the `RIFFilter` gives significantly better orientation estimates than the `RECFilter` model (Figure 7.10). This is likely due to a combination of two factors: *(i)* the fact that the `RIFFilter` produces direct estimates of orientation, unlike the `RECFilter` model in which the observation of orientation uses models trained for detection at different angles, and *(ii)* the partitioning scheme in the `RIFFilter` places the orientation partition at the end of the filtering architecture, meaning that the particles are already clustered in high-probability areas of the location state space before reaching this partition. If a larger number (i.e. finer grid) of training orientations was used, the first of these issues would be reduced at the cost of increased training time and increased memory usage at test time, however the second issue would remain.

The best orientation estimation is seen to have an average error of around 0.01, which corresponds to approximately 11°.

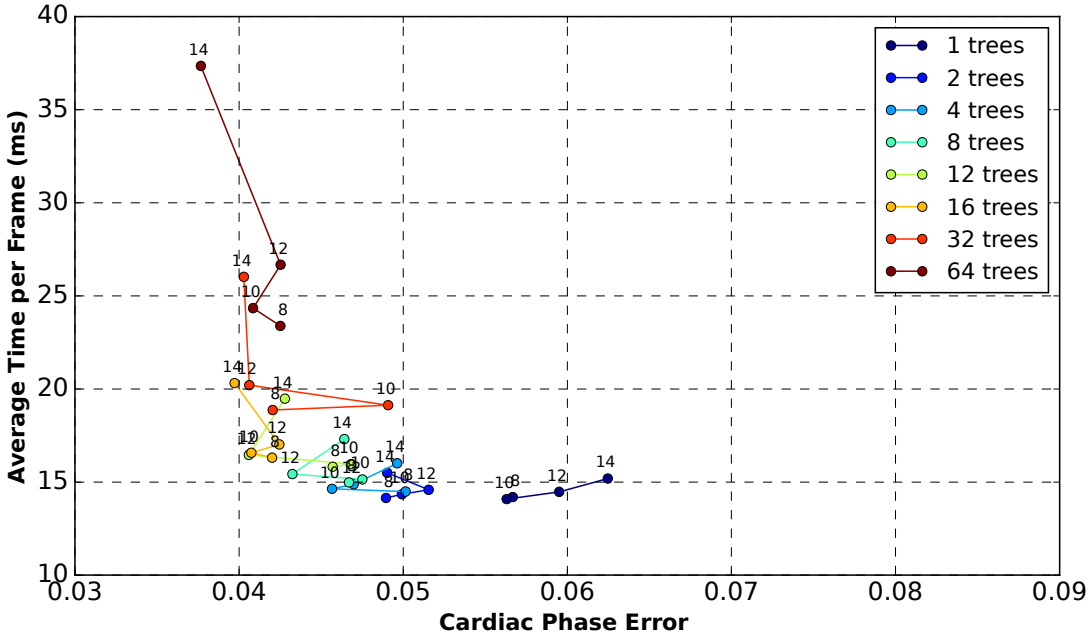


(a) `RIFFilter` with the `grad322motion322_extra` feature set using the `freq` calculation method. Filter parameters: $\lambda^* = 0.4$, $\tau = 2$ frames, $w_{\text{hidden}} = 0.3$. The composition of the `RIFPhase` model is varied.

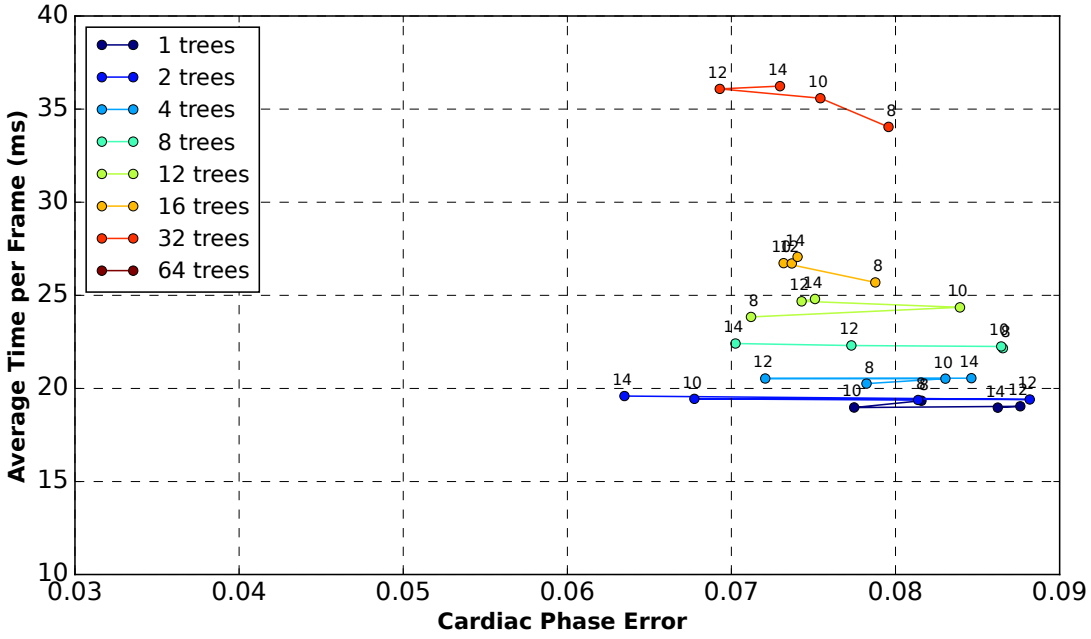


(b) `RECFilter` with `rec_gradmotion_10` feature set. Filter parameters: $\lambda^* = 0.3$, $\tau = 2$ frames, $w_{\text{hidden}} = 0.2$. The composition of the `RECDetection` models is varied.(Results for $N_{\text{trees}} = 64$ are too slow to appear on these axes.)

Figure 7.10: Orientation error (§5.1) and calculation time as the composition of the relevant forest model varies. 1000 particles were used in both experiments. Annotations represent the maximum number of levels, $D_{\max}$. Results are averaged over three trials on each test video due to the stochastic nature of the filter. Note that x-axes do not align due to the significant accuracy difference between the two models.



(a) `RIFFilter` with the `grad322motion322_extra` feature set using the `freq` calculation method. Filter parameters: $\lambda^* = 0.4$, $\tau = 2$ frames, $w_{\text{hidden}} = 0.3$



(b) `RECFilter` with `rec_gradmotion_10` feature set. Filter parameters: $\lambda^* = 0.3$, $\tau = 2$ frames, $w_{\text{hidden}} = 0.2$. (Results for $N_{\text{trees}} = 64$ trees are too slow to appear on these axes.)

Figure 7.11: Cardiac phase error (§5.1) and calculation time as the composition of the `RIFPhase`/`RECPhase` model varies. 1000 particles were used in both experiments. Annotations represent the maximum number of levels. Results are averaged over three trials on each test video due to the stochastic nature of the filter.

The `RIFFilter` is also seen to outperform the `RECFilter` in cardiac phase estimation (Figure 7.11). This is despite the fact that the `RECPhase` forest was shown to be more accurate than the `RIFPhase` model when the ground truth position and orientation was known (§5.6.5). This demonstrates the advantage of using the RIFs for cardiac phase prediction, as this does not rely on having an accurate orientation model when making evaluating the cardiac phase value. The best cardiac phase estimation is seen to have an average error of around 0.04, which corresponds to approximately 0.06 cycles.

7.4 Qualitative Evaluation

There are several qualitative observations worth discussing in addition to the quantitative results presented in §7.3. In this section, these are illustrated with example images and image sequences. The images in Figures 7.12, 7.13, 7.14, 7.15, 7.16, and 7.17 were taken from tests using the `RIFFilter` architecture with the `grad322motion322_extra` feature set and an `RIFDetection` model with $N_{\text{trees}} = 32$ trees and $D_{\max} = 10$ levels, and an `RIFPhase` model with $N_{\text{trees}} = 32$ and $D_{\max} = 8$.

First, it is important to note that a key advantage of the particle filter over the framewise methods presented in Chapter 5 is that the estimates it produces are considerably smoother because it constrains the variation of the state variables between frames. This is particularly important if the estimates are to be used in an application involving live feedback to the sonographer, as smooth estimates are considerably easier for a human to interpret. This is demonstrated quite starkly in the supplementary video[1] to the related journal article [163].

However, one major downside of the filtering approach is that it is vulnerable to what may be called *catastrophic failures* over large parts of the video sequence. This means sections of video where the filter has completely lost track of the true position of the heart, and instead has erroneously locked onto some other area of the image. An example of this is shown in Figure 7.12. Whilst rare, this sort of

[1] `http://www.medicalimageanalysisjournal.com/cms/attachment/2081856061/2072604756/mmc1.mp4` (Open Access)

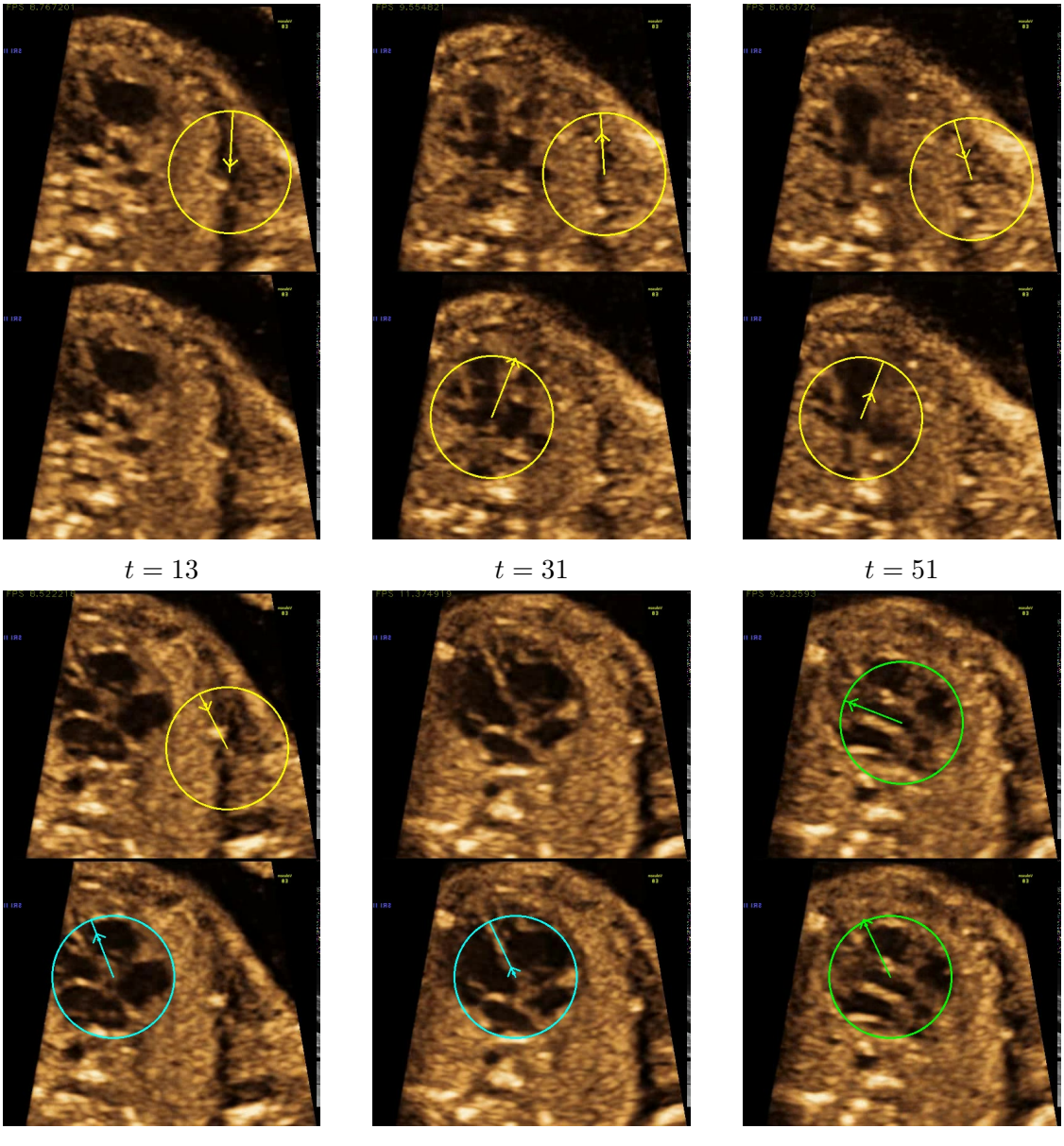


Figure 7.12: Example of a 'catastrophic' failure. At the start of the video, the heart is not visible, but the filter instead locks on to an acoustic shadow artefact, mistaking it for the 3V view (the elongated shadow is similar in appearance to the pulmonary artery in the 3V view). This mistake then persists for over 100 frames while the heart appears on the other side of the image. *Upper half of each image* automatic estimate, *lower half of each image* manual ground truth annotation. See Figure 1.4 for interpretation of the annotations. In addition the arrowhead indicates the cardiac phase variable. Times shown are frame numbers.

failure is the cause of many of the detection errors seen in §7.3. Several factors can reduce the risk of catastrophic failures, though they all have disadvantages: *(i) – many particles* a large number of particles provides better coverage of the state space, making it less likely that the true location is missed, *(ii) – high hidden weight* (w_{hidden}) many catastrophic failures begin when the heart is not visible so the filter erroneously locks onto another area, then when the true heart re-appears, the filter does not recover. A large hidden weight means that in this case there will be large number of hidden particles in the background ready to find the true heart when it re-appears, *(iii) – 'loose' prediction potentials* by overestimating the noise in the prediction potentials, for example the motion noise in position prediction potential, the risk of a catastrophic failure can be reduced due to better coverage of the state space. Another approach that could be explored to reduce the risk of catastrophic failure (described by Thrun et al. [170] among others) is injection of random particles when the average measurement weights become low, however this would have to be adapted to work with the hidden particles framework introduced here, and would reduce the quality of the tracking during normal operation.

Another downside of the filtering method is that it generally takes a few frames for the filter to lock on to the correct area of the state space at the beginning of the video or after a hidden heart re-appears in the video. Consequently there can be several frames of large errors across all the variables while the mean-shift output converges onto the correct value. An example of this is shown in Figure 7.13. The importance of this downside will depend upon the application in question. This can be controlled to an extent by tweaking the filtering parameters. Using 'looser' prediction model parameters (such as a larger standard deviation of the heart position prediction potential, or a larger probability of transition to a different view) will help to speed up this convergence, but will also reduce the quality of the tracking thereafter. It may be possible to adjust these parameters dynamically to improve performance. For example, 'looser' parameters could be used for the first few frames of the video to speed up the initial convergence, and/or 'hidden'

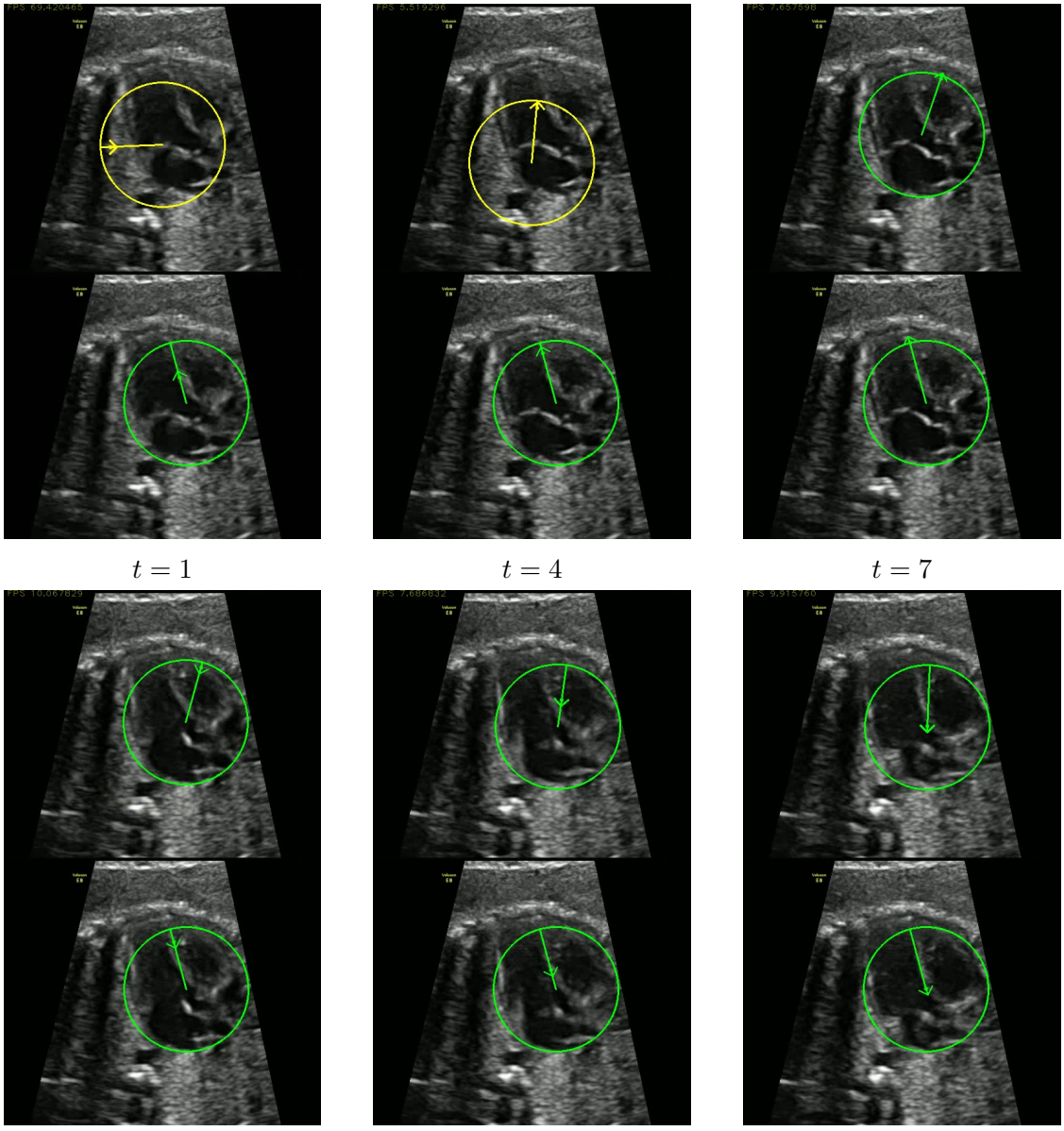


Figure 7.13: Typical example of the global localisation at the start of a video. The filter takes a few frames to slowly converge to the correct state from the initial random particle initialisation. *Upper half of each image* automatic estimate, *lower half of each image* manual ground truth annotation. See Figure 1.4 for interpretation of the annotations. In addition the arrowhead indicates the cardiac phase variable. Times shown are frame numbers.

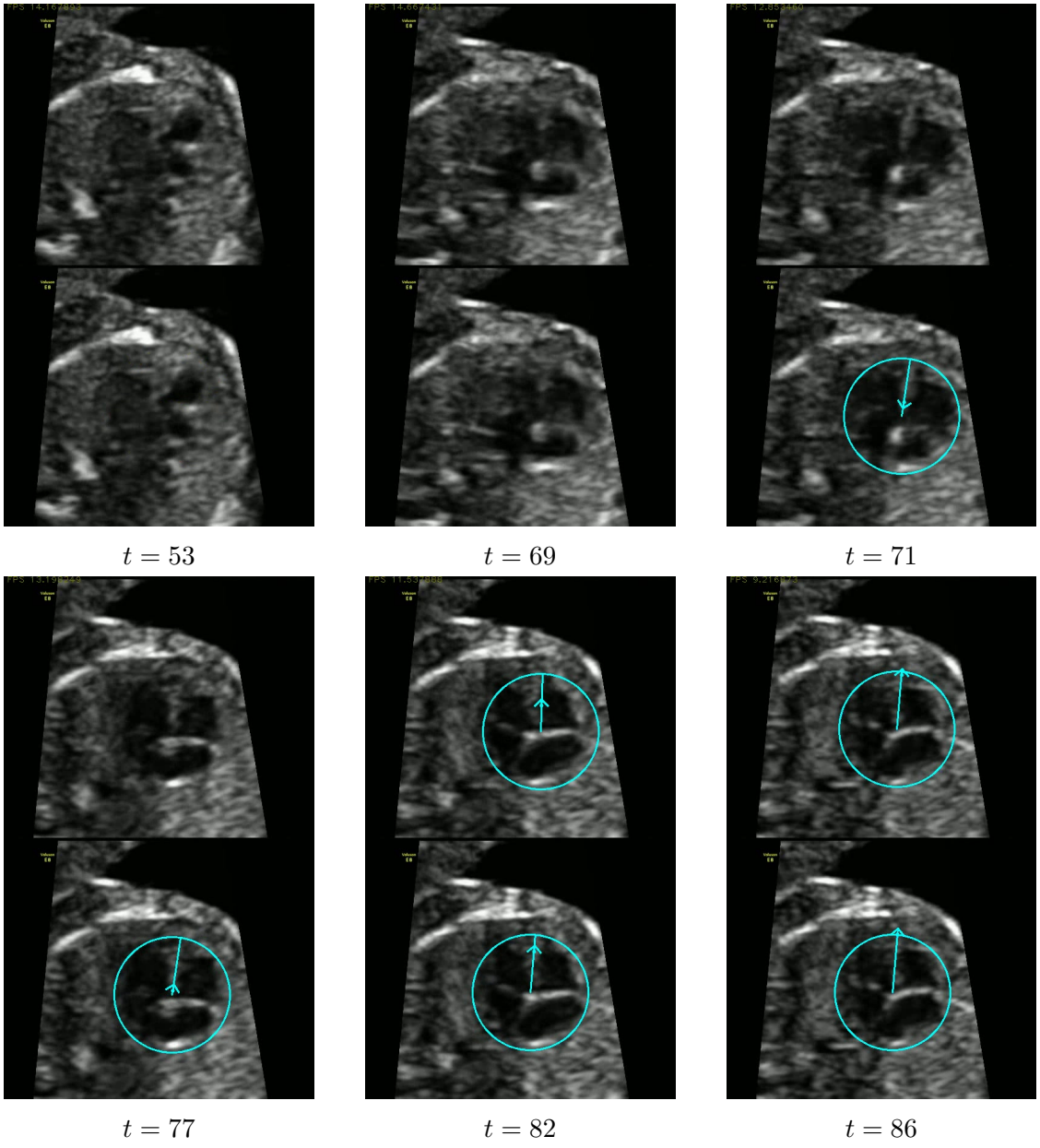


Figure 7.14: Sequence of frames with the heart re-appearing after being obscured. The transition is very gradual, and the automatic estimates and the manual annotations differ in when they decide that the heart has become visible, leading to a several 'missed' frames. *Upper half of each image* automatic estimate, *lower half of each image* manual ground truth annotation. See Figure 1.4 for interpretation of the annotations. In addition the arrowhead indicates the cardiac phase variable.

particles could be given looser parameters than visible particles in order to speed up convergence when the heart (re-)appears midway through a video.

Another important source of errors relates to transitions between visible and hidden hearts. Often this process takes place gradually over several frames. In many

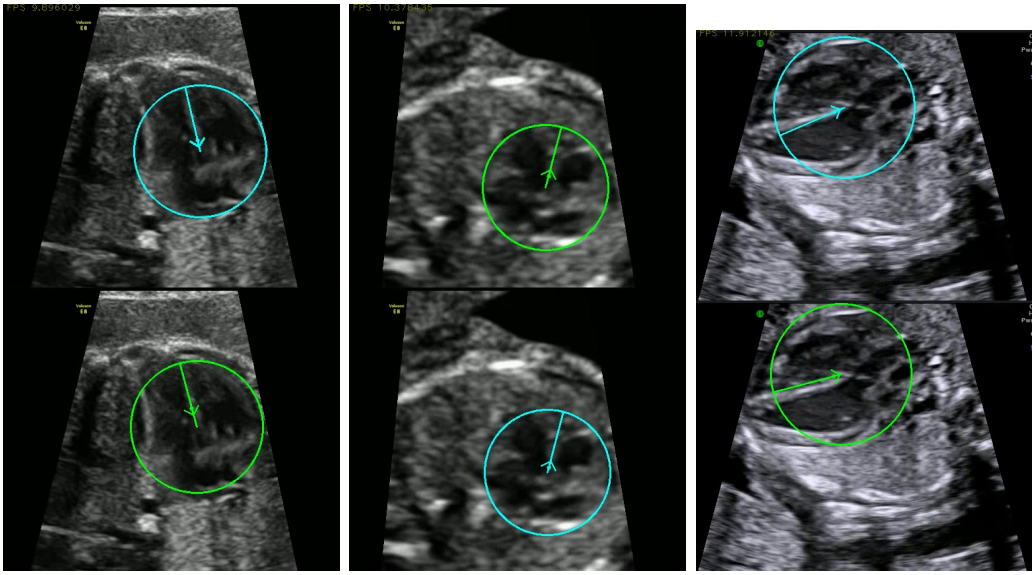


Figure 7.15: Examples of class confusions. The 4C and LVOT are commonly confused because of their similar appearance, especially when the image quality is low due to motion artefacts or other problems. *Upper row* automatic estimate, *lower row* manual ground truth annotation. See Figure 1.4 for interpretation of the annotations. In addition the arrowhead indicates the cardiac phase variable.

cases, the filter's estimates and the manual ground truth agree that a transition has taken place, but disagree about precisely which frame it occurred at, as this is highly subjective. Consequently, several false positives or misses may be recorded during this transitional sequence, but for most applications this discrepancy is unlikely to be important. An example of this is shown in Figure 7.14.

