Autonomous Robots (2026) 50:3
https://doi.org/10.1007/s10514-025-10227-2

Robot-relay: building-wide, calibration-less visual servoing with learned sensor handover networks

Luke Robinson[1] · **Matthew Gadd**[1] · **Paul Newman**[1] · **Daniele De Martini**[1]

Received: 5 June 2025 / Revised: 16 October 2025 / Accepted: 22 October 2025

Abstract

This paper proposes a novel system to conduct visual servoing of a mobile robot using multiple uncalibrated, wall-mounted cameras. Specifically, we utilise a constellation of such sensors to cover a wide area by allowing robot control to be passed between cameras in regions where their fields of view overlap. This method, in conjunction with the fact that all computing is also executed offboard, allows for simpler and cheaper robots to be deployed in controlled and finite spaces. Our method simplifies the natural installation cycle of a newly deployed camera network, eliminating the need for explicit camera positioning or orientation, both globally (relative to a building plan) and locally (among viewpoints). Our system memorises pixel-wise topological connections between viewpoints by leveraging natural human exploration of the environment. We detect graph edges through simultaneous detections of the same person across different cameras, allowing us to automatically construct an evolving graph that represents overlapping fields of view within the camera network. In combination with a hybrid-A*-based planner, our approach allows efficient planning and control of robots across a wide area by traversing cameras between areas of overlap. We validate our approach through autonomous traversals in a productive office environment, using a network of six cameras, and compare our performance against both human teleoperation and a traditional Simultaneous Localisation and Mapping (SLAM) approach.

Keywords Cloud robotics · Visual servoing · Autonomous mobile robots · Deep learning · Internet of robotic things

1 Introduction

Offboard control of mobile robots from remote viewpoints holds great promise for scalable, universal autonomy. With this paradigm, custom, expensive onboard sensors and compute can be replaced with a single, cloud-backed edge server and cameras embedded in the environment (Buoso et al., 2025). By eliminating the task of sensing and computation from the robotic platforms, these expensive pieces of infrastructure can be scaled separately from the number of robots and the efficiency of the system can therefore be enhanced (Simoens et al., 2018). However, practical implementation requires the coverage of large operational areas, which often necessitates a network of multiple sensors with overlapping fields of view (Batalin et al., 2004; Poduri & Sukhatme, 2004) to ensure comprehensive monitoring of productive spaces (e.g. industrial sites or operational environments).

Deploying such a camera network can be time-consuming and complex, and restrictions on camera placement can make installations impractical. For example, requiring ground-aligned camera image planes might be impractical for an environment with very high ceilings. Moreover, any system that requires precise calibration and setup must take into account both the initial added time cost of setup and the additional burden of maintenance when replacing cameras or when a camera becomes misaligned. A small reduction in these costs can have a significant impact when operating in a large environment that requires full coverage with many cameras.

This work presents a system that autonomously initialises and manages a network of remote camera viewpoints without prior knowledge of camera position or orientation, neither global—i.e. relative to a building plan— nor local—i.e. camera-to-camera, to control an autonomous wheeled robot.

✉ Luke Robinson
lrobinson@robots.ox.ac.uk

Daniele De Martini
daniele@robots.ox.ac.uk

[1] Mobile Robotics Group, Oxford Robotics Institute, University of Oxford, Oxford, UK

Published online: 28 November 2025

We also make no assumptions about camera placement, as long as the union of the fields of view of all cameras covers the entire desired area of operation with sufficient overlap for robots to pass between them. Instead, we leverage our prior work (Robinson et al., 2023) in effective remote control without extrinsic or intrinsic calibration, extending it to the multi-camera setting.

Our approach memorises pixel-wise topological connections representing regions of overlapping fields of view between viewpoints for a new camera network. We leverage natural human exploration of the environment and construct overlap edges in a camera graph by simultaneously detecting the same person across different cameras in raw video footage of the monitored area via state-of-the-art perception models (Varghese & Sambath, 2024; Hu et al., 2024). Using a hybrid A* approach, we can then connect overlap points in each image and across images for the robot to traverse the network by handing over control between the various cameras of the network, allowing autonomous paths to be quickly and safely created over wide areas.