Other common sources of error include class confusion between the 4C and LVOT views due to the similarity between them (Figure 7.15), and missing the 3V view (Figure 7.16). Furthermore, there are sometimes false positives arising for other reasons, and in some cases the reason is not apparent. Some examples of other false positives are shown in Figure 7.17.

7.5 Performance on Portable Hardware

To demonstrate that the method does not require high-end desktop hardware to run at reasonable speeds, the method was also run on an old, lower-specification laptop computer. The computer in question is a mid-range laptop PC dating from 2009, whose specification may be found in the appendix (Table A.2).

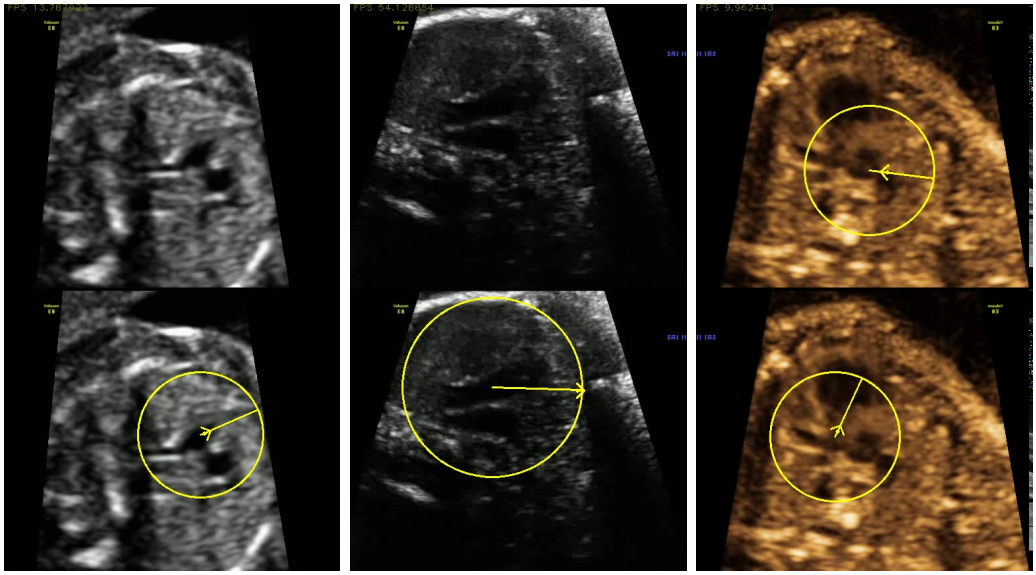


Figure 7.16: Examples of missing or mislocated 3V views. *Upper row* automatic estimate, *lower row* manual ground truth annotation. See Figure 1.4 for interpretation of the annotations. In addition the arrowhead indicates the cardiac phase variable.

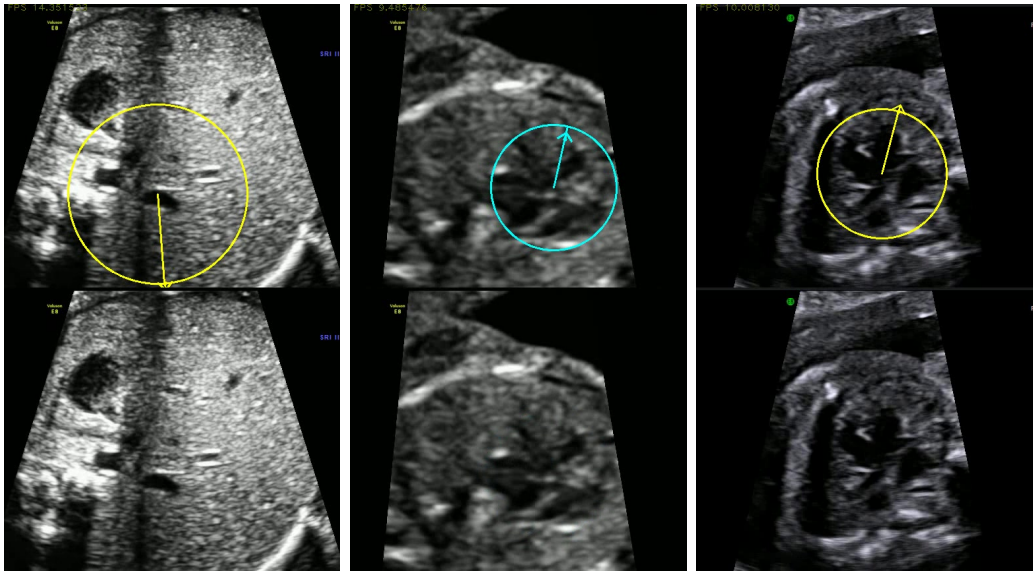


Figure 7.17: Examples of false positive detections. *Upper row* automatic estimate, *lower row* manual ground truth annotation. See Figure 1.4 for interpretation of the annotations. In addition the arrowhead indicates the cardiac phase variable.

A typical set of parameters was used for the test, with the `RIFFilter` architecture using `grad322motion322_extra` features calculated using the `freq` method, with an `RIFDetection` model with 32 trees and a maximum of 10 levels, and an `RIFPhase` model consisting of 32 trees with a maximum of 8 levels.

The average calculation time per frame was found to be 65.8 ms per frame or 15.2 frames per second, which is approaching the frame rate of the most videos, and is certainly fast enough to provide real-time feedback, perhaps by dropping every other frame.

7.6 Conclusions

The results presented in this chapter have demonstrated that the performance of the particle filtering model described in Chapter 6 is significantly better than that of the observation potentials applied independently at each frame. Furthermore, the combination of RIFs and the partitioned filtering architecture that they enable has advantages over the particle filtering method based on traditional rectangular features in terms of both estimation accuracy and running speed. This combination gives rise to an algorithm that both operates high frame rates and has accuracy that is mostly similar to the estimated inter- and intra-observer variation on the challenging clinical dataset. There are also several parameters that can be altered to trade off run speed against accuracy as the application requires.

There is some room for improvement in the orientation estimation accuracy, which may be achieved by designing a more powerful orientation regression model. However the accuracy values presented here are likely to be sufficient for many purposes. There is also some room for improvement in the trade-off between false positive rates and true positive rates, however the significant ambiguity in the definition of the ‘visible’, ‘hidden’ and ‘obscured’ labels suggests that tackling this issue would require a more sophisticated way of labelling the visibility and evaluating the estimates, as well as explicitly training the observation potentials on borderline cases.

It is difficult to define what a 'reasonable' degree of accuracy to expect or aim for in these tasks would be. This is both because of the inherent ambiguity in some of the values being predicted and because different uses of the algorithm's output will place different demands on its accuracy. For example, using the output to feed back basic information to the sonographer will not require particularly accurate results, but using it to steer a detector that target a specific area of the anatomy at a specific point in the cardiac cycle would need to be considerably more accurate. It seems reasonable to assume that an output whose accuracy approaches the inter-observer variation would be sufficiently accurate for the majority of purposes.

Furthermore, for certain purposes the quantitative accuracy of the output may be a poor measure of its quality or usefulness. For example, if the output is used for live feedback, errors with differentiating between the different views are likely to be far more important than lags changing between hidden and visible during view transitions as described in §7.4. In these cases, a better form of evaluation may be to ask expert clinicians to 'score' the algorithm's output subjectively.

The partitioning schemes presented in this work are just two examples of the possible schemes that could be used. For example, one obvious alternative to the `RECFilter` would be to train a model to detect the heart views in any orientation (using learning rather than features to give rotation invariance), followed by an explicit orientation regression partition using a circular regression forest. This would have the advantage of allowing the orientation regression stage to make use of the small appearance changes that occur when the heart is imaged in different orientations. However, it is likely that this would suffer from poor performance at the initial detection stage and would require a larger variety of data to train effectively, but may be worth investigating further.

An interesting extension to this work would be to alter the particle filtering algorithm to give a *particle smoothing* algorithm. This would allow the use of information from *all* frames, including future frames when producing estimates for any frame, and consequently would give an application that operates offline to produce a more accurate set of estimates.

8

Tracking Cardiac Structures in Video Footage

Contents

In this chapter, two extensions to the filtering architecture described in Chapter 6 are presented that enable the estimation of positions of cardiac structures such as valves and vessels. A paper describing this work has been accepted for presentation at the 8$^{\text{th}}$ International Workshop on Machine Learning in Medical Imaging at

MICCAI 2017 [180], though that paper only describes one of the two architectures presented in this chapter (the `PartitionedStructures` architecture).

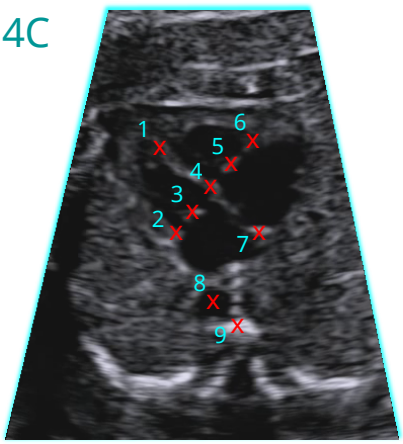
8.1 Introduction

As described in Chapter 1, this thesis aims to estimate the position of cardiac structures in the videos in addition to the 'global' variables considered in previous chapters. The structures of interest considered used here are displayed in Figure 8.1. There is one set of structures for each of the views, chosen to be points of anatomical importance that can reliably be located in the videos as well as potentially useful for diagnostic purposes. For example, the locations of the valves are important for checking for the presence of conditions such as mitral or tricuspid atresia; the locations of the base, crux and apex implicitly define the septa, which need to be checked for septal defects; and the locations of the vessels are necessary to check for problems such as transposition of the great arteries and coarctation of the aorta.

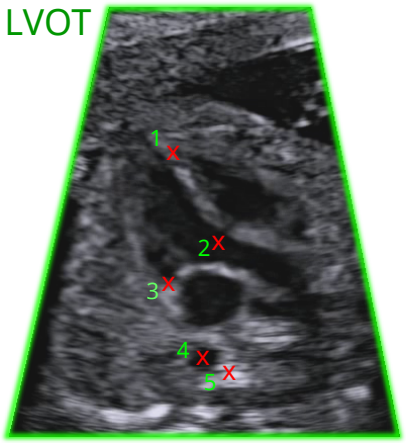
Each structure is assigned an index $a \in \mathbb{N}_0$, and the set of indices that are present in view v is denoted $\mathcal{S}_v$. The location of the structure with index a in the image at time t is represented by a 2D vector $\mathbf{q}_{a,t} \in \mathbb{R}^2$. Additionally, there is a *visibility* variable, $g_{a,t} \in \{0, 1\}$, associated with each structure that reflects whether the structure is visible in the image or hidden due to factors such as imaging artefacts (particularly acoustic shadows), the cardiac cycle, or being located off the edge of the image.

As each structure is represented by a 2D position variable and a binary visibility variable and there are several such structures, the inclusion of these additional state variables directly into the particle filtering framework described in Chapter 6 would dramatically increase the dimensionality of the state space. Consequently, a straightforward particle filter implementation would require a very large number of particles to sufficiently populate the high-dimensional state space and would therefore be computationally very expensive.

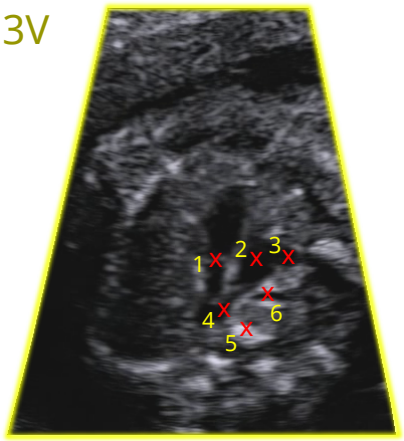
This chapter presents two extensions to the filtering architectures from Chapter 6, shown in Figure 8.2, that attempt to overcome this problem. One model,



1. Apex. Apical terminus of interventricular septum.
2. Mitral Valve End. Insertion of posterolateral valve leaflet.
3. Mitral Valve Centre. Meeting point of the two valve leaflets.
4. Crux Cordis. Intersection of the antrioventricular plane (separating the atria and ventricles) and septal plane (separating the left and right heart).
5. Tricuspid Valve Centre. The meeting point of the three valve leaflets.
6. Tricuspid Valve End. Insertion of the valve leaflet on left heart wall.
7. Base. Basal terminus of the atrial septum.
8. Descending Aorta (Centre).
9. Spine. Anterior ossification centre.



1. Apex. Apical terminus of interventricular septum.
2. Aortic Valve. Meeting point of the two valve leaflets.
3. Mitral Valve End. Insertion of the posterolateral valve leaflet.
4. Descending Aorta (Centre).
5. Spine. Anterior ossification centre.



1. Pulmonary Valve. Meeting point of the two valve leaflets.
2. Ascending Aorta (Centre).
3. Superior Vena Cava (Centre).
4. Descending Aorta (Centre).
5. Spine. Anterior ossification centre.
6. Trachea (Centre).

Figure 8.1: List of cardiac structures in each view and description of their location in the image.

`PCAStructures` (Figure 8.2a), uses one further partition that applies dimensionality reduction to reduce the state dimension of the set of structure locations. The second model, `PartitionedStructures` (Figure 8.2b), uses one additional partition per structure of interest, thereby reducing the number of particles necessary for efficient filtering. In both cases, the extra partitions may be added onto the end of the either the `RIFFilter` or `RECFilter` architecture after the existing partitions.

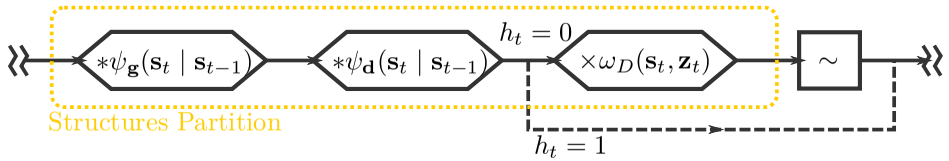
As with the filters in Chapter 6, these filters use observation potentials based on random forest models in order to evaluate the compatibility of a state value with the observed image information. These are described in §8.2. Both models make use of a Fourier model to capture the periodic nature of the cardiac cycle, which is described in §8.4. Then the two architectures are described in detail in §8.5 and §8.6.

8.2 Random Forest Models for Structure Detection

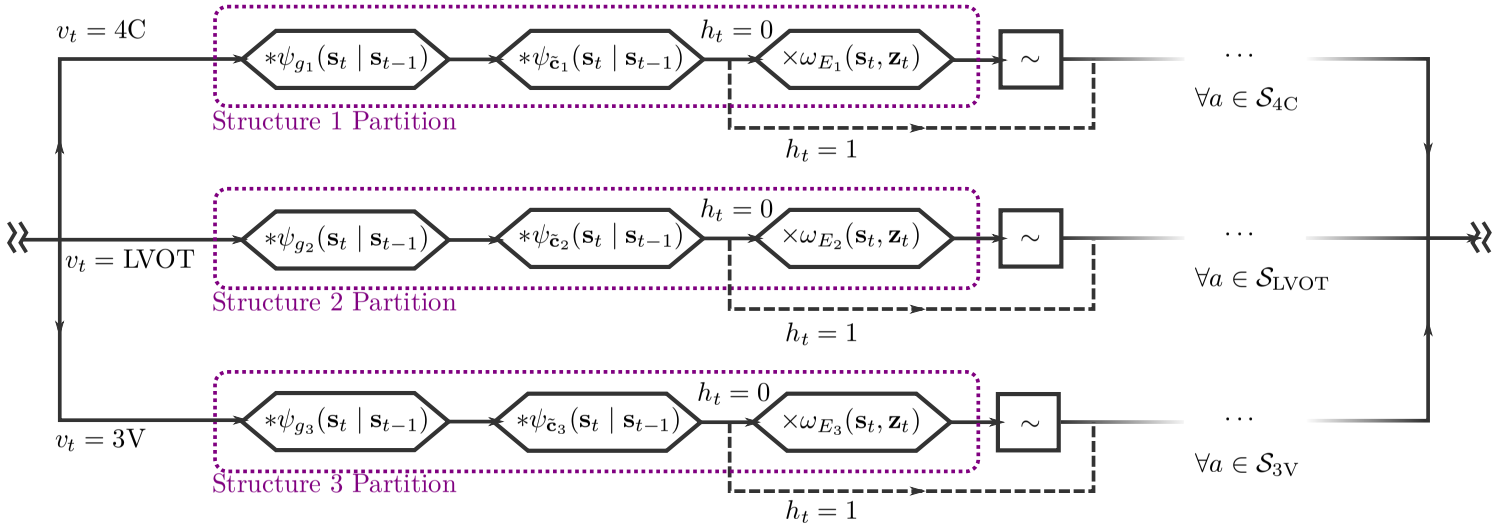
As previously, architectures using both rotation-invariant features (RIFs, Chapter 3) and traditional rectangular features are explored. In both cases, the forests are designed such that at test time, they can share much of the required preprocessing with the forests used for the previous stages of the filtering.

The following random forest models are used for the observation potentials introduced later in this chapter:

Rotation Invariant Structure Detection Forest (`RIFStructures`): This is a classification forest (§4.2) with an output space consisting of the S structure classes and a background label (abbreviated BG), and thus is used to detect the structures within the image. It is trained using a set of patches centred on the labelled positions of the the structures, and a set of random patches chosen from the video dataset as background patches. The model is invariant to the orientation of the heart and structures due to the nature of the RIFs. It is trained with a training set drawn from all points in the cardiac cycle, and as such should be relatively robust to the image variation that it introduces.



(a) Schematic of the `PCAStructures` architecture.



(b) Schematic of the `PartitionedStructures` architecture.

Figure 8.2: Schematics of the `PCAStructures` and `PartitionedStructures` architectures. These are attached to either the `RECFilter` or `RIFFilter` filters at the corresponding symbols in Figure 6.6. The `PCAStructures` architecture appends one further partition that relates to the positions of all the structures of interest. The `PartitionedStructures` architecture appends one additional partition for each of the structures of interest, arranged in three branches, one for each of the three heart views because different structures are present in each view. To simplify the diagram, it is assumed that structure indices 1,2 and 3 are the first indices belonging to the three views (4C, LVOT and 3V respectively), though in practice the specific ordering used is unimportant.

In each experiment, the RIF feature set used is based on the set used for the `RIFDetection`, `RIFPhase`, and `RIFOri` models used in that experiment. However, the `RIFStructures` forest is constrained to use only features that are derived from raw features with a radial index j that is less than some fixed value J_{structs} where this is less than the maximum radial index in the feature set ($J_{\text{structs}} < J$). This means that the features are gathered from a circular area with a radius that is smaller than the heart radius, reflecting the fact that the structures are more localised (and allowing them to be detected closer to the edge of the image than the heart centre) whilst allowing many of the features to be re-used by all the forest models.

Rectangular Structure Detection Forests (`RECStructures`): These are also classification forests trained with the same output labels and training set as the `RIFStructures` forests. However, they make use of rectangular filters (§5.4) instead of RIFs. Consequently, and as with the `RECDetection` and `RECPhase` models, it is necessary to train these models at a number of different heart orientations. In each experiment, these are the same N_{o} training orientations that are used for the `RECDetection` and `RECPhase` models. At test time, the forest models can share the integral images used for fast evaluation. However, the window size from which the features are drawn, $R_{\text{struct-train}}$, is smaller than the heart radius R_{train}.

8.3 Summary of Notation

Because of the complexity of notation in this chapter, a summary is provided in Table 8.1 for reference.

8.4 Fourier Position Model for a Cardiac Structure

Due to the nature of the cardiac cycle, over a short time interval the positions of the structures are likely to be close to periodic. Furthermore, an estimate of

Symbol	Meaning
$a \in \mathbb{N}_0$	The index of a structure.
$v \in \mathbb{N}$	The index of a cardiac view.
$t \in \mathbb{N}_0$	The time-step (frame index) in a video.
$\mathbf{q}_{a,t} \in \mathbb{R}^2$	The absolute position of structure a at time-step t in the image.
$\mathbf{p}_{a,t} \in \mathbb{R}^2$	The relative position of structure a at time-step t in the image, relative to the heart position, radius and orientation.
$b_a \in \mathbb{N}_0$	The Fourier expansion order for structure a.
$\mathbf{c}_{a,1}$ and $\mathbf{c}_{a,2} \in \mathbb{R}^{2b_a+1}$	Fourier coefficient vectors for structure a, one for each component of the relative position vector.
$\boldsymbol{\varphi}_t \in \mathbb{R}^{2b_a+1}$	Vector containing a sine and cosine expansion of the cardiac phase value ϕ_t at time t.
$\mathcal{S}_v$	The set of structure indices for the structures in view v.
$B_v \in \mathbb{N}_0$	The total number of Fourier coefficients when combining those from all structures in view v.
$\hat{\mathbf{c}}_v \in \mathbb{R}^{B_v}$	A combined coefficient vector, comprising all the Fourier coefficients from all the structures in view v.
$D \in \mathbb{N}$	The dimensionality of the reduced representation of a combined coefficient vector.
$\mathbf{d}_v \in \mathbb{R}^D$	A reduced representation of the combined coefficient vector for view v.
$\mathbf{M}_v \in \mathbb{R}^{B_v \times D}$	Matrix containing the principal axes of the combined coefficient vector view v.
$\overline{\mathbf{c}}_v \in \mathbb{R}^{B_v}$	Mean of the combined coefficient vector for view v.
$d_a \in \mathbb{N}_0$	The total number of Fourier coefficients for a single structure a.
$\tilde{\mathbf{c}}_{a,t} \in \mathbb{R}^{d_a}$	A combined coefficient vector comprising all the Fourier coefficients from a single structure a.
$\tilde{\boldsymbol{\mu}}_a \in \mathbb{R}^{d_a}$	Mean of $\tilde{\mathbf{c}}_{a,t}$ at for structure a across the entire training set.
$\tilde{\boldsymbol{\Sigma}}_a \in \mathbb{R}^{d_a \times d_a}$	Covariance matrix of $\tilde{\mathbf{c}}_{a,t}$ at for structure a across the entire training set.
$\alpha \in \mathbb{R}$	Update scaling constant.
q and $\gamma \in \mathbb{R}$	Noise scaling constants.
$g_{a,t} \in \{0, 1\}$	Visibility of structure a in frame t.
$\mathbf{g}_t \in \{0, 1\}^S$	Vector containing visibility variables of all S structures at time t.

Table 8.1: Summary of notation used throughout this chapter.

the cardiac phase variable, ϕ_t, for each particle is available from the cardiac phase partition. Rather than estimate a structure's position over the cardiac cycle in each frame independently, a simple Fourier model is used to capture this periodic behaviour over a sequence of frames.

In this model, the position of structure $a \in \mathbb{N}_0$, where a is an index variable indexing the various structures (§8.1), in the image at time t is described by the 2D column vector $\mathbf{q}_{a,t} \in \mathbb{R}^2$ containing the x and y components, i.e. $\mathbf{q}_{a,t} = \begin{bmatrix} q_{a,t,1}, q_{a,t,2} \end{bmatrix}^T$, where $q_{a,t,1} \in \mathbb{R}$ is the x-component and $q_{a,t,2} \in \mathbb{R}$ is the y-component. Firstly, this is expressed relative to the heart centre position, orientation and scale to give the *relative position vector* $\mathbf{p}_{a,t} \in \mathbb{R}^2$, where the two are related by:

$$\mathbf{q}_{a,t} = R_{\text{test}} \mathbf{R}_{[\theta_t]} \mathbf{p}_{a,t} + \mathbf{x}_t \tag{8.1}$$

where $\mathbf{R}_{[\theta_t]} \in \mathbb{R}^{2\times 2}$ is the 2D rotation matrix through angle θ_t.

The relative position vector $\mathbf{p}_{a,t}$ is calculated from the current value of the cardiac phase variable, ϕ_t, by assuming a truncated Fourier series approximation as follows:

$$\mathbf{p}_{a,t} = \begin{bmatrix} c_{a,1,1} & c_{a,2,1} \\ c_{a,1,2} & c_{a,2,2} \\ c_{a,1,3} & c_{a,2,3} \\ c_{a,1,4} & c_{a,2,4} \\ c_{a,1,5} & c_{a,2,5} \\ \vdots & \vdots \end{bmatrix}^T \cdot \begin{bmatrix} 1 \\ \cos\phi_t \\ \sin\phi_t \\ \cos 2\phi_t \\ \sin 2\phi_t \\ \vdots \end{bmatrix} \tag{8.2}$$

$$= \begin{bmatrix} \mathbf{c}_{a,1} & \mathbf{c}_{a,2} \end{bmatrix}^T \cdot \boldsymbol{\varphi}_t \tag{8.3}$$

If an approximation of order $b_a \in \mathbb{N}_0$ is used (different orders may be used for different structures), then the phase vector $\boldsymbol{\varphi}_t$ has dimension $(2b_a + 1) \times 1$ and there are two coefficient column vectors $\mathbf{c}_{a,1}$ (for the x-component) and $\mathbf{c}_{a,2}$ (for the y-component) each having dimension $(2b_a + 1) \times 1$.

In order to fit such a model to examples from training data, a least squares approach is used. Suppose that a training set of N consecutive frames is used, and that the relative positions $\mathbf{p}_{a,t}$ of the structure and the cardiac phase ϕ_t value

are known in each frame. The two coefficient vectors $\mathbf{c}_{a,1}$ and $\mathbf{c}_{a,2}$ are solved for separately by grouping the x- and y-components from all the samples into two vectors, $\hat{\mathbf{p}}_{a,1} \in \mathbb{R}^N$ containing the x-components from all samples, and $\hat{\mathbf{p}}_{a,2} \in \mathbb{R}^N$ containing the y-components from all samples:

$$\hat{\mathbf{p}}_{a,1} = \begin{bmatrix} p_{a,1,1} \\ p_{a,2,1} \\ \vdots \\ p_{a,N,1} \end{bmatrix}, \quad \text{and} \quad \hat{\mathbf{p}}_{a,2} = \begin{bmatrix} p_{a,1,2} \\ p_{a,2,2} \\ \vdots \\ p_{a,N,2} \end{bmatrix} \tag{8.4}$$

Also, the phase vectors $\boldsymbol{\varphi}_t$ from all the training samples are stacked into a phase matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N \times (2b+1)}$:

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}_1^T \\ \boldsymbol{\varphi}_2^T \\ \vdots \\ \boldsymbol{\varphi}_N^T \end{bmatrix} \tag{8.5}$$

Then the coefficient vectors $\mathbf{c}_{a,1}$ and $\mathbf{c}_{a,2}$ may be found by solving the following equation in the least squares sense using a standard least squares solver:

$$\hat{\mathbf{p}}_{a,1} = \boldsymbol{\Phi} \cdot \mathbf{c}_{a,1}, \quad \text{and} \quad \hat{\mathbf{p}}_{a,2} = \boldsymbol{\Phi} \cdot \mathbf{c}_{a,2} \tag{8.6}$$

However in some situations, there will be only a few training samples or poor coverage of the whole cardiac cycle (particularly with structures such as valves that are not visible for large parts of the cycle). This can lead to overfitting of the coefficients. One way to avoid this is to use a low order Fourier model, but it was also found to be beneficial to regularise the solution using a standard regularised least squares method [172]. A diagonal prior *precision matrix* $\boldsymbol{\Lambda} \in \mathbb{R}^{(2b_a+1)\times(2b_a+1)}$ is specified over the coefficient vectors to penalise large values for the coefficients, except the first coefficient (corresponding to the average x or y location.)

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & & & \\ & \lambda & & \\ & & \ddots & \\ & & & \lambda \end{bmatrix} \tag{8.7}$$

where $\lambda \in \mathbb{R}_0^+$ is a chosen *precision* (reciprocal variance) value.

The coefficient vectors are then found by instead solving the standard least squares problem:

$$\begin{bmatrix} \hat{\mathbf{p}}_{a,1} \\ \mathbf{0}_{2b+1} \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi} \\ \sqrt{\mathbf{\Lambda}} \end{bmatrix} \cdot \mathbf{c}_{a,1}, \quad \text{and} \quad \begin{bmatrix} \hat{\mathbf{p}}_{a,2} \\ \mathbf{0}_{2b_a+1} \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi} \\ \sqrt{\mathbf{\Lambda}} \end{bmatrix} \cdot \mathbf{c}_{a,2} \tag{8.8}$$

where $\mathbf{0}_{2b_a+1}$ is a $(2b_a + 1) \times 1$ vector of zeros, and $\sqrt{\mathbf{\Lambda}}$ can be obtained by square rooting the on-diagonal elements.

Using the methods outlined in this section, a Fourier model, defined by $\mathbf{c}_{a,1}$ and $\mathbf{c}_{a,2}$, can be fitted independently for each of the structures of interest. However the Fourier model will vary between different sections of video due to anatomical variation between the subjects and slight variations in viewing plane within a video. The model for each of the structures of interest will also be closely connected to each other, because the spatial arrangements of the structures are tightly constrained.

The two filtering models presented in §8.5 and §8.6 are similar in that they consider the coefficients of the Fourier model to be variables to track over time, rather than directly tracking the positions of the structures. However they differ in the way that they represent the relationships between the coefficients of the different structures.

8.5 The PCAStructures Model

This architecture makes use of a single filtering partition to jointly track the Fourier model coefficients of all the structures of interest. The schematic representation is shown in Figure 8.2a. This is a very high-dimensional filtering problem ($2(2b_a + 1)$ coefficients for each structure) and therefore a *naïve* particle filtering approach is likely to perform very poorly. Furthermore, it is necessary to model the spatial relationships between the different structures at one timestep.

This architecture uses principal component analysis (PCA) to both reduce the dimension of this set of coefficients and model the relationships between them.

8.5.1 Definition of the State

PCA is a widely-used mathematical technique to find a compact representation of a dataset that optimally captures the variation within the training data (e.g. [172] or many other standard texts). Here, it used to find a compact representation of the coefficient vectors $\mathbf{c}_{a,1}$ and $\mathbf{c}_{a,2}$. This occurs separately for each of the three views, as different structures appear in each.

Suppose that there are N short video segments (each at least one cardiac cycle in length but typically no longer than two or three cycles) containing a single cardiac view v, and that a Fourier model has been fitted to each of the structures present in that view (the set $\mathcal{S}_v$) as described in §8.4. This gives $\{(\mathbf{c}_{a,1}, \mathbf{c}_{a,2})\}_{a\in\mathcal{S}_v}$ for each of the N video segments, which combined represents a large collection of coefficients for each of the N segments. The number of coefficients B_v used for each view is given by:

$$B_v = 2\sum_{a\in\mathcal{S}_v}(2b_a + 1) \tag{8.9}$$

All the coefficients for a given segment n and view v are placed into a single *combined coefficient vector* $\hat{\mathbf{c}}_{n,v} \in \mathbb{R}^{B_v}$, which combines the coefficients for all Fourier orders and all structures in the view (the order in which these coefficients are placed into the vector $\hat{\mathbf{c}}_{n,v}$ is unimportant, as long as the corresponding unpacking operation is applied at test time). The mean combined coefficient vector is then subtracted from each coefficient vector and they are stacked to form a matrix $\hat{\mathbf{C}}_v \in \mathbb{R}^{N\times B_v}$ as follows:

$$\hat{\mathbf{C}}_v = \begin{bmatrix} \hat{\mathbf{c}}_{1,v}^T - \overline{\mathbf{c}}_v^T \\ \hat{\mathbf{c}}_{2,v}^T - \overline{\mathbf{c}}_v^T \\ \vdots \\ \hat{\mathbf{c}}_{N,v}^T - \overline{\mathbf{c}}_v^T \end{bmatrix} \tag{8.10}$$

where $\overline{\mathbf{c}}_v$ is the mean combined coefficient vector across all N training segments.

PCA proceeds by performing the singular value decomposition (SVD) of this matrix to decompose it into $\mathbf{U}_v \in \mathbb{R}^{N\times N}$, an orthogonal matrix, $\mathbf{S}_v \in \mathbb{R}^{N\times B_v}$, a rectangular diagonal matrix containing the singular values along the leading diagonal, and $\mathbf{V}_v \in \mathbb{R}^{B_v\times B_v}$, an orthogonal matrix:

$$\hat{\mathbf{C}}_v = \mathbf{U}_v \mathbf{S}_v \mathbf{V}_v^T \tag{8.11}$$

In what follows, it will be assumed that the rows of $\mathbf{V}_v^T$ (columns of $\mathbf{V}_v$) are normalised (i.e. have $\mathcal{L}_2$ norms of 1) and the singular values appear down the diagonal of $\mathbf{S}$ in order of decreasing size.

Now the matrix $\mathbf{V}_v$ provides a mapping from a principal component representation of a coefficient vector, $\mathbf{d}_v$, to the centred original representation of the vector. To create a compact representation of the coefficient vectors, the least important principal components (those with the smallest singular values), can be discarded, reducing the dimensionality of $\mathbf{d}_v$. If the dimension of $\mathbf{d}_v$ is chosen to be D, then the first D columns of $\mathbf{V}_v$ map from this reduced $\mathbf{d}_v$ to an approximation to $\hat{\mathbf{c}}_v$.

If, additionally, these columns of $\mathbf{V}_v$ are normalised by the standard deviation along the corresponding principal component direction, then the mapping will move a reduced representation $\mathbf{d}_v$ distributed as an isotropic zero-centred unit Gaussian into the best multivariate Gaussian approximation to the observed distribution of the combined coefficient vectors including the relationships between coefficients (and therefore) locations of each structure. These standard deviations may be calculated from the singular values that appear down the diagonal of $\mathbf{S}_v$. Using s_i as the i^{th} singular value, the standard deviation of the i^{th} component of $\mathbf{d}_v$ is:

$$\sigma_i = \sqrt{\frac{s_i^2}{N-1}} \tag{8.12}$$

If $\mathbf{M}_v \in \mathbb{R}^{B_v \times D}$ is the matrix formed by taking the first D columns of $\mathbf{V}_v$ and dividing each column by the corresponding standard deviation, then the overall transformation from $\mathbf{M}_v \mathbf{d}_v$ to $\hat{\mathbf{c_v}}$ is given by:

$$\hat{\mathbf{c_v}} = \mathbf{M}_v \mathbf{d}_v + \overline{\mathbf{c}}_v \tag{8.13}$$

Therefore, in order to create a reduced state representation of the structure positions, $\mathbf{M}_v$ to and $\overline{\mathbf{c}_v}$ are found as above for each of the three views. An example of a PCA decomposition produced in this way is shown in Figure 8.3.

In addition to the position of the structures, there is also a binary variable, $g_{a,t} \in \{0, 1\}$, describing whether the structure is *visible* or *hidden*. A value of 1 (hidden) models the possibility that the structure is not visible in the image due to the probe position or due to imaging artefacts obscuring its position. These values for all structures are gathered into the binary vector $\mathbf{g}_t$.

The `PCAStructures` model uses an additional partition with the state variables $\mathbf{d}_{v,t}$ and for $v \in \{4C, LVOT, 3V\}$ and $\mathbf{g}_t$. The overall state tuple then becomes:

$$\mathbf{s}_t = \left(h_t, v_t, \mathbf{x}_t, \theta_t, \phi_t, \dot{\phi}_t, \mathbf{d}_{1,t}, \mathbf{d}_{2,t}, \mathbf{d}_{3,t}, \mathbf{g}_t\right) \tag{8.14}$$

To be used within the particle filtering framework (Chapter 6), a prediction potential must be defined to model the evolution of the reduced state vectors and the visibility variables over time. These are described in §8.5.2 and §8.5.3.

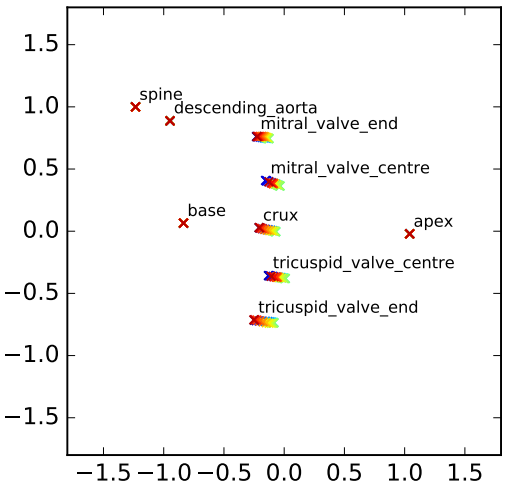
8.5.2 Structure Visibility Prediction Potential, $\psi_{\mathbf{g}}(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

The visibility variable for each structure is treated independently and in the same manner as the the visibility variable for the heart h_t as described in §6.5.1. That is, the hidden variable moves from hidden to visible according to a probability matrix whose values are chosen to give rise to a given time equilibrium fraction $\lambda^*_{\text{struct}}$ and time constant τ_{struct}. In general these two parameters may be different from λ^* and τ for the heart visibility.

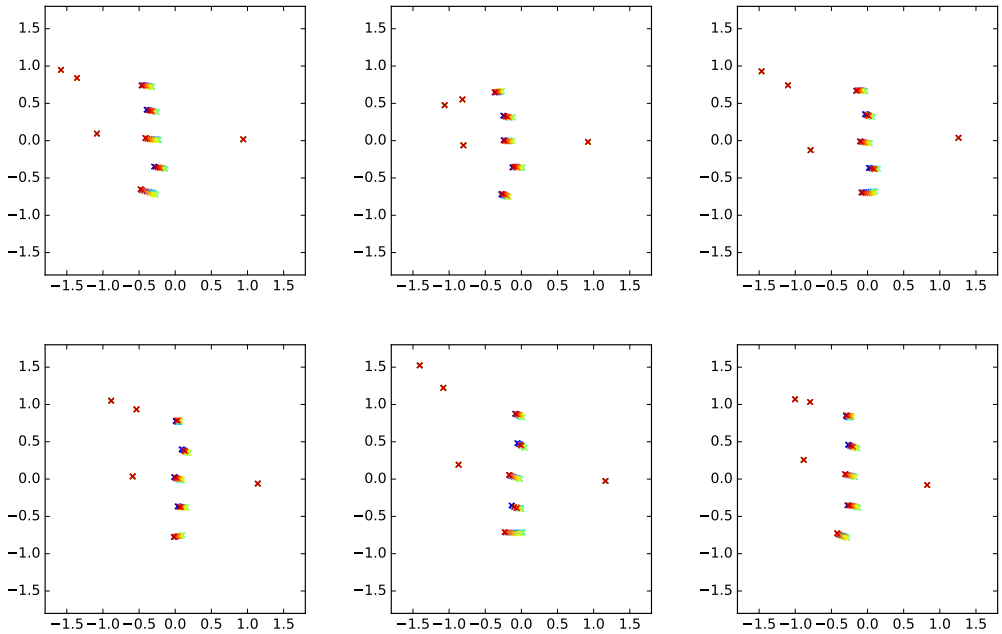
Furthermore, it is possible that the estimated location of the structure may move outside of the area where features can be evaluated, (i.e. within $R_{\text{struct-train}}$ of the edge of the ultrasound fan area). In this case, the structures are automatically considered to be hidden. Furthermore, the mitral valve and tricuspid valve are always considered hidden during diastole ($\pi < \phi_t < 2\pi$).

8.5.3 Structure Position Prediction Potential, $\psi_{\mathbf{d}}(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

The prediction potential for the reduced state vector $\mathbf{d}_v \in \mathbb{R}^D$ must be easy to sample from and represent realistic changes in the positions of the structures. Furthermore, it is important that after a number of such transition processes have



(a) Mean



(b) 1st principal axis **(c)** 2nd principal axis **(d)** 3rd principal axis

Figure 8.3: Example of a PCA decomposition of structure locations for structures in the 4C view (refer to Figure 8.1 for the definition of the structures used). (a) Shows the structure positions reconstructed from the mean state vector, while (b), (c) and (d) show the structure positions reconstructed from the state vectors 3 standard deviations away from the mean in each direction along the 1st, 2nd, and 3rd principal directions respectively. 20 points around the cardiac cycle are shown with each point represented represented by a different colour (*red-green* systole, *green-blue* diastole), and the plot axes show distance from the heart centre normalised by heart radius.

been applied, the resulting value for the reduced state vector continues to be a reasonable value. This requires that the limiting distribution of the resulting Markov chain is the same as the prior distribution of the state vector, in a similar way to the Markov chain for particle visibility described in §6.5.1.

This is achieved using a simple linear stochastic update model, in which the next state value $\mathbf{d}_{v,t}$ is formed from scaled version of the current state vector and additive Gaussian noise. In the general form this may be written:

$$\mathbf{d}_{v,t+1} = \mathbf{A}\mathbf{d}_{v,t} + \mathbf{G}\mathbf{n}_t \tag{8.15}$$

where $\mathbf{A} \in \mathbb{R}^{D\times D}$ is an update matrix, $\mathbf{G} \in \mathbb{R}^{D\times D}$ is a noise scaling matrix, and $\mathbf{n}_t \in \mathbb{R}^D$ is a Gaussian noise term with zero mean and covariance $\mathbf{Q}$:

$$\mathbf{n}_t \sim \mathcal{N}_D\left(\mathbf{n} \mid \mathbf{0}_D, \mathbf{Q}\right) \tag{8.16}$$

Note that the states for the three different views are assumed to evolve independently in order to reduce complexity.

The limiting distribution (the distribution to which $\mathbf{d}_{v,t}$ converges as $t \to \infty$) of such a process is a Gaussian distribution with zero mean and variance $\mathbf{\Sigma}_\infty$ which is found as the solution to the following *discrete time Lyapunov equation* [181]

$$\mathbf{\Sigma}_\infty = \mathbf{A}\mathbf{\Sigma}_\infty\mathbf{A}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T \tag{8.17}$$

Recall from §8.5.1 that the desired limiting distribution is the unit normal distributed, i.e. $\mathbf{\Sigma}_\infty = \mathbf{I}$. Therefore $\mathbf{A}$, $\mathbf{G}$ and $\mathbf{Q}$ must be chosen to give this limiting distribution and realistic transition behaviour. This can be achieved by setting $\mathbf{G} = \mathbf{I}$, $\mathbf{A} = \alpha\mathbf{I}$ where $\alpha \in \mathbb{R}$ is a scalar constant, and $\mathbf{Q} = q^2\mathbf{I}$, where $q \in \mathbb{R}$ is another scalar constant.

Then Equation 8.17 reduces to a simple scalar equation:

$$1 = \alpha^2 + q^2 \tag{8.18}$$

Consequently, if the value of q, the standard deviation of the random perturbation to $\mathbf{d}_{v,t}$ between two frames, then, assuming that $q \in [0, 1]$, the parameter α must be chosen as $\alpha = \sqrt{1 - q^2}$ in order to give the desired equilibrium distribution. In other words, in order to add a small random perturbation to $\mathbf{d}_{v,t}$, it must first be scaled by α, which will have the effect of moving it slightly closer to the origin (the mean of the distribution).

The prediction potential is therefore given by

$$\psi(\mathbf{d}_{v,t} \mid \mathbf{d}_{v,t-1}) = \mathcal{N}_D \left(\mathbf{d}_{v,t} \mid \left(\sqrt{1 - q^2} \right) \mathbf{d}_{v,t-1} \, , \, q\mathbf{I} \right) \tag{8.19}$$

Note that the state vectors for all three views are updated regardless of the value of the view state variable v_t.

8.5.4 Initialisation of Particles

Before the first frame of the video is processed, the visibility and reduced state vector for each particle are independently initialised by sampling from the following distributions:

- The visibility variables, g_a are sampled from a discrete distribution with a probability $\lambda^*_{\text{struct}}$ of being hidden.
- The reduced state vectors $\mathbf{d}_v$ are sampled from a unit multivariate Gaussian distribution.

8.5.5 Observation Potential, $\omega_D(\mathbf{s}_t, \mathbf{z}_t)$

The observation potential for the structures partition reflects the compatibility of the the current state estimate $\mathbf{d}_{v,t}$ with the image information. The particles are reweighted only on the basis of the reduced state vector $\mathbf{d}_{v,t}$ for the class that is currently being observed, i.e. for $v = v_t$, and the state values for the other two views are not considered.

The first step in this process is to find the current positions of the structures from the reduced state. This is performed by expanding to the full combined coefficient

vector $\hat{\mathbf{c}}_{v,t}$ using Equation 8.13, and then unpacking these coefficients into the Fourier coefficient vectors, $\mathbf{c}_{a,1}$ and $\mathbf{c}_{a,2}$, for each structure a by applying the reverse of the procedure chosen for packing (§8.5.1). The phase vector $\boldsymbol{\varphi}_t$ is then constructed from the particle's current estimate of the cardiac phase variable ϕ_t (Equation 8.2), and Equation 8.3 is applied to give the relative structure positions $\mathbf{p}_{a,t}$. These can then be used to calculate the absolute image positions, $\mathbf{q}_{a,t}$ with Equation 8.1.

Using the absolute image locations, the score Ω_a for a given structure a at the given image location may be evaluated using the relevant random forest model. This varies depending on which type of features (RIFs or rectangular features) is being used:

- When using RIFs, the score Ω_a is straightforwardly found as the likelihood of class a occuring at image location $\mathbf{q}_{a,t}$ according to the `RIFStructures` forest model.

- When using rectangular features, the fact that the orientation is discretised must be taken into account. The scores from two of the `RECStructures` forest models are averaged based on the current heart orientation θ_t. θ_t is rounded to the two closest training orientations $\theta_{\text{train},n}$, giving $\theta_{\text{train,lower}}$ below θ_t and $\theta_{\text{train,upper}}$ above θ_t. Using the two forest models at the chosen training orientations, the likelihood of class a at image location $\mathbf{q}_{a,t}$ is calculated, giving one score from each forest (Ω_{upper} and Ω_{lower}). The weighted average of these two scores is used to provide the final score Ω_a, where the weights are determined according to the distance of θ_t from the training orientations such that the score is linearly interpolated between the two.

$$\Omega_a = \frac{\theta_{\text{train,upper}} - \theta_t}{\theta_{\text{train,upper}} - \theta_{\text{train,lower}}} \cdot \Omega_{\text{lower}} + \frac{\theta_t - \theta_{\text{train,lower}}}{\theta_{\text{train,upper}} - \theta_{\text{train,lower}}} \cdot \Omega_{\text{upper}} \quad (8.20)$$

In both cases, structures that are hidden are given a small fixed score Ω_{hidden} in a similar way to when the whole heart is hidden, as described in §6.4.4.

Once the scores have been gathered from all structures in the current view v_t, they are combined into a single score for the particle by taking their mean, as follows:

$$\omega_D(\mathbf{s}_t, \mathbf{z}_t) = \frac{1}{|\mathcal{S}_v|} \sum_{a \in \mathcal{S}_v} \Omega_a \tag{8.21}$$

8.6 The PartitionedStructures Model

This architecture uses one additional filtering partition for each of the structures of interest. The partitions are arranged into three 'branches' (Figure 8.2b), one for each of the three views. This means that the resampling steps within a given branch change the set of particles within the views but do not change the distribution of particles between the three views, which is decided by the earlier position and view partition.

8.6.1 Definition of the State

Each partition is identified by the index, a, of the corresponding structure. The state vector for partition a is described by the Fourier coefficients for the relevant structure, contained in the vectors $\mathbf{c}_{a,1,t} \in \mathbb{R}^{2b_a+1}$ and $\mathbf{c}_{a,2,t} \in \mathbb{R}^{2b_a+1}$ (§8.4). These two vectors are combined into a single vector $\tilde{\mathbf{c}}_{a,t}$, where the order in which the elements are placed does not matter as long as a consistent rule is used. For notational convenience, let $d_a = 2(2b_a + 1)$ be the dimension of this state vector.