The major contributions of this work are:

- **Multi-camera visual servoing of a wheeled robot**: We introduce a multi-camera, eye-to-hand visual servoing system that can autonomously control a ground robot beyond the field of view of one camera.
- **Arbitrary camera angles**: Our system accommodates cameras mounted at arbitrary orientations rather than requiring known (e.g. downwards-facing) ones, enhancing flexibility in camera placement within complex environments.
- **No inter-camera or camera-to-robot calibration required**: Our system operates without predefined camera calibration relative to each other or the robot or workspace, simplifying setup and deployment.
- **Human-centric camera network discovery**: By using natural human movement during productive workspace use, our system learns the camera network without requiring robotic demonstrations or manual setup, facilitating faster and more adaptable deployment in dynamic environments.

To support the proposed approach, we present extensive in-the-loop experiments using a network of cameras covering several rooms in a productive office space. We also compare the performance of our system to both a traditional setup with robot-mounted sensors as well as to direct human teleoperation of the robot.

This article is an extended version of our prior work (Robinson et al., 2023), adding a new methodology for discovering the camera-to-camera connections through videos of people navigating their daily environment, further simplifying the installation process and enabling seamless extension of the camera network. We also upgrade our planning and control approach from a potential field to a hybrid A* and pure pursuit approach, which improved the system in planning speed and control stability. We present this new approach with further experiments supporting its validity in a real-world application.

The remainder of this work is organised as follows. Section 2 discusses related work on autonomous navigation systems, specifically focusing on prior methods that have enabled progress in dynamic, multi-camera environments. Section 3 gives some preliminary detail of our prior work, which is the core remote-viewpoint autonomous system that this paper builds upon. Section 4 then discusses incidental advancements to that prior work made in this paper, in identifying drivable regions and more usable control policies, before describing our core contributions: a method to organically expand a graph of possible routes across a network of cameras tracking humans' day-to-day movements, and a hierarchical planning approach to traverse it by planning over the graph via the overlap points. Section 5 and 6 finally detail experiments demonstrating the system's capability to expand the route network and its application in traversing complex, interconnected workspaces, showcasing its effectiveness in multi-environment navigation.

2 Related work

This work intersects two key domains: multi-camera visual servoing and multi-camera person identification and tracking. This section briefly overviews relevant works within the two fields, highlighting those that have particularly influenced our approach. We also discuss how our proposed method differs from and builds upon existing research.

2.1 Multi-camera visual servoing

The total removal of sensors is an idea that goes back as far as (Kruse & Wahl, 1998) and has seen application in industry context too (Shalom & Rofe, 2022). Several works tackled the problem of multi-camera visual servoing by leveraging specific topologies of the camera network. In particular, a common solution is to constrain the cameras to face downwards, perpendicular to the driving surface. For instance, (Dönmez & Kocamaz, 2019) use web cameras following this strategy and, furthermore, stitch images from overlapping cameras together based on common features. Similarly, (Whitaker et al., 2020) use overhead cameras facing straight down to automatically generate waypoint graphs across multiple camera fields of view. Our approach, too, plans and controls a robot across multiple cameras, but

the fact that we naturally handle cameras mounted at arbitrary poses starkly improves the system's flexibility.

A different strategy is to calibrate the cameras in a 3D world coordinate system. Elsheikh et al. (2016) tackle this problem by manually calibrating the cameras at deployment time using a calibration board visible by pairs of cameras. Rekleitis et al. (2006), instead, used fixed driving patterns to calibrate the external cameras. Finally, Rekleitis and Dudek (2005) exploited a checkboard pattern mounted on a mobile platform to calibrate a camera network while it traverses the operational space. In contrast, we show that this extrinsic calibration is unnecessary and opt for a topological instead of a topometric representation of the camera network.