Just like in the PCAStructures architecture, there is also a binary variable, $g_{a,t}$ describing the each structure's visibility.

The full state tuple is therefore given by:

$$\mathbf{s}_t = \left(h_t, v_t, \mathbf{x}_t, \theta_t, \phi_t, \dot{\phi}_t, \{(\tilde{\mathbf{c}}_{a,t}, g_{a,t})\}_{a \in \mathbb{Z}_{1,S}}\right) \tag{8.22}$$

Note that a given particle only participates in the resampling stage after the partition for structure a if the heart visibility variable $h_t = 1$ and structure a is present in the current view v_t (see Figure 8.2).

8.6.2 Structure Visibility Prediction Potential, $\psi_{g_a}(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

The visibility prediction potential for each structure's visibility variables operates in exactly the same way as that for the `PCAStructures` model, as described is in §8.5.2.

8.6.3 Structure Position Prediction Potential, $\psi_{\tilde{\mathbf{c}}_a}(\mathbf{s}_t \mid \mathbf{s}_{t-1})$

The prediction potential is similar to that described in §8.5.3 for the `PCAStructures` model. At training time, a mean vector $\tilde{\boldsymbol{\mu}}_a \in \mathbb{R}^{d_a}$ and covariance matrix $\tilde{\boldsymbol{\Sigma}}_a \in \mathbb{R}^{d_a \times d_a}$ are calculated for the coefficient vector $\tilde{\mathbf{c}}_{a,t}$, assuming its distribution to be a multivariate Gaussian.

As in §8.5.3, the prediction potential is assumed to be a linear transition followed by additive Gaussian noise on the centred coefficient vector, i.e. of the general form

$$(\tilde{\mathbf{c}}_{a,t+1} - \tilde{\boldsymbol{\mu}}_a) = \mathbf{A}\,(\tilde{\mathbf{c}}_{a,t} - \tilde{\boldsymbol{\mu}}_a) + \mathbf{G}\mathbf{n}_t \tag{8.23}$$

Again, it is desirable to ensure that the limiting distribution of the Markov chain created by the prediction potential process is the same as the prior distribution. The limiting distribution for the centred coefficient vector has covariance matrix $\boldsymbol{\Sigma}_\infty \in \mathbb{R}^{d_a \times d_a}$, which is given as the solution of the discrete time Lyapunov equation (Equation 8.17). Therefore, to design a prediction potential whose limiting distribution has mean $\tilde{\boldsymbol{\mu}}_a$ and covariance matrix $\tilde{\boldsymbol{\Sigma}}_a$, the update matrix is set $\mathbf{A} = \alpha\mathbf{I}$ where $\alpha \in [0, 1]$, the covariance of the noise vector $\mathbf{n}_t \in \mathbb{R}^{d_a}$ is set to be $\mathbf{Q} = \tilde{\boldsymbol{\Sigma}}_a \in \mathbb{R}^{d_a \times d_a}$, and the noise scaling is set to be $\mathbf{G} = \gamma\mathbf{I}$, where $\gamma \in [0, 1]$ and $1 = \alpha^2 + \gamma^2$.

Note that the prediction potential described above predicts each structure's position relative to the position of the heart centre, but conditionally independent of the position of the other structures. However, there is no reason that this has to be the case. In fact, the prediction potential for a certain structure could in principle be conditioned upon the position of one or more structures from previous partitions in order to take into account the spatial relationships between the structures. A simple example of this would be to define the position of each structure relative to the position of a single previous structure, rather than relative to the heart

centre. This was not pursued in this thesis due to time limitations, and because preliminary experiments suggested the gain would not be significant. There are also a number of practical problems with the approach, including what to do when the position of one structure considered to be conditional on the position of another structure, but the second structure is 'hidden'.

8.6.4 Initialisation of Particles

Before the first frame of the video is processed, the visibility variable and reduced state vector for each particle and each structure are independently initialised by sampling from the following distributions:

- The visibility variables, g_a are sampled from a discrete distribution with a probability $\lambda^*_{\text{struct}}$ of being hidden (as with the `PCAStructures` model).
- The coefficient vectors $\tilde{\mathbf{c}}_a$ are sampled from the relevant multivariate Gaussian distributions with mean $\tilde{\boldsymbol{\mu}}_a$ and covariance $\tilde{\boldsymbol{\Sigma}}_a$.

8.6.5 Observation Potential, $\omega_{E_a}(\mathbf{s}_t, \mathbf{z}_t)$

Before applying the observation potential, it is first necessary to reconstruct the position of the relevant structure from the coefficient vector $\tilde{\mathbf{c}}_{a,t}$. First $\tilde{\mathbf{c}}_{a,t}$ is split into the vectors for the x- and y-components individually, by reversing the packing procedure applied in §8.6.1 giving $\mathbf{c}_{a,1,t}$ and $\mathbf{c}_{a,2,t}$. The phase vector $\boldsymbol{\varphi}_t$ is then constructed from the particle's current estimate of the cardiac phase variable ϕ_t (Equation 8.2), and Equation 8.3 is applied to give the relative structure positions $\mathbf{p}_{a,t}$. These can then be used to calculate the absolute image positions, $\mathbf{q}_{a,t}$ with Equation 8.1.

The observation potential then finds a score, Ω_a, for the likelihood of structure a appearing at location $\mathbf{q}_{a,t}$ following the same procedure described for the `PCAStructures` model in §8.5.5. However, in the `PartitionedStructures` architecture, each structure is reweighted individually, and the particle set is resampled before the next structure is considered. Therefore, there is no need to

combine the scores from the different structures and they are applied directly to the particle weights:

$$\omega_{E_a}(\mathbf{s}_t, \mathbf{z}_t) = \Omega_a \tag{8.24}$$

As in the observation potential for the `PCAStructures` model (§8.5.5), hidden particles are given a small fixed score Ω_{hidden}.

8.7 Extracting State Estimates from a Particle Set

A point estimate for the locations of the structures can be extracted from the particle set using the mean-shift algorithm, as described in §6.7. This can be performed for each structure independently. Firstly, the particles vote for the visibility of the structure by simple majority of the g_a variable. If the particle is visible, mean-shift is performed on the locations of the particles, $\mathbf{q}_{a,t}$, directly, and not the underlying coefficient vectors that form the state. The width of the mean shift window is $K_{\mathbf{q}}$ and the tolerance is $\epsilon_{\mathbf{q}}$.

9

Experimental Validation of Structure Tracking

Contents

This chapter presents an experimental evaluation of the filtering architectures introduced in Chapter 8 for tracking cardiac structures of interest through video streams. At the time of writing, some of the results in this chapter have been included in a conference paper that is under review for MICCAI 2017.

9.1 Experimental Dataset

The dataset described in §1.5 was used for all experiments in this chapter. In addition to the global heart variables, manual annotations for the position, $\mathbf{q}_{a,t}$, and visibility, $g_{a,t}$, of each of the structures in Figure 8.1 were used to train and test the localisation of structures using the cross-validation approach described in §5.2.

Structure	Fourier Order, b_a
Crux	3
Apex	0
Base	0
Mitral Valve End	3
Mitral Valve Centre	3
Tricuspid Valve End	3
Tricuspid Valve Centre	3
Aortic Valve	3
Aorta (3V View)	1
Pulmonary Valve	1
Trachea	0
SVC	1
Spine	0
Descending Aorta	0

Table 9.1: Fourier model orders for the structures used in experiments. A value of 3 was used for structures exhibiting significant, complex motion over the cardiac cycle, such as valves. A value of 1 was used for structures with small displacements over the cardiac cycle, such as the centres of vessels. A value of 0 was used for structures that remain stationary over the cycle, such as the spine and trachea.

9.2 Fitting Prediction Potential Models

The first step in fitting the prediction potentials for either of the `PCAStructures` or `PartitionedStructures` architectures is to fit Fourier models $(\mathbf{c}_{a,1}, \mathbf{c}_{a,2})$ to short sequences of video (see §8.4). This was done by automatically searching the videos in the training set for sequences that, according to the ground truth manual annotations, are one cardiac cycle in length, contain only a single view of the heart, and contain only frames where the heart is 'present'. All such sequences that were found in the training set videos were used to fit one Fourier model $(\mathbf{c}_{a,1}, \mathbf{c}_{a,2})$ for every structure, a, in that view. Table 9.1 shows the values used for the Fourier model orders, b_a, for each structure a. These were chosen manually to capture the degree of complexity of the periodic motion of each structure.

Once these Fourier models were fitted, each resulting Fourier model was used as one example to fit the `PCAStructures` and `PartitionedStructures` models.

Certain parameters were held at fixed values during the fitting and testing stages. These are shown in Table 9.2. In particular, the values for q and γ were

Parameter	Value	Description
λ	1.0 pixels^{-2}	Prior precision (inverse variance) for the Fourier coefficient values (see Equations 8.7 and 8.8).
D	5	The number of PCA components in the `PCAStructures` model.
q	0.3	Noise standard deviation (as a fraction of prior standard deviation) for the `PCAStructures` prediction potential.
γ	0.3	Noise standard deviation (as a fraction of prior standard deviation) for the `PartitionedStructures` prediction potential.
$K_{\mathbf{q}}$	5.0 pixels	Kernel size for structure position mean shift.
$\epsilon_{\mathbf{q}}$	1.0 pixels	Tolerance for the structure position mean shift.

Table 9.2: List of filter parameter values for the prediction and mean shift algorithms of the structures models.

chosen empirically to give a reasonable trade off between the prediction potentials being too 'loose' (giving poor tracking performance) or 'tight' (giving poor initial localisation and recovery from error). Also the number of PCA components, D, was chosen to give a reasonable trade-off between the amount of variation captured by the model, and the ability of the filter to track in a high dimensional space. The values listed in Table 7.2 were re-used for all structures experiments.

9.3 Fitting Observation Potential Models

The `RECStructures` and `RIFStructures` forest models are trained using a similar procedure as the `RECDetection` and `RIFFilter` models (Chapter 5) with the parameters in Table 5.1. 5000 positive samples were chosen from the training set videos for each structure, and for each positive sample a random background sample with a random orientation was chosen. For the `RECStructures` model, a random offset angle in the range $[-\frac{2\pi}{2N_o}, \frac{2\pi}{2N_o}]$ was then applied to each patch, such that forest model learns to recognise all structures within its orientation range.

For all experiments in this chapter, the `freq` calculation method was used for calculating RIFs. All RIF-based models used the `grad422_motion422_extra` fea-

ture set with the `RIFStructures` model using only the two innermost radial profiles ($J_{\text{structs}} = 2$). All rectangular-feature based models used the `rec_gradmotion_10` feature set with the `RECStructures` model using a patch size of $R_{\text{struct-train}} = R_{\text{train}}/2 = 15$ pixels, such that both the `RIFStructures` and `RECStructures` use patches of the same size.

9.4 Evaluation Metrics

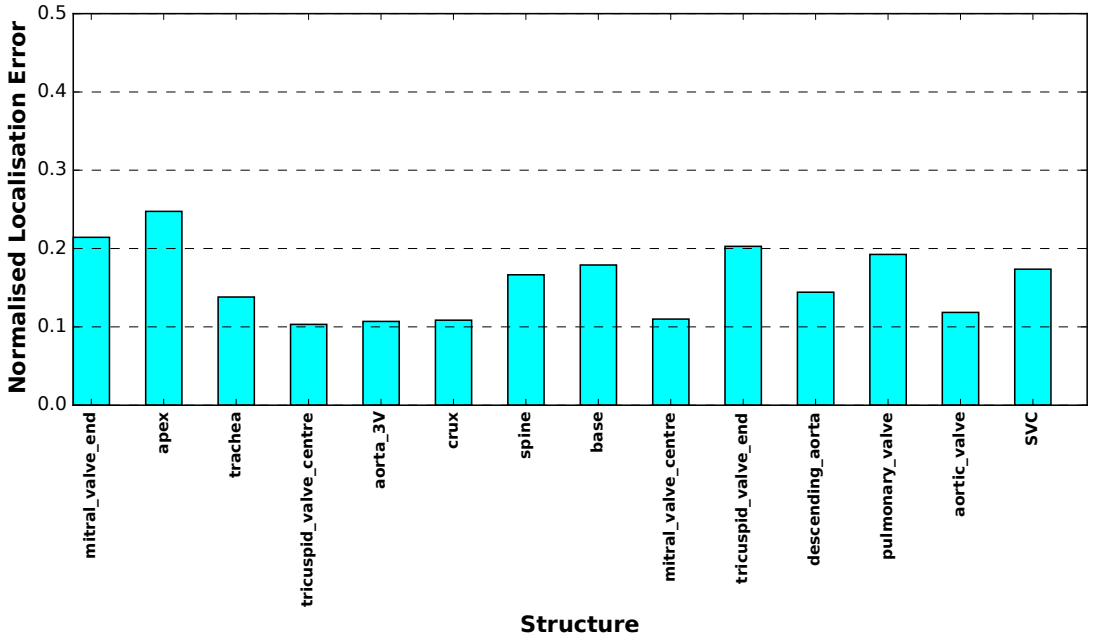
The following new evaluation metric was used for the evaluating the structure localisation:

Normalised Localisation Error This is defined for a given structure as the normalised Euclidean distance between the detected location for the structure and the ground truth location for the structure, where normalisation is by the ground truth heart radius. This is averaged over all frames in which the heart was correctly detected, as defined in §5.1, and in which the relevant structure is visible according to both the ground truth and detection.

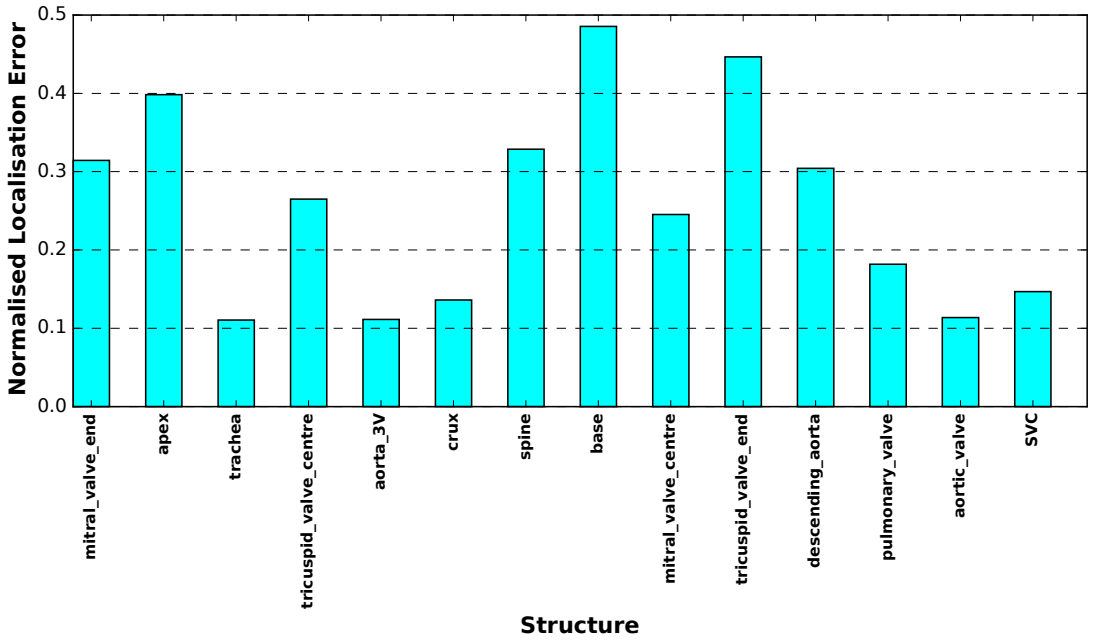
9.5 Results

Localisation error for the different structures with the `RIFFilter` and `RECFilter` architectures with each of the `PCAStructures` and `PartitionedStructures` extensions (giving four combinations in total) are shown in Figures 9.1 and 9.2. It is clear that for both the `PCAStructures` and `PartitionedStructures` extensions, the use of the `RECFilter` results in considerably greater localisation error for the structures. This is primarily because of the lower orientation accuracy of the `RECFilter` architecture (§7.3.5), which has in turn reduces the structure localisation accuracy because the structure positions are represented relative to the heart position and orientation.

There is quite a wide variation in the localisation accuracy among the different structures. Typically those structures whose position in the image is well defined,

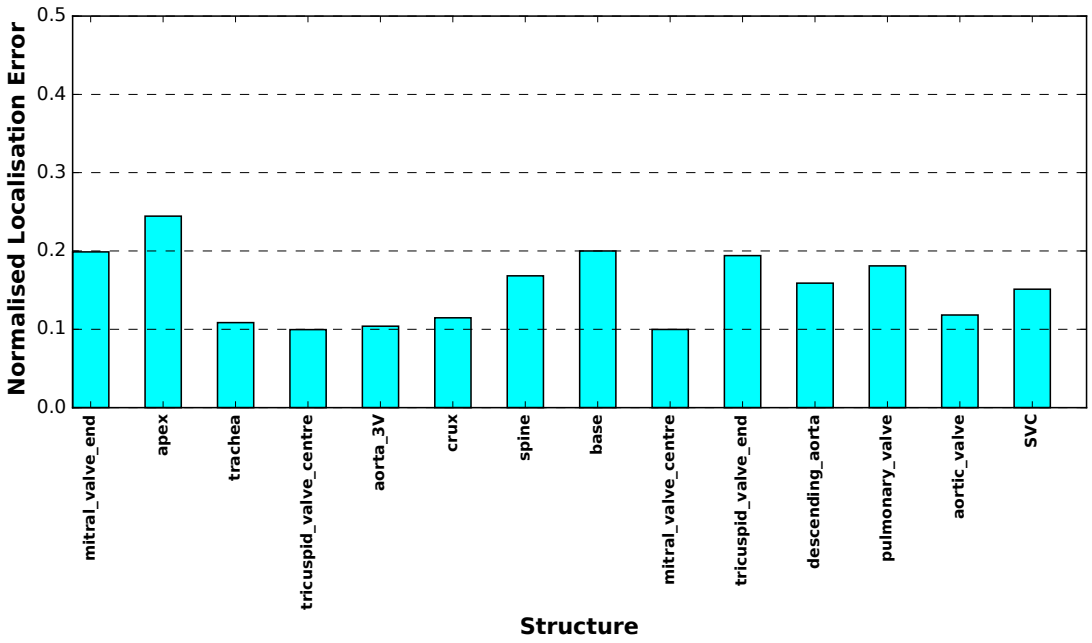


(a) `RIFFilter` using the `grad422motion422_extra` feature set with $J_{structs} = 2$ and the `PCAStructures` extension.

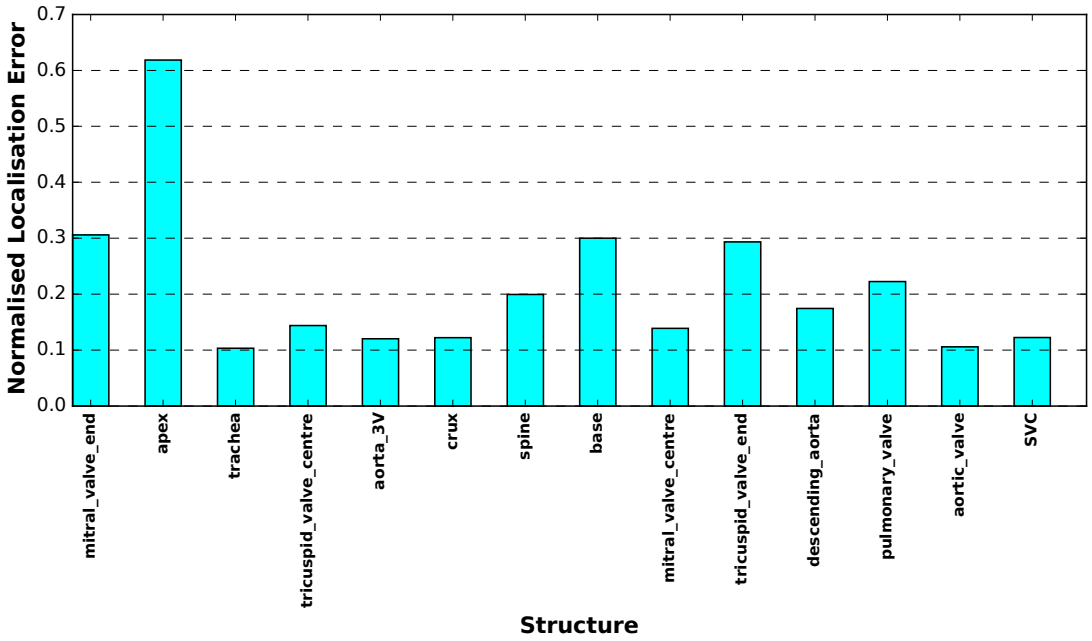


(b) `RECFilter` using the `rec_gradmotion_10` feature set and the `PCAStructures` extension.

Figure 9.1: Normalised localisation error (§9.4) for models with the `PCAStructures` extension. The visibility filtering parameters for the heart were set to $\lambda^* = 0.4$, $\tau = 2.0$, $w_{\text{hidden}} = 0.3$, and for each structure were set to $\lambda^*_{\text{struct}} = 0.4$, $\tau_{\text{struct}} = 2.0$, $w_{\text{hidden}} = 0.05$. The composition of the forest models was as follows: detection forests $N_{\text{trees}} = 16$ and $D_{\max} = 10$, phase regression forests $N_{\text{trees}} = 32$ and $D_{\max} = 10$, structures forests $N_{\text{trees}} = 16$ and $D_{\max} = 10$. The particle set contained 1000 particles.



(a) `RIFFilter` using the `grad422motion422_extra` feature set with $J_{structs} = 2$ and the `PartitionedStructures` extension.



(b) `RECFilter` using the `rec_gradmotion_10` feature set and the `PartitionedStructures` extension. Note the different scale due to the large error for the apex.

Figure 9.2: Normalised localisation error (§9.4) for models with the `PartitionedStructures` extension. The visibility filtering parameters for the heart were set to $\lambda^* = 0.4$, $\tau = 2.0$, $w_{\text{hidden}} = 0.3$, and for each structure were set to $\lambda^*_{\text{struct}} = 0.4$, $\tau_{\text{struct}} = 2.0$, $w_{\text{hidden}} = 0.05$. Other parameters were as in Figure9.1.

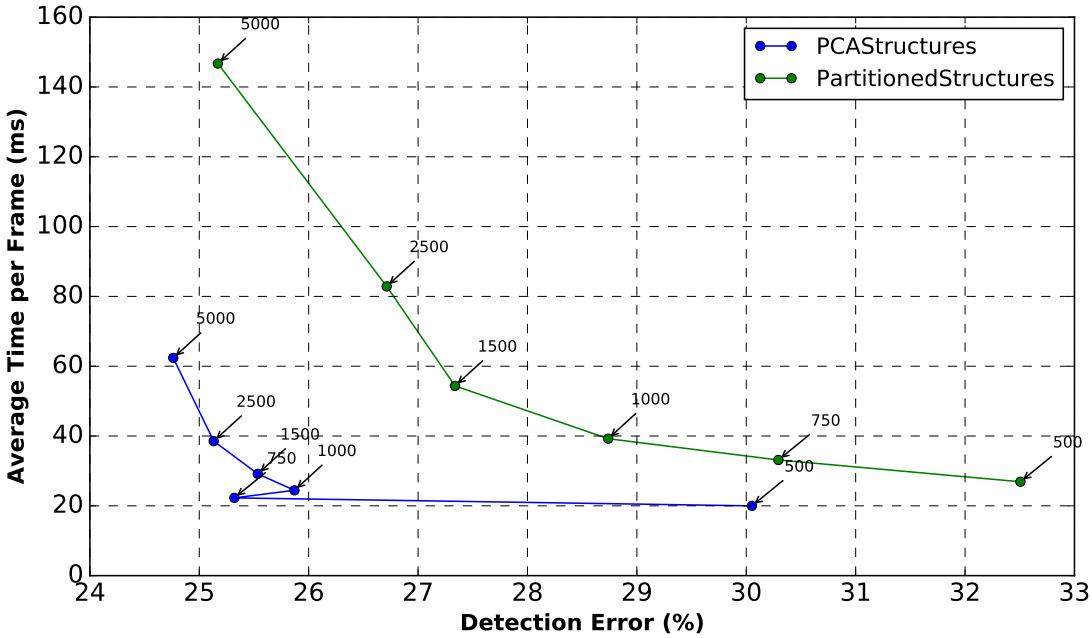


Figure 9.3: Detection error and total calculation time with different numbers of particles for the `RIFFilter` architecture with both the `PCAStructures` and `PartitionedStructures` extensions. The `grad422motion422_extra` feature set was used in both cases, with $J_{structs} = 2$. The visibility filtering parameters for the heart were set to $\lambda^* = 0.4$, $\tau = 2.0$, $w_{\text{hidden}} = 0.3$, and for each structure were set to $\lambda^*_{\text{struct}} = 0.4$, $\tau_{\text{struct}} = 2.0$, $w_{\text{hidden}} = 0.05$. The composition of the forest models was as follows: detection forests $N_{\text{trees}} = 16$ and $D_{\max} = 10$, phase regression forests $N_{\text{trees}} = 32$ and $D_{\max} = 10$, structures forests $N_{\text{trees}} = 16$ and $D_{\max} = 10$. Annotations show the number of particles in the set.

such as the crux, valves (mitral, tricupid, and aortic), and vessels (ascending and descending aorta and SVC), are localised to a high degree of accuracy.

By contrast, those whose precise position is more ambiguous, such as the base, apex, spine, and the ends of the atrio-ventricular valves are localised poorly. In these cases, it is difficult to choose a single point on the image as the locations of the structure. Results for the pulmonary valve and trachea are also relatively poor. This is a result of the fact that the visibility of these structures depends strongly on the precise imaging plane. These structures are however also typically less important for diagnostic purposes. Due to the very laborious nature of the annotation task no estimate of inter- or intra-observer was performed, but it is likely that it would display a similar pattern of well and poorly localised structures.

A more detailed comparison of the `PCAStructures` and `PartitionedStructures`

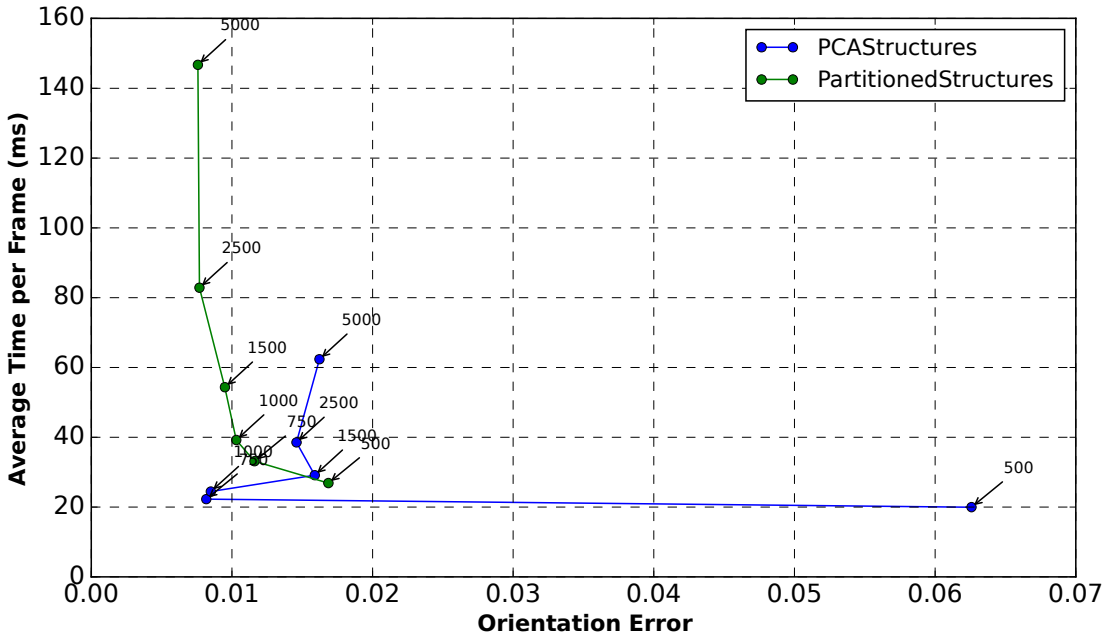


Figure 9.4: Orientation error and total calculation time for the experiments in Figure 9.3.

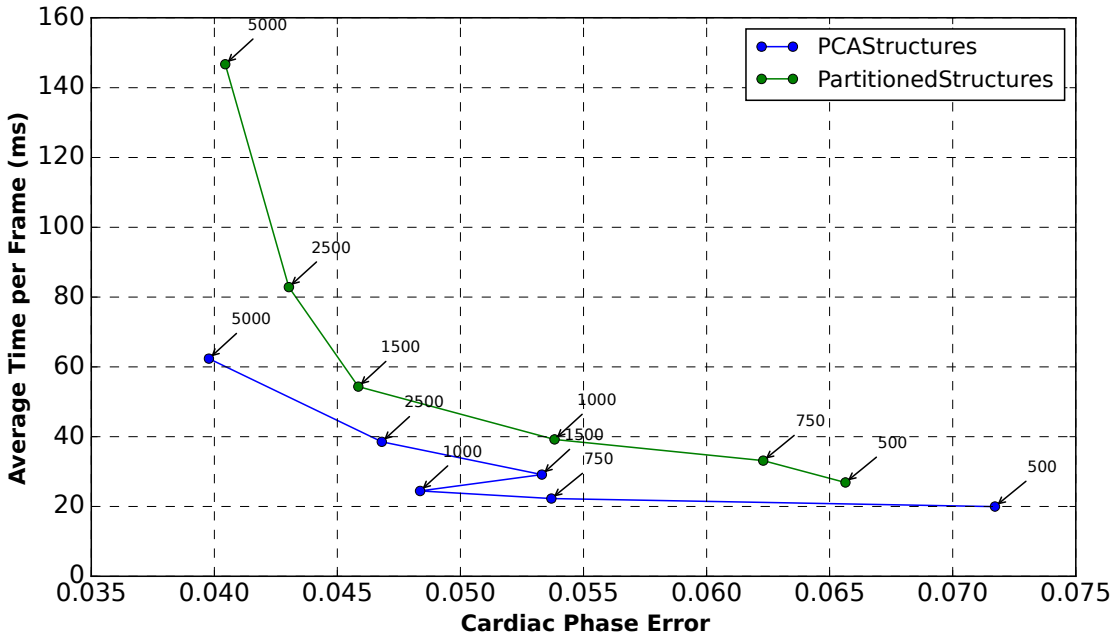
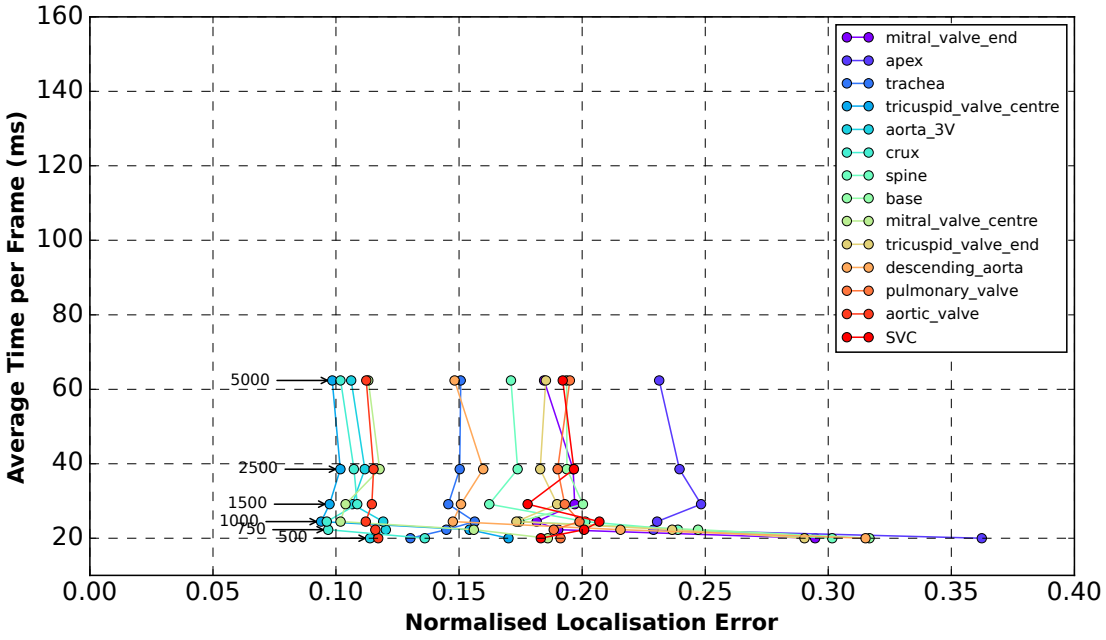
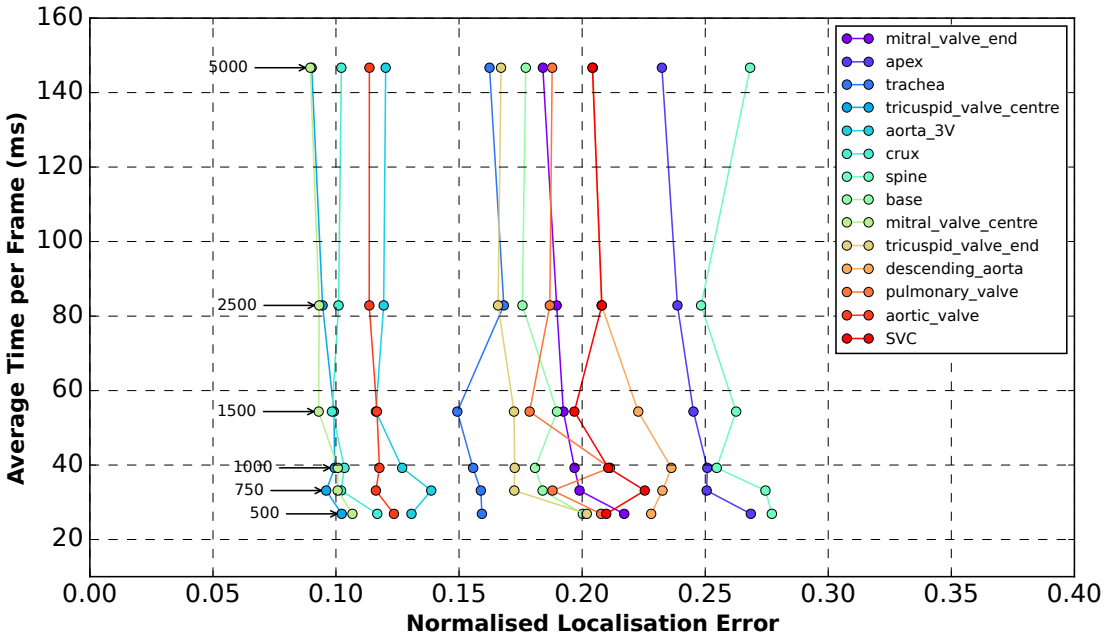


Figure 9.5: Cardiac phase error and total calculation time for the experiments in Figure 9.3.



(a) The `PCAStructures` model.



(b) The `PartitionedStructures` model.

Figure 9.6: Structure localisation error and total calculation time for the experiments in Figure 9.3.

models is shown in Figures 9.3, 9.4, 9.5 and 9.6, which respectively show the heart detection error, orientation error, cardiac phase error and structure localisation for a `RIFFIlter` architecture error as the number of particles in the particle set is varied. This demonstrates the effect that adding the structure localisation stage has on the entire framework.

These plots show that the two methods are broadly similar in terms of the structure localisation performance. This suggests that the dimensionality reduction applied by the `PCAStructures` model is effective and does not reduce the representational power of the state space to the degree that it begins to effect localisation performance. However it also suggests that the spatial correlations between structures captured by the PCA do not significantly improve results over the `PartitionedStructures` method in which each structure's location is conditioned on the heart centre alone. One important difference is that the `PartitionedStructures` model is significantly slower due to the larger number of prediction potential and resampling operations that need to be applied (the number of calculations for the observation potentials should be identical in both cases).

Comparing Figures 9.3, 9.4, 9.5 with their counterparts in Chapter 7 (Figures 7.2, 7.8 and, 7.9) shows that introducing the `PartitionedStructures` extension onto the `RIFFilter` architecture slightly *reduces* the heart detection accuracy of the filter, unless a large number of particles (around 5000 or more) is used. By contrast the `PCAStructures` model does not suffer from this issue. This shows that the partitioning of the state space has not completely alleviated the problems due to its high dimension (although it is reasonable to assume that the performance is significantly better than without partitioning). This particularly manifests itself at the start of the video, during the initial global localisation stage. At this stage, many of the particles will not be particularly close to the true heart centre, and as such the location of their structures will be quite far from the true locations, even if the output from the `RIFDetection` model is quite high. Applying several update and resampling operations to the particles at this stage will exaggerate

small random variations in the `RIFStructures` observation potentials, which serves to confuse the heart localisation process.

The orientation and phase accuracies do not seem to have been significantly affected by the addition of the structures localisation extensions.

It was found that altering the composition of the structure detection forests did not significantly alter the performance of the framework. There are also only a very small number of frames in which structures were labelled as hidden (when the heart itself was not marked as hidden), and furthermore the annotations here were even more inconsistent than those for the heart visibility, consequently the analysis of true and false positive rates for the structures did not produce meaningful results.

9.6 Qualitative Evaluation

Figure 9.7 shows the structure localisation results in some example frames. In general, there is little difference between the results for the `PartitionedStructures` and `PCAStructures` models. The second row is a good example of how the 'base' structure is poorly localised by both models due to having few distinct image features to use to localise it, and similarly for the spine in the fourth row. The fifth row shows significant disagreement about the locations of the spine and descending aorta.

9.7 Conclusions

This chapter has shown that the structures extensions introduced in Chapter 8 can be used to localise a number of structures of the fetal heart to a reasonable degree of accuracy with no user initialisation at high frame rates in clinical videos containing multiple views. To the author's knowledge this has not been attempted before in the literature. This serves of an example of how such a model could be used to automatically localise structures of diagnostic importance when developing an algorithm for detection of CHD. Some structures are localised better than others, depending on how well-defined their location is based on the image features.

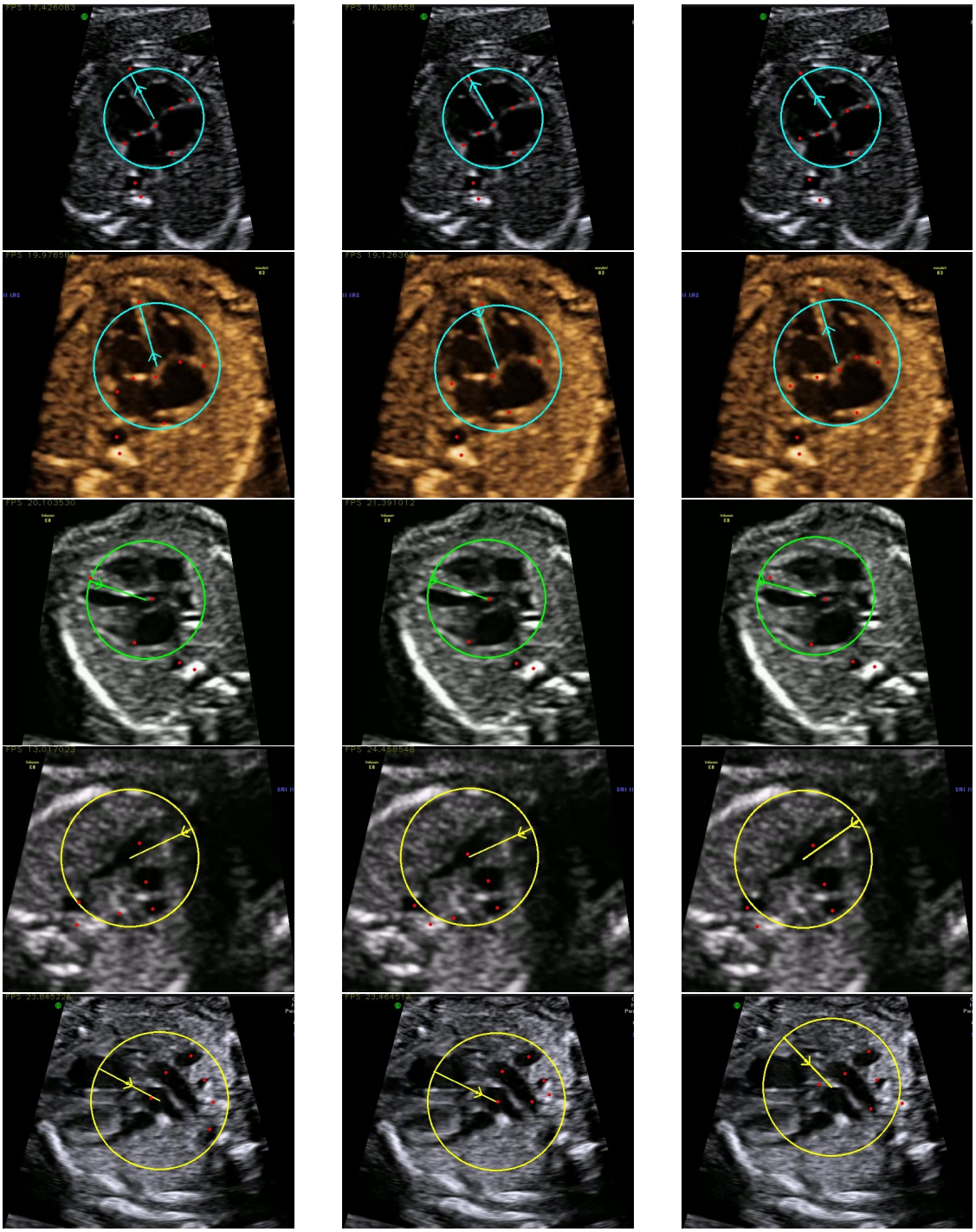


Figure 9.7: Example outputs from the structure localisation models. *Left* `PartitionedStructures` model, *Centre* `PCAStructures` model, *Right* manual ground truth. Frames drawn from the experiments shown in Figure 9.3 with 1000 particles. Red dots show locations of structures.

Many of the structures of diagnostic importance, such as the centres of the atrio-ventricular valves in the four-chamber view, and the vessels in the three vessels view, can be localised with average distance errors of approximately 0.1 times the radius of the heart.

The results demonstrate that the `PCAStructures` and `PartitionedStructures` architectures are both effective ways to deal with the problem of a filtering in a high dimensional state space. However, the `PCAStructures` model has a lower computational demand and does not interfere as strongly with the localisation and classification of the heart.

Again, it is difficult to define what would be a 'reasonable' acccuracy to expect from the structure localisation results, especially as there is no estimate of inter- and intra-observer variation to compare with in this case. The required accuracy would again depend on the intended use of the output. However, an error of $0.1\times$ the radius of the heart semms likely to be sufficiently accurate for the majority of purposes including guiding detectors to screen for particular anomalies, and several of the structures chosen can be localised to that level of accuracy by the presented algorithms.

10 Conclusions and Future Work

Contents

10.1 Summary of Contributions

This thesis has presented and evaluated a complete, fully automated framework for extracting estimates of key *state variables* (heart location, visibility, orientation, view, cardiac phase, and locations of cardiac structures) from fetal heart ultrasound videos, which is a task that has previously received very little attention. The core of the framework is a particle filtering method that captures the temporal dynamics of the variables and their interdependencies and makes a fully probabilistic estimate of the state variables in each frame. The probabilistic estimate is represented non-parametrically by a particle set, which is able to represent complex multi-modal distributions, and therefore has a high degree of robustness to the uncertainty and

ambiguity inherent in the interpretation of fetal ultrasound imagery.

The framework is defined by two sets of models: *prediction potentials* modelling relationships between values in neighbouring frames, and *observation potentials* modelling the relationship between state variables and the information in the images. This approach is very general and allows different learning methods and feature sets to be used for the observation potentials, different prediction potentials to be used and different independence assumptions to be made between the observation and predictions potentials applied to each of the state variables. A few specific instantiations of this general framework were designed and investigated using prediction potentials based on simple probability distributions and observation potentials based on random forest models.

Chapter 3 began the presentation of this framework with the observation models by discussing the use of rotation-invariant features (RIFs) for fetal ultrasound imagery. Such features have the advantage of naturally and efficiently dealing with the unknown orientation of the fetal anatomy relative to the probe, which is problematic for more traditional feature sets. Several novel contributions to the implementation of RIFs were made, most notably the use of frequency domain calculations, that allow the RIFs to be used at high frame rates in conjunction with the random forests algorithm. Experimental validation in Chapter 5 suggested that the RIF based random forest models performed at the level of, or better than, rectangular features for the detection task, due to dealing better with the unknown orientation.

Chapter 4 introduced a method for performing regression onto a circular output space in a principled way using the random forests algorithm. The resulting *circular regression forests* algorithm was shown in Chapter 5 to give reasonable estimates (with respect to ground truth annotations) for the challenging task of cardiac phase estimation within ultrasound fetal heart videos, particularly if used with motion features extracted from the image in addition to intensity gradient based features.

Chapter 6 showed how the random forests observation models from the previous chapters may be integrated into the particle filtering framework and combined

with prediction potentials that capture prior knowledge about the videos. Two different architectures were evaluated: one in which the invariance of the RIFs was exploited to *partition* the filter into three partitions that can operate sequentially, and a second using rectangular filters that uses two partitions. It was in Chapter 7 found that the use of the particle filtering was able to substantially improve upon the accuracy of the independent framewise estimates and still operate at speeds suitable for real-time use on modest hardware.

Furthermore, it was found that the architecture using RIFs outperformed the architecture based on rectangular features in terms of both accuracy and calculation speed. This is likely due to a combination of two factors: firstly the superior performance of the RIFs in the presence of unknown orientations, and secondly the finer partitioning of the state that the rotation-invariant observation model permits results in more efficient use of a finite particle set.

Chapter 8 introduced two methods to extend the framework for the task of predicting the locations of particular structures of interest represented by a single image location. The first method is based on dimensionality reduction of the combined locations of several structures, and tracking the resultant reduced representation. The second method uses an additional partition of the state space for each structure, thereby tracking each structure independently whilst making efficient use of the particle set in the very large resulting state space. Experimental results in Chapter 9 suggested that both methods provide an effective way to deal with the problem in a high dimensional state space, but that the PCA-based method has a lower computational demand.

Each part of the framework was validated on a dataset of ultrasound videos from a clinical setting with a manually annotated ground truth. The results were compared against estimates of inter- and intra-observer variation obtained from repeated sets of annotations. The very high level of disagreement between these sets of annotations, particularly regarding the visibility and view classification of the heart, reveals the inherent ambiguities in the task and sets a realistic upper bound for the performance that may be achieved with automatic methods.

In terms of detection error (including view classification), the error rate from the automatic method is comparable to the inter-observer variation at around 25% but above the intra-observer variation. The orientation error is slightly greater than the intra- and inter-observer variation with an average cosine error corresponding to about 11°. The cardiac phase error reaches the level of the human annotators, with an average cosine error corresponding to 0.06 cycles. The automatic method was found to be somewhat inferior to human annotators in terms of achieving a low false positive rate for a true positive rate. Tackling this problem is likely to require improving the way the data is annotated and used to train the observation potentials, as well as how it can be more meaningfully evaluated. Structure localisation errors were not compared to human annotators, but the best structures were localised to an average error of 0.1 times the heart radius.

There are a number of parameter decisions to be made when implementing the framework, many of which involve understanding some important performance trade-offs. One key trade off is between calculation speed and accuracy, broadly defined. There are a number of ways to improve accuracy at the expense of calculation speed, including choosing more expressive feature sets, increasing the number of trees in the forest models, and increasing the number of particles in the particle set.

A second important trade-off is between the competing aims of achieving a high true positive rate (detecting frames where the heart is present) and a low false positive rate (rejecting frames in which the heart is hidden). This can be achieved by altering the parameters of the heart visibility model and the *hidden weight parameter*.

A third important trade-off that is inherent in all particle filtering methods is choosing the prediction model parameters to achieve the competing aims of good tracking performance and good initial localisation performance and recovery from errors.

Many aspects of the presented framework are quite general and similar ideas may be useful in creating algorithms for other ultrasound (or natural) video analysis tasks that require the estimation of a number of interrelated variables through time, with strong prior knowledge and relatively weak image information.