Additionally, Rekleitis et al. (2006) directed the robot to explore the camera network through various graph search strategies. Instead, we instruct our robot on fixed missions through the network (specific actions being important at the productive workspaces viewed by the cameras). We also do not map out the network with robot exploration as is done by Rekleitis et al. (2006), considering this to be inefficient and limited to having to drive the robot through the environment first manually, and instead expand it organically by observing human users of the workspaces.

2.2 Multi-camera person identification and tracking

Active object tracking, particularly the challenge of tracking people across multiple camera views, has garnered significant attention due to its applications in surveillance, smart environments, and human behaviour analysis. This field aims to seamlessly identify and track individuals as they move between the fields of view of multiple cameras.

A foundational challenge in multi-camera tracking is resolving the correspondence problem—associating individuals observed in one camera with their representations in another. Early approaches (Javed et al., 2008; Gray & Tao, 2008; Gheissari et al., 2006) employed hand-engineered feature techniques matching mainly consisting of colour histograms. However, such methods often faltered under dynamic lighting conditions, occlusions and inter-person similarity.

Recent advancements have increasingly relied on deep learning based methods. Learned methods have demonstrated significant success in person re-identification, which aims to create robust embeddings for individuals across varying viewpoints (Ye et al., 2022). These embeddings allow for the seamless transfer of tracking across cameras. One notable recent example of these methods is PersonVit (Hu et al., 2024) which not only performs well on standardised person re-ID datasets but also provides pretrained weights and an open-source implementation.

Other approaches to person and other object tracking enhance or even avoid completely the problems associated with re-identification by making use of multiple overlapping camera viewpoints to track the movement of objects without having to rely as heavily on their features (Li et al., 2020; Esterle et al., 2014; Shah et al., 2021). However, due to the robustness of modern re-identification techniques, this added complexity is not needed in this work.

3 Overview and preliminaries

Our system is illustrated in Fig.1, where a robot can navigate throughout the camera network by moving across cameras at specific *handover regions*—here highlighted in orange and purple.

3.1 System description

In our remote-viewpoint autonomous navigation system, overhead cameras monitor a robot, detecting its position and orientation and sending it actuator commands to navigate the space. For each camera, the environment is classified into drivable areas and obstacles using semantic segmentation and an in-image path planning and following method is used to control the robot.

Multiple cameras are installed, each overlapping its field of view with at least one other camera, to expand the area of operations for robots beyond the range of a single camera. The regions seen by multiple cameras are denoted as *overlap regions*. They contain a specified *handover point* used as the foundational connection between cameras. To navigate the robot through the workspace, the robot's control is iteratively passed among the cameras, with each camera guiding the robot to a handover point and then passing the control on until the robot has reached its desired end goal.

The foundational mode of autonomy, upon which our work is built, has a robot detected through a learned position and orientation methodology and controlled purely in image space. Based on previous work (Robinson et al., 2023), this preliminary system is described in Sect. 3.2. Additions required to extend this to multiple cameras are described in Sect. 4.

3.2 Robot detection and orientation estimation

Our detection module comprises two blocks: (1) a position and (2) an orientation detector. We directly employ the setup of our previous work (Robinson et al., 2023), where an updated YOLOv8 (Varghese & Sambath, 2024) network recovers the bounding box of the robot in operation and a learned orientation detector (Ruiz et al., 2018) recovers the