Parameter	Value
Number of trees in the `RIFDetection` forest (N_{trees})	16
Number of levels in the `RIFDetection` forest ($D_{\max}$)	10
Number of trees in the `RIFPhase` forest (N_{trees})	32
Number of levels in the `RIFPhase` forest ($D_{\max}$)	10
Number of trees the `RIFStructures` forest (N_{trees})	16
Number of levels in the `RIFStructures` forest ($D_{\max}$)	10
Number of particles	1000
Image representations used for feature extraction	gradient, motion
Number of radial profiles in RIF extraction (J)	3
Number of rotation orders in RIF extraction (K)	3
Fourier histogram expansion in RIF extraction (M)	2
Calculation method for RIF extraction	`freq`
Coupling method for RIF extraction	`extra`
Hidden particle weight (w_{hidden})	0.3
Equilibrium fraction of hidden particles (λ^*)	0.4
Hidden particle time constant (τ)	2 frames

Table 10.1: Recommended set of parameters for a `RIFFilter` architecture.

In order to enable and encourage future work towards the goals of this thesis, and to support open and reproducible science, source code of the implementation used for all experiments and analysis in this thesis has been made publicly available (see Appendix §A.1). Unfortunately, the author is not at liberty to make the video dataset available.

10.2 Recommended Parameters

In previous chapters, various suggestions for suitable parameter values have been made, and there has been some discussion of the inherent trade-offs behind these choices. Overall, the recommended architecture is the `RIFFilter` architecture using rotation-invariant features, along with the `PCAStructures` extension if structure tracking is required. Table 10.1 provides a list of suggested values for the most important parameters for those wishing to compare the performance of this method against other methods.

10.3 Directions for Future Work

This section proposes four broad, overlapping directions for future work enabled by this thesis: working towards clinical tools, extension to detection of CHD, use of deep learning methods, and extension to other areas of fetal scanning.

10.3.1 Towards Clinical Tools and Trials

While the work presented here has been motivated by the aim of creating clinical tools, there is still significant work to be done before this is achieved. This work would likely have several facets.

The algorithms need to be trained and evaluated on a larger dataset featuring longer videos and compared to the annotations of several experts. This should also include a greater variety of views, starting with a finer classification of the right ventricular outflow views (see §1.4.4) and possibly continuing to other views such as sagittal views of the aortic and ductal arches (though this moves beyond the requirements standard screening scans). This stage should also think carefully about a more consistent and clinically useful way to deal with the ambiguity surrounding hidden, visible and obscured hearts.

Two key assumptions of the work presented in this thesis should also be examined. Firstly the assumption that the size of the heart in the image can be specified in advance, based on gestational age and image magnification factors, should be investigated empirically. If it is found that the heart size cannot be specified to a sufficient degree of accuracy for the detection models to give good performance, changes to the framework to allow for unknown size should be considered. This would likely involve incorporating a 'scale' variable into the state tuple, with different observation potentials trained to detect structures at different scales.

The second assumption is that all the videos contain images with a single 'flip convention', i.e. the probe is positioned such that the imaging plane is viewed from the direction running from the fetal head to the fetal feet. It is unlikely that it is appropriate to make this assumption in practice, as sonographers use different flip conventions. This could be addressed in similar way to the scale,

by incorporating the 'flip convention' into the state tuple and using observation potentials trained for each of the two cases.

Once the above issues have been addressed, the aim would be to produce a usable prototype of a clinical tool that can analyse video streams in real time and present them back to the user. The first task here is choosing suitable hardware and software tools to implement a system that can capture and analyse ultrasound imaging data in real time. The existing, open source PLUS framework[1] is one example of an existing project that may be able to help considerably in this stage. If computational performance is a problem, the use of graphics processing units (GPUs) in some key tasks in the framework could be considered. Several computational tasks in the framework lend themselves well to the parallel nature of GPU processing, in particular performing the FFT operations involved in calculating the RIFs, querying a random forest model containing a large number of trees, and sampling from the predictions independently for each particle in a large particle set.

There are then number of important decisions to be made surrounding how the user should interact with the tool, and how the algorithm's estimates are presented to the sonographer or used for guidance. For example, should the user have controls to alter the parameters of the tool, or should they be able to reset it if it fails? Should the tool present the user with a checklist of views to be investigated, and which have been already observed? Should the tool indicate which direction to move the probe in order to reach the views that have not yet been observed?

Once such a prototype has been constructed, a number of clinical trials should be conducted to determine the effectiveness of the tool for helping sonographers of different levels of experience perform screening scans. Possible performance metrics to monitor could include the rate at which key views are missed or captured inadequately. Another key question is what insight can be gained from these sorts of data aggregated across several sonographers and scanning sessions.

[1] `https://app.assembla.com/spaces/plus/wiki`, see also [182]

10.3.2 Extension to Automatic Detection of CHD

The most promising aspect of this work is its potential to serve as a basis for systems that can automatically screen for CHD. However, this has not yet been explored due to a lack of data from subjects displaying signs of CHD. Such data is difficult to obtain in the numbers required to train and test a machine learning model.

This task is broad and open-ended due to the large variety of defects within CHD. On top of this there will be complications due to the fact that defects often co-occur. The different forms of CHD are likely to need different approaches. For example, subtle and/or highly localised anomalies such as septal defects are unlikely to adversely affect the operation of the particle filtering framework presented. Consequently it may be possible to take the output of the existing algorithm without alteration and use it to build a classifier to distinguish healthy from unhealthy cases by focusing on the relevant parts (in time and space) of the video. However, other defects such as hypoplastic left heart syndrome alter the appearance of the heart in a significant way. Such defects would therefore likely disrupt the existing algorithm, and therefore require alterations to the filter itself. This could take the form of extra discrete state space variables representing healthy and unhealthy, with observation potentials designed to reweight the particles according to these variables.

The detection of some anomalies may require more detailed analysis of tissue motion and blood flowusing techniques such as Doppler ultrasound and speckle tracking in addition to the raw image input.

10.3.3 Use of Deep Learning Methods

As discussed in §2.1.4, deep learning methods, and in particular convolutional neural networks (CNNs), have produced state-of-the-art results on a number of image analysis tasks in the last few years. It is therefore natural to explore whether this class of methods can improve upon the performance of the framework presented here. There are two levels at which this could operate: as a drop-in replacement for the random forest based observation potentials, and as a fully end-to-end trained model incorporating also the temporal model.

Firstly, the observation potentials used in this work could be replaced with appropriate CNN-based models, and the rest of the particle filtering model left mostly as presented. This would benefit from the highly expressive models that can be learnt with CNNs, and potentially help overcome an important downside of the presented framework, which is that the local RIF-based detectors cannot capture contextual information from the rest of the image. This is relatively straightforward for the classification/detection task and has been attempted by Sundaresan et al. [183], who used a fully convolutional network to localise the fetal heart in the image and classify it according to view, but did not make use of any temporal model or filtering. However, calculating a truly circular output in a principled way for the cardiac phase regression step using CNNs is a far less studied problem. Spatial transformer networks [184] provide a principled way of dealing with the problem of unknown orientation.

Training a full end-to-end deep model for the task requires a fundamentally different approach. Previous work on using deep learning methods to make predictions from sequential inputs has focused on recurrent neural networks (RNNs) §2.1.7). Huang et al. [185] have taken this approach to learn multi-task CNN models to predict the heart location, classification, and orientation (but not cardiac phase) and connect high level features using a recurrent network (under review at time of writing). This has the crucial advantage of allowing the whole framework, including all the feature extraction stages and temporal model, to be jointly optimised for the estimation task in an end-to-end manner. However, unlike the Bayesian filtering approach, it is not able to explicitly represent the filtering distribution probabilistically and therefore may not be as robust as the particle filtering approach to highly ambiguous sequences, and the uncertainty in the output can not be so easily interpreted.

The works of Huang et al. [185] and Sundaresan et al. [183] were collaborative projects that derived from this thesis.

10.3.4 Extension to Navigation in Broader Fetal Scanning

To a considerable extent, this work has drawn on ideas from the robotics literature relating to how robotic systems navigate in uncertain environments in the face of uncertain or cluttered sensory information [170]. Because the fetal heart is relatively small area of the fetal anatomy and the imaging planes of interest are approximately parallel transverse planes, it was sufficient to consider only the two dimensions within the imaging plane.

There is considerable scope to explore applying these ideas to the broader problem of navigation within full fetal ultrasound scanning, but this will require explicit modelling of the full 3D nature of the problem. This could be used to create systems that could analyse video streams from full fetal screening scans, and estimate and display the 3D location and orientation of the probe relative to anatomical structures, based on a learned statistical model of the fetal anatomy. This would require designing observation potentials to identify a broad set of anatomical structures, and deciding on a suitable representation and degrees of freedom for the anatomical model.

Integrating the imaging information with measurements from some sort of probe position sensor would be a very important part of such a system, as it would dramatically reduce the uncertainty in the inference problem. Tracking of the probe has been successfully demonstrated before using a number of different of different methods, including electromagnetic [186] and optical [187] trackers, some of which are implemented in the PLUS framework (§10.3.1).

Appendices

A
Appendix

Contents

A.1 Implementation of Algorithms

All algorithms described in this thesis were implemented by the author in the C++ programming language using the following open source software libraries:

- **OpenCV 3.1** (`http://opencv.org`) For reading video files and basic image processing.
- **OpenMP 4.0** (`http://openmp.org`) For multi-threading (as implemented as compiler extensions within GCC 5.4.0).

Year of Purchase	2013
CPU	Intel Core i7-3770
Clock Frequency	3.40 GHz
Threads	8
Cache	8 MB
RAM	32 GB
Operating System	Ubuntu 16.04 GNU/Linux
Architecture	`x86_64`
C++ Compiler	GCC 5.4.0

Table A.1: Specification for the desktop PC used for experiments.

Year of Purchase	2009
CPU	Intel Core 2 Duo T6400
Clock Frequency	2.00 GHz
Threads	2
Cache	2 MB
RAM	4 GB
Operating System	Manjaro GNU/Linux
Architecture	`x86_64`
C++ Compiler	GCC 6.3.1

Table A.2: Specification for the laptop PC used for the experiment in §7.5.

- **Eigen 3** (`http://eigen.tuxfamily.org`) For linear algebra and optimisation routines.
- **Boost 1.58** (`http://boost.org`) For a range of mathematical routines and other basic software utilities.

Care was taken to ensure that the implementations are computationally efficient, although some further performance gains will be possible. All C++ code was compiled using compiler optimisations to optimise for speed.

Many non-performance-critical processes, such as offline model fitting and analysis of results, were implemented in the Python programming language using the Numpy (`http://numpy.org`), Scipy (`http://scipy.org`), and Matplotlib (`http://matplotlib.org`) libraries.

The code used for all the experiments described in this thesis is available on the author's Github profile at `http://github.com/CPBridge` under the GNU public licence, and relies upon entirely open-source software. For instructions

for downloading, compiling and running the code for the full fraemwork, please see `http://github.com/CPBridge/fetal_heart_analysis`. Alternativey, certain parts of the framework are available as stand-alone software libraries that can be used for other projects:

- RIF extraction: `http://github.com/CPBridge/RIFeatures`
- Random forest models: `http://github.com/CPBridge/canopy`
- Manual annotation tool: `http://github.com/CPBridge/heart_annotation_tool`

A.2 Hardware Specifications

Unless otherwise noted, all experiments were performed on a desktop PC with the specification shown in Table A.1. The exception to this is the experiment in §7.5, which used the laptop PC in Table A.2

A.3 Multi-Threaded Feature Extraction Code

The following greatly-simplified C++ programme listing shows the basics of how raw features are calculated in a thread-safe manner.

```
#include <opencv2/core/core.hpp> /* The OpenCV Library */
#include <opencv2/imgproc/imgproc.hpp> /* The OpenCV Library */
#include <opencv2/highgui/highgui.hpp>  /* The OpenCV Library */
#include <vector> /* STL Vector */
#include <complex> /* STL complex numbers */
#include <omp.h> /* OpenMP for multi-threading */

// This class provides the rotation-invariant feature extraction object
class RIFeatExtractor
{
  public:
    // ...

  private:

    // Methods
    void refreshImage();
    void rawFeatureFrequencyCalculation(const int f, const int u, const int m);
    void rawFeatureFrequencyCalculation(const int f);
    bool checkRawFeatureValidity(const int f);
    // ...

    // Data
    // The frequency-domain basis function images
    std::vector<cv::Mat_<cv::Vec2f>> U_freq;
    // Stores the complex-valued spatial domain raw feature images
    std::vector<cv::Mat_<cv::Vec2f>> raw_feat_images;
    // Stores the FFTs of the Fourier histogram expansion of the input image
    std::vector<cv::Mat_<cv::Vec2f>> FFT_im;
    // Stores whether each faw feature image is valid
    std::vector<char> raw_features_valid;

    // Stores the value of m-values for each raw feature
    std::vector<int> raw_feat_m_list;
    // Stores the basis to use for each raw feature
    std::vector<int> raw_feat_basis_list;
    // Special multithreading lock variables for each raw feature
    // used to limit access to each raw feature to a single thread at one time
    std::vector<omp_lock_t> raw_feat_frequency_creation_thread_lock;

    // ...
};

// This function is used to calculate a single raw feature for every pixel
// in the image using Fourier domain multiplication followed by inverse FFT
void RIFeatExtractor::rawFeatureFrequencyCalculation(
  const int f, // raw feature index
  const int u, // basis function index
  const int m  // fourier histogram coefficent index
  )
{
  // A temporary intermediate array
  cv::Mat_<cv::Vec2f> temp;

  // Perform frequency domain filtering and store in the raw_feat_images
  // array

  // Element-wise product of complex-valued FFT of relevant coefficient image
  cv::mulSpectrums(FFT_im[m],U_freq[u],temp,0);
  // Inverse fast Fourier transform
  cv::idft(temp, raw_feat_images[f],cv::DFT_SCALE);
  // ...
```

```
}

// Overloaded version where the basis index u and the m value are looked up
void RIFeatExtractor::rawFeatureFrequencyCalculation(
  const int f /* raw feature index*/
  )
{
  // Look up the basis index and the value of m to use for this raw feature
  // and pass to overloaded function
  rawFeatureFrequencyCalculation(f,raw_feat_basis_list[f],raw_feat_m_list[f]);
}

// Check the validity of a raw feature image in a thread-safe manner and
// recalculate if required. This should be called any time a raw feature is
// required before attempting to read from the raw_feat_images variable
bool RIFeatExtractor::checkRawFeatureValidity(
  const int f /* raw feature index */
  )
{
  // ...

  // Placeholder for the result
  bool valid;

  // Set the lock to prevent other threads accessing this raw feature
  omp_set_lock(&(raw_feat_frequency_creation_thread_lock[f]));

  // If it's already valid, just return true
  if(raw_features_valid[f])
  {
    valid = true;
  }
  // If it's not already valid calculate the feature image
  else
  {
    rawFeatureFrequencyCalculation(f);
    valid = true;
  }

  // Release the lock to allow other threads access again
  omp_unset_lock(&(raw_feat_frequency_creation_thread_lock[f]));

  return valid;
}

// This code gets called whenever a new frame is input
void RIFeatExtractor::refreshImage()
{
  // ...

  // Mark any existing raw feature results as invalid
  std::fill(raw_features_valid.begin(),raw_features_valid.end(),false);

  // ...
}
```

A.4 Fourier-Domain Representations of the RIF Basis Functions

This section contains the derivation of the Fourier-domain representations of the rotation-invariant basis functions, $u_{j,k}(r,\theta)$, defined in §3.2.1 and specifically

Equations 3.3 and 3.4 (page 52). This representation derived here allows the calculation of rotation invariant features very efficiently via multiplication in the Fourier domain.

A.4.1 Fourier Transforms of Radially Symmetric Basis Functions

For what follows it will be convenient to describe image coordinates with a polar representation, (r, θ), and also use a polar representation, (ρ, ψ), for the spatial frequency coordinates with frequency domain radial coordinate ρ and frequency domain angular coordinate ψ. One can move directly from the polar representation of an image $u(r, \theta)$ to the polar representation of its Fourier transform $U(\rho, \psi)$ using

$$U(\rho, \psi) = \int_0^{2\pi} \int_0^1 u(r, \theta) e^{-\mathbf{i} r \rho \cos(\psi - \theta)} r \, \mathrm{d}r \, \mathrm{d}\theta \tag{A.1}$$

If the image is *radially symmetric*, i.e. a function of r only, then it can be shown that the 2D Fourier transform is also radially symmetric, i.e. a function of ρ only, and may be expressed in terms of the zero-order *Hankel transform*, $\mathcal{H}_0[\cdot]$, of the radial profile of the image [188]:

$$U(\rho) = 2\pi \mathcal{H}_0[u(r)](\rho) \tag{A.2}$$

The Hankel transform is an integral transform that expresses a continuous function as the weighted sum of *Bessel functions of the first kind*. The n^{th} order Hankel transform is defined as:

$$\mathcal{H}_n[u(r)](\rho) = \int_0^\infty u(r) J_n(\rho r) r \, \mathrm{d}r \tag{A.3}$$

where $J_n(\cdot)$ is a Bessel function of the first kind of order n.

This result can be generalised for separable functions that can be expanded into a Fourier series on the angular component [188], i.e. if

$$u(r, \theta) = \sum_{k=-\infty}^{\infty} u_k(r) e^{\mathbf{i} k \theta} \tag{A.4}$$

where

$$u_k(r) = \frac{1}{2\pi}\int_0^{2\pi} u(r,\theta)e^{-\mathbf{i}k\theta}\,\mathrm{d}\theta \tag{A.5}$$

then the polar frequency domain representation may be similarly expanded in terms of Hankel transforms of different orders

$$U(\rho,\psi) = 2\pi\sum_{k=-\infty}^{\infty}\mathbf{i}^{-k}e^{\mathbf{i}k\psi}\mathcal{H}_k[u_k(r)](\rho) \tag{A.6}$$

The basis function $u_{j,k}(r,\theta)$ used here is a special case of the form of Equation A.6, with just one term in the sum. Hence, the problem of finding the Fourier domain representations of the basis functions has been reduced to that of finding the k^{th} order Hankel transform of the radial profile and then substituting this into

$$\begin{aligned} U_{j,k}(\rho,\psi) &= \mathcal{F}_2[u_{j,k}(r,\theta)](\rho,\psi) \\ &= 2\pi\mathbf{i}^{-k}e^{\mathbf{i}k\psi}\mathcal{H}_k[p_j(r)](\rho) \end{aligned} \tag{A.7}$$

A.4.2 Hankel Transforms of Cone Profiles

This section considers finding the k^{th} order Hankel transform of a simple 'cone' profile. Later in §A.4.3 it shall be shown that the 'soft histogram' profiles used in this thesis can be constructed from 'cone' profiles by superposition. Due to the linearity of the Hankel transform, the Hankel transforms of the 'soft histogram' profiles (as required for the Equation A.7) may therefore be constructed from the Hankel transform of this cone profile using straightforward superposition also. A 'cone' profile, $q_a(r)$, of with radius a is defined as:

$$q_a(r) = \begin{cases} 1-\frac{r}{a}, & 0 \le r < a \\ 0, & \text{otherwise} \end{cases} \tag{A.8}$$

Now, defining $\hat{Q}_{a,k}(\rho)$ to be the k^{th} order Hankel transform of $q_a(r)$, for positive k:

$$\begin{aligned}\hat{Q}_{a,k}(\rho) &= \int_0^\infty q_a(r) J_k(\rho r) r \,\mathrm{d}r \\ &= \int_0^a \left(1 - \frac{r}{a}\right) J_k(\rho r) r \,\mathrm{d}r \\ &= \int_0^a r J_k(\rho r) \,\mathrm{d}r - \frac{1}{a}\int_0^a r^2 J_k(\rho r) \,\mathrm{d}r \end{aligned} \tag{A.9}$$

However, in the higher-order transforms, issues arise due to singularities in the indefinite integrals at $x = 0$. To deal with these, it is necessary to evaluate instead the improper integral:

$$\hat{Q}_{a,k}(\rho) = \lim_{\epsilon \to 0} \left(\int_\epsilon^a r J_k(\rho r) \,\mathrm{d}r \right) - \frac{1}{a} \lim_{\epsilon \to 0} \left(\int_\epsilon^a r^2 J_k(\rho r) \,\mathrm{d}r \right) \tag{A.10}$$

This may be achieved by using a Taylor series expansion of the indefinite integral around $x = 0$. Integrating Equation (A.10) for positive k (this can be achieved using computer algebra software, such as Sympy[1]) gives:

$$\begin{aligned}\hat{Q}_{a,k}(\rho) = &\frac{a^{k+2}\left(\frac{\rho}{2}\right)^k}{(k+2)\,\Gamma(k+1)} \,{}_1F_2\left(\begin{matrix}\frac{k}{2}+1 \\ \frac{k}{2}+2,\, k+1\end{matrix}\middle| -\frac{a^2\rho^2}{4}\right) \cdots \\ &- \frac{a^{k+2}\left(\frac{\rho}{2}\right)^k (k+1)\,(k+2)}{\Gamma(k+4)} \,{}_1F_2\left(\begin{matrix}\frac{k}{2}+\frac{3}{2} \\ \frac{k}{2}+\frac{5}{2},\, k+1\end{matrix}\middle| -\frac{a^2\rho^2}{4}\right) \end{aligned} \tag{A.11}$$

where ${}_1F_2\left(\begin{matrix}a_0 \\ b_0, b_1\end{matrix}\middle| x\right)$ is a generalised hypergeometric function, and $\Gamma(x)$ is a gamma function. The form in Equation A.11 is a general form for any (non-negative) value of k.

An alternative that lends itself to more straightforward computation may be found by rewriting the generalised hypergeometric functions in terms of Bessel and Struve functions, giving a different form for each value of k. This can be achieved using integral tables such as those of Rosenheinrich [189]. Experiments have shown that these forms give more efficient implementations. These forms are given in Table A.3 for orders $k \in \{0, 1, 2, 3, 4, 5, 6\}$. This table makes use of the following definitions:

[1] `www.sympy.org`

k	$\hat{Q}_{a,k}(\rho)$
0	$\frac{1}{a\rho^3}\Phi(a\rho)$
1	$\frac{a}{\rho}J_0(a\rho) - \frac{2}{\rho^2}J_1(a\rho) + \frac{1}{\rho^2}\Phi(a\rho)$
2	$\frac{1}{\rho^2}J_0(a\rho) + \frac{2}{\rho^2} - \frac{3}{a\rho^3}\Lambda_0(a\rho)$
3	$\frac{8}{a\rho^3}J_0(a\rho) - \frac{2}{\rho^2}J_1(a\rho) - \frac{8}{a\rho^3} + \frac{3}{\rho^2}\Lambda_0(a\rho)$
4	$-\frac{1}{\rho^2}J_0(a\rho) + \frac{24}{a\rho^3}J_1(a\rho) + \frac{4}{\rho^2} - \frac{15}{a\rho^3}\Lambda_0(a\rho)$
5	$-\frac{8}{a\rho^3}J_0(a\rho) + \left(\frac{64}{a^2\rho^4} - \frac{6}{\rho^2}\right)J_1(a\rho) - \frac{24}{a\rho^3} + \frac{5}{\rho^2}\Lambda_0(a\rho)$
6	$\left(\frac{1}{\rho^2} - \frac{160}{a^2\rho^4}\right)J_0(a\rho) + \left(\frac{16}{a\rho^3} + \frac{320}{a^3\rho^5}\right)J_1(a\rho) + \frac{6}{\rho^2} - \frac{35}{a\rho^3}\Lambda_0(a\rho)$

Table A.3: Table of Hankel transforms of the conical profiles expressed using Bessel and Struve functions (see Equations A.12 and A.13 for definitions of Φ and Λ_0).

$$\Phi(x) = \frac{\pi x^2}{2}\left(J_1(x)H_0(x) - J_0(x)H_1(x)\right) \tag{A.12}$$

$$\Lambda_0(x) = xJ_0(x) + \Phi(x) \tag{A.13}$$

where $H_n(\cdot)$ is a *Struve function* of order n.

Using these identities and Equation A.10 (making the substitution $x = \rho r$), it is relatively straightforward to arrive at the expressions in Table A.3.

A.4.3 Sets of Basis Functions

Returning to the set of basis functions defined in Equation 3.3, each of the profiles can be written as a combination of the cone profiles from §A.4.2 (see Figure A.1 for an illustration):

$$p_j(r) = \begin{cases} q_{a_1}(r), & j = 0 \\ 2q_{a_2}(r) - 2q_{a_1}(r), & j = 1 \\ (j+1)q_{a_{j+1}} - 2jq_{a_j} + (j-1)q_{a_{j-1}}, & j = 2, 3, \ldots, J-1 \end{cases} \tag{A.14}$$

By linearity of the Hankel transform, the Hankel transforms of the basis function profiles can therefore be evaluated using the results in Equation A.11 or Table A.3 and substitute these into Equation A.7 giving, for positive k,

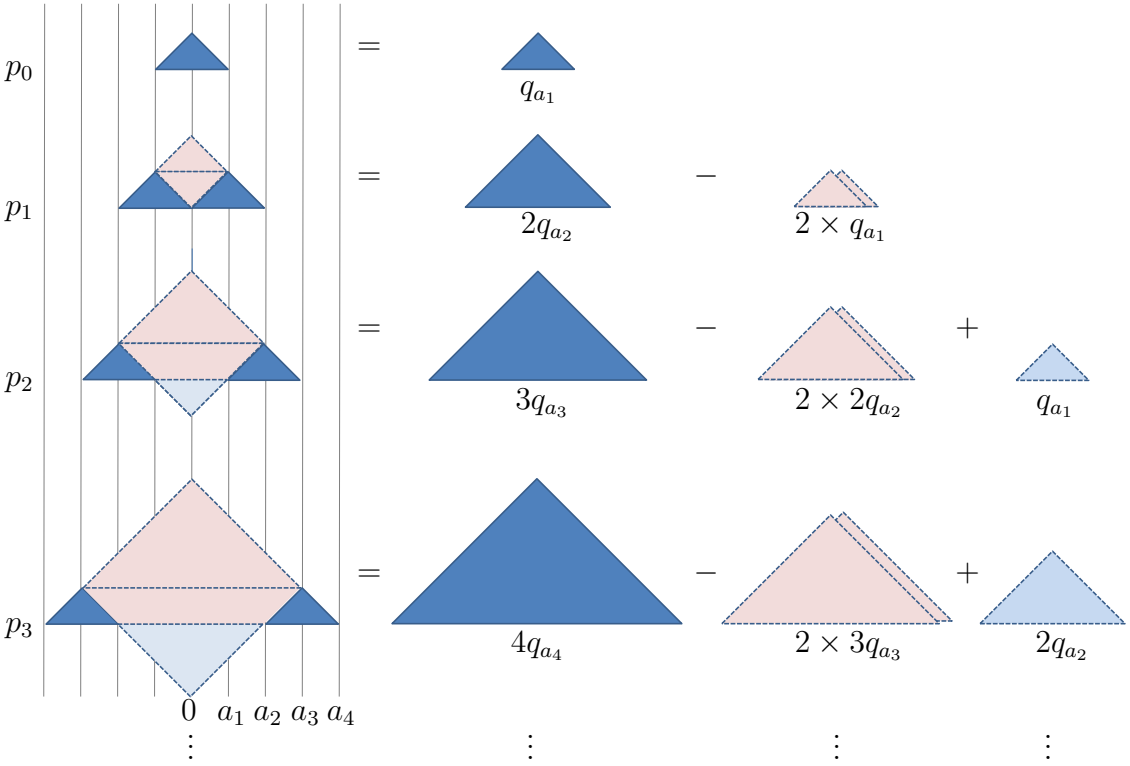


Figure A.1: Geometric illustration of Equation A.14. The radial profile of each basis function (*left*) can be constructed by the sum of the 'cone' profiles from §A.4.2 (*right*).

$$U_{j,k}(\rho,\psi) = \begin{cases} 2\pi \mathbf{i}^{-k} e^{\mathbf{i}k\psi} \hat{Q}_{a_1,k}(\rho), & j = 0 \\ 2\pi \mathbf{i}^{-k} e^{\mathbf{i}k\psi} \left[2\hat{Q}_{a_2,k}(\rho) - 2\hat{Q}_{a_1,k}(\rho) \right], & j = 1 \\ 2\pi \mathbf{i}^{-k} e^{\mathbf{i}k\psi} \times \ldots \\ \left[(j+1)\hat{Q}_{a_{j+1},k}(\rho) - 2j\hat{Q}_{a_j,k}(\rho) + (j-1)\hat{Q}_{a_{j-1},k}(\rho) \right], & 2 \le j \le J-1 \end{cases} \tag{A.15}$$

For negative k, note that the form of Equation 3.1 is such that the basis function for $-k$ is the complex conjugate of that for positive k:

$$u_{j,-k}(r,\theta) = \overline{u_{j,k}(r,\theta)} \tag{A.16}$$

and therefore, using well-known results for the Fourier transform, the spectra of the basis functions with negative k are found by simply flipping the corresponding spectra for positive k.

$$U_{j,-k}(\rho,\psi) = U_{j,k}(-\rho,-\psi) \tag{A.17}$$

References

[1] S. Yagel, H. Silverman, and G. Ulrich, eds. *Fetal Cardiology*. 2nd Edition. Informa Healthcare, 2009 (cit. on p. 2).

[2] N. Archer and N. Manning. *Fetal Cardiology*. Oxford University Press, 2009 (cit. on pp. 2, 4, 6).

[3] J. S. Carvalho, L. D. Allan, R. Chaoui, J. A. Copel, G. R. DeVore, K. Hecher, W. Lee, H. Munoz, D. Paladini, B. Tutschek, and S. Yagel. "ISUOG practice guidelines (updated): sonographic screening examination of the fetal heart". In: *Ultrasound Obstet Gynecol* 41 (2013), pp. 348–359 (cit. on pp. 5, 7, 8, 17, 21).

[4] *United Nations Millennium Development Goals*. `http://www.un.org/millenniumgoals/`. Accessed: 13/12/2016 (cit. on p. 5).

[5] M. K. Friedberg, N. H. Silverman, A. J. Moon-Grady, E. Tong, J. Nourse, B. Sorenson, J. Lee, and L. K. Hornberger. "Prenatal detection of congenital heart disease". In: *The Journal of Pediatrics* 155.1 (2009), pp. 26–31 (cit. on p. 5).

[6] D. Bonnet, A. Coltri, G. Butera, L. Fermont, J. Le Bidois, J. Kachacner, and D. Sidi. "Detection of transposition of the great arteries in fetuses reduces neonatal morbidity and mortality". In: *Circulation* 99 (Feb. 1999), pp. 916–918 (cit. on p. 5).

[7] W. Tworetzky, D. B. McElhinney, V. M. Reddy, M. M. Brook, F. L. Hanley, and N. H. Silverman. "Improved Surgical Outcome After Fetal Diagnosis of Hypoplastic Left Heart Syndrome". In: *Circulation* 103 (Mar. 2001), pp. 1269–1273 (cit. on p. 5).

[8] R. Andrews, R. Tulloh, G. Sharland, J. Simpson, S. Rollings, E. Baker, S. Qureshi, E. Rosenthal, C. Austin, and D. Anderson. "Outcome of staged reconstructive surgery for hypoplastic left heart syndrome following antenatal diagnosis". In: *Archives of Disease in Childhood* 85 (Dec. 2001), pp. 474–477 (cit. on p. 5).

[9] O. Franklin, M. Burch, N. Manning, K. Sleeman, S. Gould, and N. Archer. "Prenatal diagnosis of coarctation of the aorta improves survival and reduces morbidity". In: *Heart* 87 (Jan. 2002), pp. 67–69 (cit. on p. 5).

[10] L. J. Salomon, Z. Alfirevic, V. Berghella, C. Bilardo, E. Hernandez-Andrade, S. L. Johnsen, K. Kalache, K. Y. Leung, G. Malinger, H. Munoz, F. Prefumo, A. Toi, and W. Lee. "Practice Guidelines for Performance of the Routine Mid-Trimester Fetal Ultrasound Scan". In: *Ultrasound Obstet Gynecol* 37 (2011), pp. 116–126 (cit. on p. 7).

[11] W. Lee, L. Allan, J. Carvalho, R. Chaoui, J. Copel, G. DeVore, K. Hecher, H. Munoz, T. Nelson, D. Paladini, and S. Yagel. "ISUOG Consensus Statement: what constitutes a fetal echocardiogram?" In: *Ultrasound Obstet Gynecol* 32 (Aug. 2008), pp. 239–242 (cit. on p. 7).

[12] R. Chaoui. “The four-chamber view: four reasons why it seems to fail in screening for cardiac abnormalities and suggestions to improve detection rate”. In: *Ultrasound Obstet Gynecol* 22 (2003), pp. 3–10 (cit. on pp. 7, 8).

[13] H.-D. Kim, D.-J. Kim, I.-J. Lee, B.-J. Rah, Y. Sawa, and J. Schaper. “Human fetal heart development after mid-term: Morphometry and ultrastructural study”. In: *Journal of Molecular and Cellular Cardiology* 24.9 (1992), pp. 949–965 (cit. on p. 8).

[14] G. D. Hill, J. Block, J. Tanem, and M. A. Frommelt. “Health Disparities in the Prenatal Detection of Critical Congenital Heart Disease”. Presented at the Pediatric Academic Societies Annual Meeting, San Diego. 2015 (cit. on p. 8).

[15] J. S. Carvalho, E. Mavrides, S. E. A., S. Campbell, and B. Thilaganathan. “Improving the effectiveness of routine prenatal screening for major congenital heart defects”. In: *Heart* 88 (2002), pp. 387–391 (cit. on p. 8).

[16] L. Allan. “Antenatal diagnosis of heart disease”. In: *Heart* 83.3 (2000), p. 367. eprint: `http://heart.bmj.com/content/83/3/367.full.pdf+html` (cit. on p. 8).

[17] P. Pézard, L. Bonnemains, F. Boussion, L. Sentilhes, P. Allory, C. Lépinard, A. Guichet, S. Triau, F. Biquard, M. Leblanc, D. Bonneau, and P. Descamps. “Influence of ultrasonographers’ training on prenatal diagnosis of congenital heart diseases: a 12-year population-based study”. In: *Prenatal Diagnosis* 28.11 (2008), pp. 1016–1022 (cit. on p. 8).

[18] S. Hunter, A. Heads, J. Wyllie, and S. Robertson. “Prenatal diagnosis of congenital heart disease in the northern region of England: benefits of a training programme for obstetric ultrasonographers”. In: *Heart* 84.3 (Sept. 2000), pp. 294–298 (cit. on p. 8).

[19] A. N. Mocumbi, E. Lameira, A. Yaksh, L. Paul, M. B. Ferreira, and D. Sidi. “Challenges on the management of congenital heart disease in developing countries”. In: *International Journal of Cardiology* 148.3 (May 2011), pp. 285–288 (cit. on p. 9).

[20] S. Shah, B. A. Bellows, A. A. Adedipe, J. E. Totten, B. H. Backland, and D. Sajed. “Perceived barriers in the use of ultrasound in developing countries”. In: *Critical Ultrasound Journal* 7.11 (June 2015) (cit. on p. 9).

[21] P. Viola and M. Jones. “Rapid Object Detection Using a Boosted Cascade of Simple Features”. In: *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1. 2001, pp. 511–518 (cit. on pp. 24, 36, 49, 92).

[22] Y. Freund and R. E. Schapire. “A Decision-theoretic Generalization of On-line Learning and an Application to Boosting”. In: *J. Comput. Syst. Sci.* 55.1 (Aug. 1997), pp. 119–139 (cit. on p. 24).

[23] N. Dalal and B. Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on*. Vol. 1. June 2005, pp. 886–893 (cit. on pp. 25, 53).

[24] S. Maji, A. Berg, and J. Malik. “Classification Using Intersection Kernel SVMs is Efficient”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2008 (cit. on p. 25).

[25] Q. Zhu, M. C. Yeh, K. T. Cheng, and S. Avidan. "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. 2006, pp. 1491–1498 (cit. on pp. 25, 92).

[26] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. "Object Detection with Discriminatively Trained Part-Based Models". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.9 (Sept. 2010), pp. 1627–1645 (cit. on pp. 25, 31).

[27] P. Dollár, Z. Tu, P. Perona, and S. Belongie. "Integral Channel Features". In: *BMVC*. 2009 (cit. on p. 25).

[28] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. "Pedestrian detection at 100 frames per second". In: *Computer Vision and Pattern Recognition, 2012 IEEE Conference on*. June 2012, pp. 2903–2910 (cit. on pp. 25, 26).

[29] C. Wojek, S. Walk, and B. Schiele. "Multi-cue onboard pedestrian detection". In: *Computer Vision and Pattern Recognition, 2013 IEEE Conference on* (2009), pp. 794–801 (cit. on p. 25).

[30] L. Bourdev and J. Brandt. "Robust object detection via soft cascade". In: *Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on*. Vol. 2. June 2005, pp. 236–243 (cit. on p. 26).

[31] C. Zhang and P. Viola. "Multiple-instance pruning for learning efficient cascade detectors". In: *NIPS*. 2007 (cit. on p. 26).

[32] P. Dollár, S. Belongie, and P. Perona. "The Fastest Pedestrian Detector in the West". In: *BMVC*. 2010 (cit. on p. 26).

[33] P. Dollár, R. Appel, and W. Kienzle. "Crosstalk Cascades for Frame-Rate Pedestrian Detection". In: *European Conference on Computer Vision 2012*. 2012 (cit. on p. 26).

[34] J. Marín, D. Vázquez, A. M. López, J. Amores, and B. Leibe. "Random Forests of Local Experts for Pedestrian Detection." In: *ICCV*. IEEE Computer Society, 2013, pp. 2592–2599 (cit. on p. 26).

[35] L. Breiman. *Random Forests*. Tech. rep. TR567. U. C. Berkeley, 1999 (cit. on pp. 26, 28, 37, 73).

[36] L. Breiman. "Random Forests". In: *Machine learning* 45.1 (2001), pp. 5–32 (cit. on pp. 26, 37, 73).

[37] A. Criminisi, J. Shotton, and E. Konukoglu. *Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning.* Tech. rep. MSR-TR-2011-114. Microsoft Research, Oct. 2011 (cit. on pp. 26, 73, 74).

[38] P. Dollár, C. Wojek, B. Schiele, and P. Perona. "Pedestrian Detection: An Evaluation of the State of the Art". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.4 (Apr. 2012), pp. 743–761 (cit. on p. 26).

[39] C. Harris and M. Stephens. "A combined corner and edge detector". In: *Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151 (cit. on p. 27).

[40] D. G. Lowe. “Object Recognition from Local Scale-Invariant Features”. In: *Proceedings of the International Conference on Computer Vision*. Vol. 2. 1999, pp. 1150–1157 (cit. on pp. 27, 48).

[41] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. “Speeded-Up Robust Features (SURF)”. In: *Comput. Vis. Image Underst.* 110.3 (June 2008), pp. 346–359 (cit. on p. 27).

[42] B. Leibe, A. Leonardis, and B. Schiele. “Robust Object Detection with Interleaved Categorization and Segmentation”. In: *International Journal of Computer Vision* 77 (2008), pp. 259–289 (cit. on p. 27).

[43] R. Fergus, P. Perona, and A. Zisserman. “Object class recognition by unsupervised scale-invariant learning”. In: *Computer Visiona and Pattern Recognition, 2003 IEEE Conference on*. 2003, pp. 264–271 (cit. on pp. 27, 30).

[44] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. “Visual categorization with bags of keypoints”. In: *Workshop on Statistical Learning in Computer Vision, ECCV*. 2004, pp. 1–22 (cit. on p. 27).

[45] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. “Discovering objects and their location in images”. In: *Computer Vision, Tenth IEEE International Conference on*. Vol. 1. Oct. 2005, pp. 370–377 (cit. on pp. 27, 30).

[46] R. O. Duda and P. E. Hart. “Use of the Hough Transformation to Detect Lines and Curves in Pictures.” In: *Commun. ACM* 15.1 (1972), pp. 11–15 (cit. on p. 27).

[47] D. H. Ballard. “Generalizing the Hough transform to detect arbitrary shapes.” In: *Pattern Recognition* 13.2 (Mar. 29, 2006), pp. 111–122 (cit. on p. 27).

[48] J. Gall and V. Lempitsky. “Class-specific Hough forests for object detection”. In: *Computer Vision and Pattern Recognition, 2009 IEEE Conference on*. June 2009, pp. 1022–1029 (cit. on p. 28).

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE*. Vol. 86. 1998, pp. 2278–2324 (cit. on p. 28).

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on p. 28).

[51] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014) (cit. on pp. 28, 29).

[52] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition.” In: *CoRR* abs/1512.03385 (2015) (cit. on pp. 28, 29).

[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions”. In: *CVPR*. IEEE Computer Society, 2015, pp. 1–9 (cit. on pp. 28, 29).

[54] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”. In: *Psychological Review* 65.6 (1958), pp. 386–408 (cit. on p. 28).

[55] P. J. Werbos. “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences”. PhD thesis. Harvard University, 1974 (cit. on p. 28).

[56] X. Glorot, A. Bordes, and Y. Bengio. “Deep Sparse Rectifier Neural Networks.” In: *AISTATS*. Ed. by G. J. Gordon, D. B. Dunson, and M. Dudík. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 315–323 (cit. on p. 29).

[57] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *CoRR* abs/1207.0580 (2012) (cit. on p. 29).

[58] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. “How transferable are features in deep neural networks?” In: *CoRR* abs/1411.1792 (2014) (cit. on pp. 29, 44).

[59] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation.” In: *CoRR* abs/1311.2524 (2013) (cit. on p. 29).

[60] R. B. Girshick. “Fast R-CNN”. In: *CoRR* abs/1504.08083 (2015) (cit. on p. 29).

[61] S. Ren, K. He, R. B. Girshick, and J. Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *NIPS*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. 2015, pp. 91–99 (cit. on p. 29).

[62] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation.” In: *CVPR*. IEEE Computer Society, 2015, pp. 3431–3440 (cit. on pp. 29, 44).

[63] L. Fei-Fei, R. Fergus, and P. Perona. “Learning Generative Visual Models for 101 Object Categories”. In: *Computer Vision and Image Understanding* (2006) (cit. on p. 30).

[64] R. Fergus, P. Perona, and A. Zisserman. “A Sparse Object Category Model for Efficient Learning and Exhaustive Recognition”. In: *Computer Vision and Pattern Recognition, 2005 IEEE Conference on*. Vol. 1. 2005, pp. 380–397 (cit. on p. 30).

[65] M. Weber, M. Welling, and P. Perona. “Unsupervised learning of models for recognition”. In: *ECCV*. 2000, pp. 18–32 (cit. on p. 30).

[66] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. “Spatial priors for part-based recognition using statistical models”. In: *CVPR*. 2005, pp. 10–17 (cit. on p. 30).

[67] M. A. Fischler and R. A. Elschlager. “The Representation and Matching of Pictorial Structures.” In: *IEEE Trans. Computers* 22.1 (1973), pp. 67–92 (cit. on p. 30).

[68] P. F. Felzenszwalb and D. P. Huttenlocher. “Pictorial Structures for Object Recognition”. In: *Int. J. Comput. Vision* 61.1 (Jan. 2005), pp. 55–79 (cit. on pp. 30, 31).

[69] P. F. Felzenszwalb, R. B. Girshick, and D. A. McAllester. “Cascade object detection with deformable part models”. In: *CVPR*. 2010, pp. 2241–2248 (cit. on p. 31).

[70] Y. Yang and D. Ramanan. “Articulated pose estimation with flexible mixtures-of-parts.” In: *CVPR*. IEEE Computer Society, 2011, pp. 1385–1392 (cit. on p. 31).

[71] X. Zhu and D. Ramanan. “Face detection, pose estimation, and landmark localization in the wild.” In: *CVPR*. IEEE Computer Society, 2012, pp. 2879–2886 (cit. on p. 31).

[72] J. Shotton, R. B. Girshick, A. W. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. “Efficient Human Pose Estimation from Single Depth Images.” In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.12 (2013), pp. 2821–2840 (cit. on pp. 31, 73, 92).

[73] A. Criminisi, D. P. Robertson, E. Konukoglu, J. Shotton, S. Pathak, S. White, and K. M. Siddiqui. “Regression forests for efficient anatomy detection and localization in computed tomography scans.” In: *Medical Image Analysis* 17.8 (2013), pp. 1293–1303 (cit. on pp. 31, 73, 92).

[74] M. Dantone, J. Gall, G. Fanelli, and L. J. V. Gool. “Real-time facial feature detection using conditional regression forests”. In: *CVPR*. IEEE Computer Society, 2012, pp. 2578–2585 (cit. on pp. 31, 32).

[75] H. Yang and I. Patras. “Fine-Tuning Regression Forests Votes for Object Alignment in the Wild.” In: *IEEE Trans. Image Processing* 24.2 (2015), pp. 619–631 (cit. on pp. 31, 32).

[76] X. Jia, H. Yang, K.-P. Chan, and I. Patras. “Structured Semi-supervised Forest for Facial Landmarks Localization with Face Mask Reasoning.” In: *BMVC*. Ed. by M. F. Valstar, A. P. French, and T. P. Pridmore. BMVA Press, 2014 (cit. on pp. 31, 32).

[77] N. Razavi, J. Gall, P. Kohli, and L. J. van Gool. “Latent Hough Transform for Object Detection.” In: *ECCV (3)*. Ed. by A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid. Vol. 7574. Lecture Notes in Computer Science. Springer, 2012, pp. 312–325 (cit. on pp. 31, 32).

[78] M. Sun, P. Kohli, and J. Shotton. “Conditional regression forests for human pose estimation.” In: *CVPR*. IEEE Computer Society, 2012, pp. 3394–3401 (cit. on pp. 31, 32).

[79] H. Yang and I. Patras. “Privileged information-based conditional regression forest for facial feature detection.” In: *FG*. IEEE Computer Society, 2013, pp. 1–6 (cit. on pp. 31, 32).

[80] M. Dantone, J. Gall, C. Leistner, and L. J. V. Gool. “Human Pose Estimation Using Body Parts Dependent Joint Regressors.” In: *CVPR*. IEEE Computer Society, 2013, pp. 3041–3048 (cit. on p. 32).

[81] A. Toshev and C. Szegedy. “DeepPose: Human Pose Estimation via Deep Neural Networks.” In: *CVPR*. IEEE Computer Society, 2014, pp. 1653–1660 (cit. on p. 32).

[82] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. “R-CNNs for pose estimation and action detection”. In: *arXiv preprint arXiv:1406.5212* (2014) (cit. on pp. 32, 33).

[83] A. Newell, K. Yang, and J. Deng. “Stacked Hourglass Networks for Human Pose Estimation.” In: *ECCV (8)*. Ed. by B. Leibe, J. Matas, N. Sebe, and M. Welling. Vol. 9912. Lecture Notes in Computer Science. Springer, 2016, pp. 483–499 (cit. on pp. 32, 33).

[84] V. Belagiannis and A. Zisserman. "Recurrent Human Pose Estimation." In: *CoRR* abs/1605.02914 (2016) (cit. on pp. 32, 33).

[85] Y. Sun, X. Wang, and X. Tang. "Deep Convolutional Network Cascade for Facial Point Detection." In: *CVPR*. IEEE Computer Society, 2013, pp. 3476–3483 (cit. on p. 32).

[86] M. Oberweger, P. Wohlhart, and V. Lepetit. "Hands Deep in Deep Learning for Hand Pose Estimation." In: *CoRR* abs/1502.06807 (2015) (cit. on p. 32).

[87] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. "Dynamic Textures." In: *International Journal of Computer Vision* 51.2 (2003), pp. 91–109 (cit. on p. 33).

[88] A. B. Chan and N. Vasconcelos. "Classifying Video with Kernel Dynamic Textures." In: *CVPR*. IEEE Computer Society, 2007 (cit. on pp. 33, 34).

[89] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. "Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions". In: *Computer Vision and Pattern Recognition, 2009 IEEE Conference on*. June 2009, pp. 1932–1939 (cit. on pp. 33, 34).

[90] A. Yilmaz, O. Javed, and M. Shah. "Object tracking: A survey". In: *ACM Comput. Surv.* 38.4 (2006) (cit. on pp. 34, 35).

[91] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. "Tracking in Low Frame Rate Video: A Cascade Particle Filter with Discriminative Observers of Different Life Spans." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 30.10 (2008), pp. 1728–1740 (cit. on p. 34).

[92] D. Comaniciu, V. Ramesh, and P. Meer. "Kernel-Based Object Tracking". In: *IEEE Transactions On Pattern Analysis and Machine Intelligence (PAMI)* 25.5 (2003), pp. 564–575 (cit. on p. 34).

[93] C. Tomasi and T. Kanade. *Detection and Tracking of Point Features*. 1991 (cit. on p. 34).

[94] M. Isard and A. Blake. *Condensation – conditional density propagation for visual tracking*. 1998 (cit. on p. 35).

[95] M. Isard. "PAMPAS: Real-Valued Graphical Models for Computer Vision". In: *CVPR (1)*. IEEE Computer Society, 2003, pp. 613–620 (cit. on p. 35).