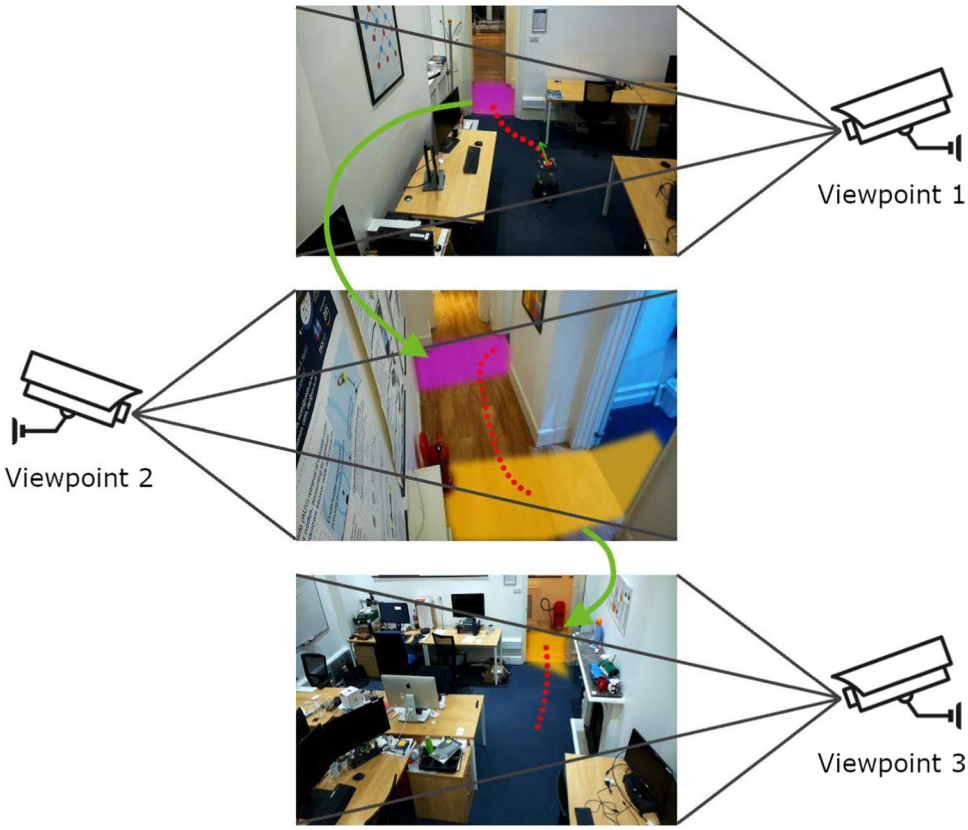


Fig. 1 A network of viewpoints from multiple overlapping cameras allows autonomous control of the ground robot beyond the field of view of one camera. Starting in viewpoint 1, the robot is controlled to navigate towards the purple overlap region, where control is then handed over to viewpoint 2. This is repeated through successive overlapping cameras until the robot reaches its goal in the final viewpoint

orientation in image space—i.e. the expected direction of motion—from the robot's appearance.

Briefly, let the input image be represented by I, where $I \in \mathbb{R}^{H \times W \times C}$, with H, W, and C as the height, width, and number of channels of the image. We detect the robot in the image using the trained YOLOv8 network, which returns the bounding box coordinates $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$ of the robot, which we can use to define the robot position as the bounding box centroid

$$x_r = \frac{x_{\min} + x_{\max}}{2}, \quad y_r = \frac{y_{\min} + y_{\max}}{2} \tag{1}$$

By then extracting the image patch of the bounding box, we can recover the orientation of the robot θ_r by feeding it through the orientation detector neural network. θ_r represents the estimate of the robot heading in image plane, i.e. its predicted direction of motion.

This in-camera detection system is replicated for all cameras, each with a position and orientation detector trained specifically on their individual scene through (1) labelled spins and (2) random environment walks (Robinson et al., 2023). For (1) the object detector is trained via *labelled spins*, observing four 360 ° rotations without translation, capturing the chassis from all sides and distances. Labels are only needed for key frames at each spin location, with interpolation for the rest, reducing manual labelling time and effort. For (2), the orientation estimator is trained through *random environment walks*, with the robot autonomously moving in straight lines within the segmented drivable region. Orientation labels are generated by observing how the centre of the robot's bounding box moves across the image when the robot is travelling in a straight line. In this way, the robot can autonomously generate a large and diverse labelled dataset of cropped robot images and corresponding orientation values, which can then be used to train our robust orientation estimator model.

For both, more detail is provided in Robinson et al. (2023).