[96] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. "Nonparametric Belief Propagation". In: *CVPR (1)*. IEEE Computer Society, 2003, pp. 605–612 (cit. on p. 35).

[97] L. Sigal. "Continuous-state Graphical Models for Object Localization, Pose Estimation and Tracking ". PhD thesis. Brown University, 2008 (cit. on p. 35).

[98] L. Sigal, M. Isard, H. W. Haussecker, and M. J. Black. "Loose-limbed People: Estimating 3D Human Pose and Motion Using Non-parametric Belief Propagation." In: *International Journal of Computer Vision* 98.1 (2012), pp. 15–48 (cit. on p. 35).

[99] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. "Visual Hand Tracking Using Nonparametric Belief Propagation." In: *CVPR Workshops*. IEEE Computer Society, 2004, p. 189 (cit. on p. 35).

[100] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9 (1997), pp. 1735–1780 (cit. on p. 35).

[101] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. “Long-term Recurrent Convolutional Networks for Visual Recognition and Description”. In: *CoRR* abs/1411.4389 (2014) (cit. on p. 35).

[102] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. “Recurrent Network Models for Human Dynamics”. In: *ICCV*. IEEE Computer Society, 2015, pp. 4346–4354 (cit. on p. 36).

[103] P. Ondrúška and I. Posner. “Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks.” In: *CoRR* abs/1602.00991 (2016) (cit. on p. 36).

[104] B. Rahmatullah, I. Sarris, A. Papageorghiou, and J. A. Noble. “Quality control of fetal ultrasound images: Detection of abdomen anatomical landmarks using AdaBoost”. In: *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*. Mar. 2011, pp. 6–9 (cit. on pp. 36, 92).

[105] B. Rahmatullah, A. Papageorghiou, and J. A. Noble. “Automated Selection of Standardized Planes from Ultrasound Volume”. English. In: *Machine Learning in Medical Imaging*. Vol. 7009. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 35–42 (cit. on p. 36).

[106] P. Kovesi. “Symmetry and asymmetry from local phase”. In: *Tenth Australian Joint Conference on Artificial Intelligence*. Vol. 190. 1997 (cit. on pp. 36, 42).

[107] B. Rahmatullah, A. Papageorghiou, and J. Noble. “Integration of Local and Global Features for Anatomical Object Detection in Ultrasound”. English. In: *Medical Image Computing and Computer-Assisted Intervention*. Vol. 7512. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 402–409 (cit. on pp. 37, 92).

[108] B. Georgescu, X. Zhou, D. Comaniciu, and A. Gupta. “Database-guided segmentation of anatomical structures with complex appearance”. In: *Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on*. Vol. 2. June 2005, pp. 429–436 (cit. on pp. 37, 92).

[109] T. Karavides, K. Y. E. Leung, P. Paclik, E. A. Hendriks, and J. G. Bosch. “Database guided detection of anatomical landmark points in 3D images of the heart.” In: *ISBI*. IEEE, 2010, pp. 1089–1092 (cit. on pp. 37, 92).

[110] S. K. Zhou, J. H. Park, B. Georgescu, D. Comaniciu, C. Simopoulos, and J. Otsuki. “Image-Based Multiclass Boosting and Echocardiographic View Classification”. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. 2006, pp. 1559–1565 (cit. on pp. 37, 40, 92).

[111] G. Carneiro, B. Georgescu, S. Good, and D. Comaniciu. “Detection and Measurement of Fetal Anatomies from Ultrasound Images using a Constrained Probabilistic Boosting Tree”. In: *Medical Imaging, IEEE Transactions on* 27.9 (Sept. 2008), pp. 1342–1355 (cit. on pp. 37, 39, 48, 92).

[112] G. Carneiro, F. Amat, B. Georgescu, S. Good, and D. Comaniciu. "Semantic-based indexing of fetal anatomies from 3-D ultrasound data using global/semi-local context and sequential sampling". In: *Computer Vision and Pattern Recognition, 2008 IEEE Conference on*. June 2008, pp. 1–8 (cit. on pp. 37, 38, 92).

[113] Z. Tu. "Probabilistic boosting-tree: learning discriminative models for classification, recognition, and clustering". In: *Computer Vision, Tenth IEEE International Conference on*. Vol. 2. Oct. 2005, pp. 1589–1596 (cit. on p. 37).

[114] M. Sofka, J. Zhang, S. Zhou, and D. Comaniciu. "Multiple Object Detection by Sequential Monte Carlo and Hierarchical Detection Network". In: *Computer Vision and Pattern RecognitionProceedings, 2010 IEEE Conference on*. San Francisco, CA, June 2010 (cit. on pp. 37–39, 92).

[115] M. Sofka, K. Ralovich, N. Birkbeck, J. Zhang, and S. K. Zhou. "Integrated Detection Network (IDN) for Pose and Boundary Estimation in Medical Images". In: *Proceedings of the 8th International Symposium on Biomedical Imaging (ISBI 2011)*. Chicago, IL, Apr. 2011 (cit. on pp. 37–39, 92).

[116] M. Sofka, J. Zhang, S. Good, S. K. Zhou, and D. Comaniciu. "Automatic Detection and Measurement of Structures in Fetal Head Ultrasound Volumes Using Sequential Estimation and Integrated Detection Network (IDN)". In: *IEEE Transactions on Medical Imaging* 33.5 (May 2014), pp. 1054–1070 (cit. on pp. 37–39, 92).

[117] J. Park, M. Sofka, S. Lee, D. Kim, and S. K. Zhou. "Automatic Nuchal Translucency Measurement from Ultrasonography". In: *Proceedings of the 16th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2013)*. Nagoya, Japan, Sept. 2011 (cit. on pp. 37–39, 92).

[118] A. I. L. Namburete, B. Rahmatullah, and J. A. Noble. "Nakagami-Based AdaBoost Learning Framework for Detection of Anatomical Landmarks in 2D Fetal Neurosonograms". In: *Annals of the BMVA* 2 (2013), pp. 1–16 (cit. on pp. 37, 92).

[119] C. Wachinger, T. Klein, and N. Navab. "The 2D analytic signal for envelope detection and feature extraction on ultrasound images". In: *Medical Image Analysis* 16.6 (2012), pp. 1073–1084 (cit. on p. 37).

[120] F. Destrempes, J. Meunier, M.-F. Giroux, G. Soulez, and G. Cloutier. "Segmentation in ultrasonic B-mode images of healthy carotid arteries using mixtures of Nakagami distributions and stochastic optimization". In: *Medical Imaging, IEEE Transactions on* 28.2 (2009), pp. 215–229 (cit. on p. 37).

[121] D. Ni, X. Yang, C. Xin, C.-T. Chin, S. Chen, P.-A. Heng, S. Li, J. Qin, and T. Wang. "Standard Plane Localization in Ultrasound by Radial Component Model and Selective Search". In: *Ultrasound in Medicine and Biology* 40.11 (2014), pp. 2728–2742 (cit. on pp. 37, 38, 73).

[122] M. Yaqub, B. Kelly, A. T. Papageorghiou, and J. A. Noble. "Guided Random Forests for Identification of Key Fetal Anatomy and Image Categorization in Ultrasound Scans." In: *MICCAI (3)*. Ed. by N. Navab, J. Hornegger, W. M. W. III, and A. F. Frangi. Vol. 9351. Lecture Notes in Computer Science. Springer, 2015, pp. 687–694 (cit. on pp. 37, 73).

[123] M. Yaqub, R. Napolitano, C. Ioannou, A. T. Papageorghiou, and J. A. Noble. "Automatic detection of local fetal brain structures in ultrasound images". In: *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*. May 2012, pp. 1555–1558 (cit. on pp. 38, 73).

[124] M. Yaqub, M. K. Javaid, C. Cooper, and J. A. Noble. "Improving the Classification Accuracy of the Classic RF Method by Intelligent Feature Selection and Weighted Voting of Trees with Application to Medical Image Segmentation". In: *MLMI*. Ed. by K. Suzuki, F. Wang, D. Shen, and P. Yan. Vol. 7009. Lecture Notes in Computer Science. Springer, 2011, pp. 184–192 (cit. on pp. 38, 73).

[125] V. S. Lempitsky, M. Verhoek, J. A. Noble, and A. Blake. "Random Forest Classification for Automatic Delineation of Myocardium in Real-Time 3D Echocardiography." In: *FIMH*. Ed. by N. Ayache, H. Delingette, and M. Sermesant. Vol. 5528. Lecture Notes in Computer Science. Springer, 2009, pp. 447–456 (cit. on pp. 38, 73).

[126] M. A. Maraci, R. Napolitano, A. Papageorghiou, and J. A. Noble. "Object Classification in an Ultrasound Video Using LP-SIFT Features". In: *MCV*. Ed. by B. H. Menze, G. Langs, A. Montillo, B. M. Kelm, H. Müller, S. Zhang, T. W. Cai, and D. N. Metaxas. Vol. 8848. Lecture Notes in Computer Science. Springer, 2014, pp. 71–81 (cit. on p. 38).

[127] M. A. Maraci, R. Napolitano, A. Papageorghiou, and J. A. Noble. "Fisher vector encoding for detecting objects of interest in ultrasound videos". In: *ISBI*. IEEE, 2015, pp. 651–654 (cit. on p. 38).

[128] A. M. C. Kumar and K. S. Shriram. "Automated scoring of fetal abdomen ultrasound scan-planes for biometry." In: *ISBI*. IEEE, 2015, pp. 862–865 (cit. on p. 38).

[129] D. Agarwal, K. S. Shriram, and N. Subramanian. "Automatic view classification of echocardiograms using Histogram of Oriented Gradients". In: *Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on*. Apr. 2013, pp. 1368–1371 (cit. on p. 40).

[130] H. Wu, D. M. Bowers, T. T. Huynh, and R. Souvenir. "Echocardiogram view classification using low-level features". In: *Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on*. Apr. 2013, pp. 752–755 (cit. on p. 40).

[131] A. Oliva and A. Torralba. "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope". In: *International Journal of Computer Vision* 42 (2001), pp. 145–175 (cit. on p. 40).

[132] J. H. Park, S. K. Zhou, C. Simopoulos, J. Otsuki, and D. Comaniciu. "Automatic cardiac view classification of echocardiogram". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, pp. 1–8 (cit. on p. 40).

[133] Y. Qian, L. Wang, C. Wang, and X. Gao. "The Synergy of 3D SIFT and Sparse Codes for Classification of Viewpoints from Echocardiogram Videos". In: *Medical Content-Based Retrieval for Clinical Decision Support*. Vol. 7723. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 68–79 (cit. on p. 40).

[134] R. Kumar, F. Wang, D. Beymer, and T. Syeda-Mahmood. "Echocardiogram view classification using edge filtered scale-invariant motion features". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. June 2009, pp. 723–730 (cit. on p. 40).

[135] S. Ebadollahi, S.-F. Chang, and H. Wu. "Automatic View Recognition in Echocardiogram Videos Using Parts-based Representation". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR'04. Washington, D.C., USA: IEEE Computer Society, 2004, pp. 2–9 (cit. on p. 41).

[136] L. Yeo and R. Romero. "Fetal Intelligent Navigation Echocardiography (FINE): a novel method for rapid, simple, and automatic examination of the fetal heart". In: *Ultrasound in Obstetrics & Gynecology* 42.3 (2013), pp. 268–284 (cit. on p. 41).

[137] R. Kwitt, N. Vasconcelos, S. Razzaque, and S. R. Aylward. "Recognition in Ultrasound Videos: Where Am I?" In: *MICCAI (3)*. Ed. by N. Ayache, H. Delingette, P. Golland, and K. Mori. Vol. 7512. Lecture Notes in Computer Science. Springer, 2012, pp. 83–90 (cit. on p. 41).

[138] R. Kwitt, N. Vasconcelos, S. Razzaque, and S. R. Aylward. "Localizing target structures in ultrasound video - A phantom study." In: *Medical Image Analysis* 17.7 (2013), pp. 712–722 (cit. on p. 41).

[139] M. A. Maraci, R. Napolitano, A. Papageorghiou, and J. A. Noble. "Searching for Structures of Interest in an Ultrasound Video Sequence". In: *Machine Learning in Medical Imaging*. Vol. 8679. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 133–140 (cit. on pp. 41, 42).

[140] M. A. Maraci, C. P. Bridge, R. Napolitano, A. T. Papageorghiou, and J. A. Noble. "A Framework for Analysis of Linear Ultrasound Videos to Detect Fetal Presentation and Heartbeat". In: *Medical Image Analysis* 37 (Apr. 2017), pp. 22–36 (cit. on pp. 41, 42).

[141] C. P. Bridge. "Introduction To The Monogenic Signal". In: *arXiv* (2017). arXiv: `1703.09199 [cs.CV]` (cit. on p. 42).

[142] G. Jacob, J. A. Noble, and A. Blake. "Robust contour tracking in echocardiographic sequences". In: *Computer Vision, 1998. Sixth International Conference on*. Jan. 1998, pp. 408–413 (cit. on p. 42).

[143] J. C. Nascimento and J. S. Marques. "Robust Shape Tracking With Multiple Models in Ultrasound Images". In: *IEEE Transactions on Image Processing* 17 (2008), pp. 392–406 (cit. on p. 42).

[144] G. Carneiro and J. C. Nascimento. "Combining Multiple Dynamic Models and Deep Learning Architectures for Tracking the Left Ventricle Endocardium in Ultrasound Data". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.11 (Nov. 2013), pp. 2592–2607 (cit. on p. 42).

[145] L. Yang, B. Georgescu, Y. Zheng, P. Meer, and D. Comaniciu. "3D ultrasound tracking of the left ventricle using one-step forward prediction and data fusion of collaborative trackers". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. June 2008, pp. 1–8 (cit. on p. 42).

[146] C. Butakoff, F. Sukno, A. Doltra, E. Silva, M. Sitges, and A. F. Frangi. “Order Statistic Based Cardiac Boundary Detection in 3D+t Echocardiograms.” In: *FIMH*. Ed. by D. N. Metaxas and L. Axel. Vol. 6666. Lecture Notes in Computer Science. Springer, 2011, pp. 359–366 (cit. on p. 42).

[147] M. Leitman, P. Lysyansky, S. Sidenko, V. Shir, E. Peleg, M. Binenbaum, E. Kaluski, R. Krakover, and Z. Vered. “Two-dimensional strain–a novel software for real-time quantitative echocardiographic assessment of myocardial function”. In: *Journal of the American Society of Echocardiography* 17.10 (2004), pp. 1021–1029 (cit. on p. 43).

[148] H. Pavlopoulos and P. Nihoyannopoulos. “Strain and strain rate deformation parameters: from tissue Doppler to 2D speckle tracking”. In: *The International Journal of Cardiovascular Imaging* 24.5 (June 2008), pp. 479–491 (cit. on p. 43).

[149] H. Blessberger and T. Binder. “Two dimensional speckle tracking echocardiography: basic principles”. In: *Heart* 96.9 (2010), pp. 716–722 (cit. on p. 43).

[150] G. R. DeVore, B. Polanco, G. Satou, and M. Sklansky. “Two-Dimensional Speckle Tracking of the Fetal Heart”. In: *Journal of Ultrasound in Medicine* 35.8 (2016), pp. 1765–1781 (cit. on p. 43).

[151] H. Chen, Q. Dou, D. Ni, J.-Z. Cheng, J. Qin, S. Li, and P.-A. Heng. “Automatic Fetal Ultrasound Standard Plane Detection Using Knowledge Transferred Recurrent Neural Networks”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Vol. 9349. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 507–514 (cit. on p. 44).

[152] Y. Gao, M. A. Maraci, and J. A. Noble. “Describing Ultrasound Video Content Using Deep Convolutional Neural Networks”. In: *ISBI*. IEEE, Apr. 2016, pp. 787–790 (cit. on p. 44).

[153] C. F. Baumgartner, K. Kamnitsas, J. Matthew, S. Smith, B. Kainz, and D. Rueckert. “Real-Time Standard Scan Plane Detection and Localisation in Fetal Ultrasound Using Fully Convolutional Neural Networks.” In: *MICCAI (2)*. Ed. by S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. B. Ünal, and W. Wells. Vol. 9901. Lecture Notes in Computer Science. 2016, pp. 203–211 (cit. on p. 44).

[154] C. F. Baumgartner, K. Kamnitsas, J. Matthew, T. P. Fletcher, S. Smith, L. M. Koch, B. Kainz, and D. Rueckert. “Real-Time Detection and Localisation of Fetal Standard Scan Planes in 2D Freehand Ultrasound”. In: *arXiv* abs/1612.05601 (2016) (cit. on pp. 44, 45).

[155] C. P. Bridge and J. A. Noble. “Object Localisation in Fetal Ultrasound Images Using Invariant Features”. In: *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. Apr. 2015, pp. 156–159 (cit. on p. 47).

[156] M. Villamizar, F. Moreno-Noguer, J. Andrade-Cetto, and A. Sanfeliu. “Efficient rotation invariant object detection using boosted Random Ferns”. In: *Computer Vision and Pattern Recognition, 2010 IEEE Conference on*. June 2010, pp. 1038–1045 (cit. on p. 49).

[157] M. Özuysal, P. Fua, and V. Lepetit. "Fast Keypoint Recognition in Ten Lines of Code". In: *In Proc. IEEE Conference on Computing Vision and Pattern Recognition*. IEEE Computer Society, 2007 (cit. on p. 49).

[158] K. Liu, Q. Wang, W. Driever, and O. Ronneberger. "2D/3D Rotation-Invariant Detection using Equivariant Filters and Kernel Weighted Mapping". In: *Computer Vision and Pattern Recognition, 2012 IEEE Conference on*. 2012 (cit. on p. 49).

[159] K. Liu, H. Skibbe, T. Schmidt, T. Blein, K. Palme, T. Brox, and O. Ronneberger. "Rotation-Invariant HOG Descriptors Using Fourier Analysis in Polar and Spherical Coordinates". In: *International Journal of Computer Vision* 106.3 (2014), pp. 342–364 (cit. on pp. 49, 52, 53, 57, 58, 62, 65, 66, 97).

[160] H. Skibbe and M. Reisert. "Circular Fourier-HOG features for rotation invariant object detection in biomedical images". In: *ISBI*. 2012, pp. 450–453 (cit. on p. 49).

[161] A. Khotanzad and Y. H. Hong. "Invariant image recognition by Zernike moments". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 12.5 (May 1990), pp. 489–497 (cit. on p. 52).

[162] G. Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion." In: *SCIA*. Ed. by J. Bigün and T. Gustavsson. Vol. 2749. Lecture Notes in Computer Science. Springer, 2003, pp. 363–370 (cit. on p. 66).

[163] C. P. Bridge, C. Ioannou, and J. A. Noble. "Automated Annotation and Quantitative Description of Ultrasound Videos of the Fetal Heart". In: *Medical Image Analysis* 36 (Feb. 2017), pp. 147–161 (cit. on pp. 73, 86, 114, 149, 170).

[164] A. I. L. Namburete, R. V. Stebbing, B. Kemp, M. Yaqub, A. T. Papageorghiou, and J. A. Noble. "Learning-based prediction of gestational age from ultrasound images of the fetal brain." In: *Medical Image Analysis* 21.1 (2015), pp. 72–86 (cit. on pp. 73, 92).

[165] L. Breiman, J. H. Friedman, A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth, 1984 (cit. on p. 78).

[166] S. R. Jammalamadaka and A. SenGupta. *Topics in Circular Statistics*. World Scientific Pub Co Inc, 2001 (cit. on pp. 79–81, 143).

[167] G. Stienne, S. Reboul, M. Azmani, J.-B. Choquel, and M. Benjelloun. "A multi-temporal multi-sensor circular fusion filter." In: *Information Fusion* 18 (2014), pp. 86–100 (cit. on p. 81).

[168] P. Viola, M. J. Jones, and D. Snow. "Detecting pedestrians using patterns of motion and appearance". In: *Computer Vision, 2003 Ninth IEEE International Conference on*. IEEE. 2003, pp. 734–741 (cit. on p. 92).

[169] J. Shotton, J. Winn, C. Rother, and A. Criminisi. "TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation". In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*. ECCV'06. Graz, Austria: Springer-Verlag, 2006, pp. 1–15 (cit. on p. 92).

[170] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005 (cit. on pp. 115–118, 120, 172, 224).

[171] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001 (cit. on pp. 115, 118).

[172] K. P. Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA: The MIT Press, 2012 (cit. on pp. 116, 117, 124, 136, 187, 189).

[173] I. J. Cox and S. L. Hingorani. "An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 18.2 (1996), pp. 138–150 (cit. on p. 118).

[174] G. Kurz, I. Gilitschenski, and U. D. Hanebeck. "Recursive nonlinear filtering for angular data based on circular distributions." In: *ACC*. IEEE, 2013, pp. 5439–5445 (cit. on p. 118).

[175] J. MacCormick and M. Isard. "Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking." In: *ECCV (2)*. Ed. by D. Vernon. Vol. 1843. Lecture Notes in Computer Science. Springer, 2000, pp. 3–19 (cit. on pp. 119, 120, 124, 125, 166).

[176] J. MacCormick. "Stochastic algorithms for visual tracking: probabilistic modelling and stochastic algorithms for visual localisation and tracking." PhD thesis. University of Oxford, United Kingdom, UK, 2000 (cit. on pp. 120, 125, 134).

[177] B. Limketkai, D. Fox, and L. Liao. "CRF-Filters: Discriminative Particle Filters for Sequential State Estimation". In: *Robotics and Automation, 2007 IEEE International Conference on*. Apr. 2007, pp. 3142–3147 (cit. on p. 123).

[178] Y. Cheng. "Mean Shift, Mode Seeking, and Clustering." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 17.8 (1995), pp. 790–799 (cit. on p. 146).

[179] A. Karamalis, W. Wein, T. Klein, and N. Navab. "Ultrasound confidence maps using random walks." In: *Medical Image Analysis* 16.6 (2012), pp. 1101–1112 (cit. on p. 163).

[180] C. P. Bridge, C. Ioannou, and J. A. Noble. "Localizing Cardiac Structures in Fetal Heart Ultrasound Video". In: *8th International Workshop on Machine Learning in Medical Imaging, MICCAI 2017*. Québec City, Canada, Sept. 2017 (cit. on p. 180).

[181] P. R. Kumar and P. Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, 1986 (cit. on p. 193).

[182] A. Lasso, T. Heffter, A. Rankin, C. Pinter, T. Ungi, and G. Fichtinger. "PLUS: Open-Source Toolkit for Ultrasound-Guided Intervention Systems." In: *IEEE Trans. Biomed. Engineering* 61.10 (2014), pp. 2527–2537 (cit. on p. 221).

[183] V. Sundaresan, C. P. Bridge, C. Ioannou, and J. A. Noble. "Automated characterization of the fetal heart in ultrasound images using fully convolutional neural networks." In: *ISBI*. IEEE, 2017, pp. 671–674 (cit. on p. 223).

[184] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. "Spatial Transformer Networks." In: *NIPS*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. 2015, pp. 2017–2025 (cit. on p. 223).

[185] W. Huang, C. P. Bridge, and J. A. Noble. "Temporal HeartNet: Towards Human-Level Automatic Analysis of Fetal Cardiac Screening Video". In: *MICCAI 2017*. Quebéc City, Canada, Sept. 2017 (cit. on p. 223).

[186] P. R. Detmer, G. Bashein, T. Hodges, K. W. Beach, E. P. Filer, D. H. Burns, and D. E. Strandness. "3D Ultrasonic Image Feature Localization Based on Magnetic Scanhead Tracking: In Vitro Calibration and Validation". In: *Ultrasound in Medicine and Biology* 20.9 (1994), pp. 923–936 (cit. on p. 224).

[187] S.-Y. Sun, M. W. Gilbertson, and B. W. Anthony. "Probe Localization for Freehand 3D Ultrasound by Tracking Skin Features." In: *MICCAI*. Ed. by P. Golland, N. Hata, C. Barillot, J. Hornegger, and R. D. Howe. Vol. 8674. Lecture Notes in Computer Science. Springer, 2014, pp. 365–372 (cit. on p. 224).

[188] N. Baddour. "Operational and convolution properties of two-dimensional Fourier transforms in polar coordinates". In: *Journal of the Optical Society of America A* 26.8 (Aug. 2009), pp. 1767–1777 (cit. on p. 232).

[189] W. Rosenheinrich. *Tables of Some Indefinite Integrals of Bessel Functions*. Tech. rep. Ernst-Abbe-Hochschule, Jena, 2003 (cit. on p. 234).