4 Methodology

Sections 4.3 and 4.4 describe the *core contribution of this work*, in which a graph of the known cameras, connected by overlap regions, is automatically constructed through observations of people naturally using the space, and a high-level planner instructs the robot on which cameras to traverse to reach a designated goal position in a target camera.

Incidentally, for this new multi-camera setup, we have improved our previous base system with automatic drivable region segmentation in Sect. 4.1 and improved low-level

planning within each camera based on the hybrid A* algorithm in Sect. 4.2.

4.1 Drivable region estimation

Labelling of drivable regions can be tedious, and automatic detection can be challenging due to the variety of scenarios we may encounter. Therefore, we use a semi-automatic labelling approach that exploits recent advances in foundational vision models. This comprises the manual selection of multiple points in the drivable region through a GUI, which is passed as a prompt to Segment Anything Model 2 (SAM 2) (Ravi et al., 2024), with all other regions being considered non-drivable—as shown in Fig.2a. In this example, the user specifies points on the carpet and wooden floor as part of the drivable region, this is then segmented by SAM2 and marked as drivable through a binary mask.

Areas outside this region are designated as non-drivable (e.g., the desks and other objects in the office, shown as dark in Fig. 2a).

This operation is notably done *only once per camera*. Once the camera is installed, it remains stationary for the entire operation lifetime, so static obstacles in the main drivable region need only be determined once at installation. This work does not consider dynamic obstacles, although they could be detected separately and added to the static obstacle map. This approach minimises user input and maintains consistent navigation boundaries.

4.2 Low-level planner and controller

With the drivable region specified, robot navigation within each camera view proceeds in three stages: an obstacle expansion phase (applied ahead of time), a hybrid-astar-based (Dolgov et al., 2008) planning phase, and finally a control phase based on a custom pure pursuit controller (Corke, 2017). When each camera takes control of the robot, it will briefly pause the robot's motion to plan a route from the current pose to the next handover or goal point instructed by the high-level planner. Then, once a plan has been formed, the low-level controller will steer the robot along that path before alerting the high-level planner when the robot has reached the specified end point. All three stages are performed entirely in the image frame of reference due to the uncalibrated nature of our system. An example these stages applied sequentially to a camera in our system is shown in Fig. 2.

4.2.1 In-image obstacle expansion

Since the position of the detected robot in each image is determined by the centre of it's bounding box and we define a collision as the intersection of this bounding box with an obstacle, the non-drivable regions of the image need to be expanded before planning can take place. However, since the extrinsics and intrinsics of each camera are unknown, it is not immediately clear how large this expansion should be, despite full knowledge of the robot's physical

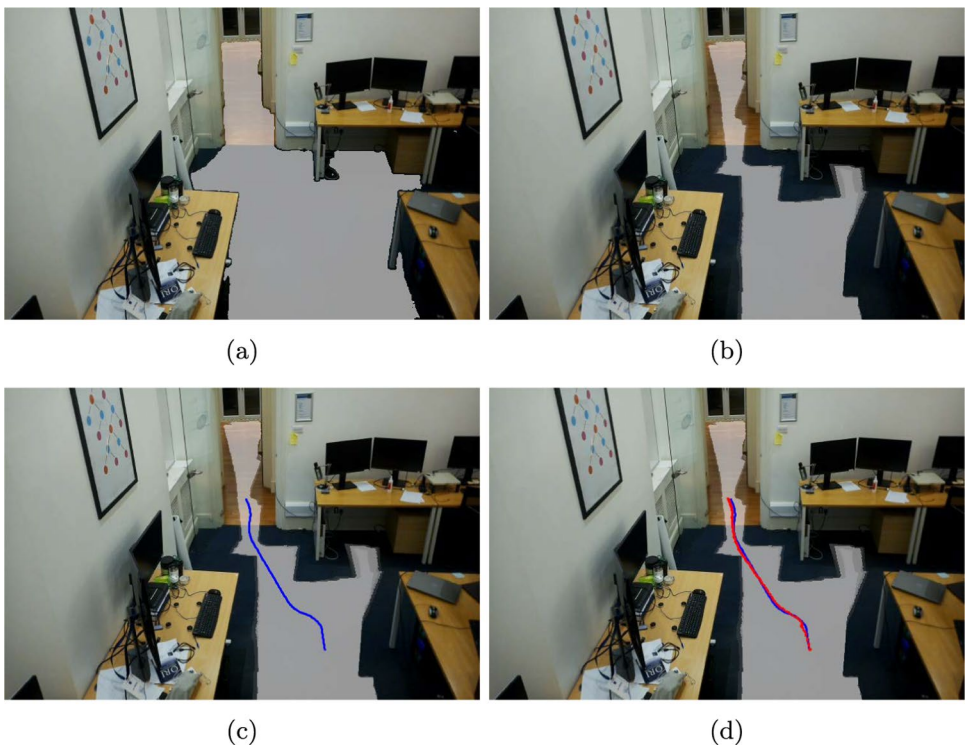


Fig. 2 **a** Drivable region segmented in camera 4 with SAM2 (Ravi et al., 2024) shown as the brighter floor. The drivable region is then shrunk according to the predicted robot bounding box **b**, with darker gray regions showing areas where the robot would collide in some orientations but not others. The hybrid A* planner then creates a path **c** shown here in blue and finally the custom pure pursuit controller directs the robot along this path **d** with detected robot positions shown here in red. Further details on the process can be found in Sect. 4.2

size. Furthermore, the amount of expansion needed will change depending on the robot's location due to the camera perspective.

To solve this issue, we train a simple multilayer perceptron to predict the bounding-box sizes for the robot from its pose in the image (x, y, θ). This network is trained using the same data used to train the orientation detector as discussed in Sect. 3.2. The fact that this was collected by performing long random walks across the entire drivable region makes it well-distributed across the images. Finally, this obstacle expansion is precomputed for each camera and stored as a look-up table to avoid adding extra computation to the low-level planner during operation.

4.2.2 Low-level path planning

To find a path between handover points in each image, we propose using a hybrid A* planner (Dolgov et al., 2008), which offers benefits over other grid-based planners by avoiding excessively sharp turns which challenge the controller. Five primitives are used for node expansion in the planner: three involve moving a specified distance (in pixels) either in a straight line or with a constant radius of turn in either direction; the last two involve keeping the robot stationary and rotating it on the spot by a specified angle in either direction. The cost of the first three primitives is given as the distance travelled in the image (with a slight penalty for turning), and the cost for the other two is left as a user-defined parameter. The reason for the addition of these two turn-on-the-spot primitives is so that a large radius of turn could be specified to ease control in the general case for open areas without sacrificing the ability to navigate through confined regions as well. Finally, an additional cost is added to the planner for each expanded node, proportional to the reciprocal of the distance to the nearest obstacle. This has the effect of increasing the overall safety of the system by encouraging the planner to keep a reasonable distance from obstacles.

4.2.3 Low-level control

Finally, after a path has been found, it is handed to a custom pure pursuit controller (Corke, 2017) which gives commands to the robot to follow the line in image space. Again, due to the perspective of the camera, following a point that moves at a constant speed in the image would not lead to the robot moving at a constant speed on the ground. Therefore, instead of having the robot follow a pursuit point moving at a steady speed along the path, we instruct the robot to move with constant linear velocity and calculate the pursuit point based on finding the closest path point to the current robot position and moving some short distance further along the path. This way, we do not have to convert between image and ground speed, and the robot is always following a point slightly ahead of it on the path as intended in the pure pursuit controller. When the robot reaches a point along the path were it must turn on the spot, it is instructed to stop and a PD controller adjusts the angular velocity until the robot is aligned with the new intended heading in the image.

4.3 High-level camera graph discovery

With the preliminaries established, we propose an approach to discover overlap regions among the cameras. To minimise manual intervention, we leverage the natural movement of people within the space, using simultaneous detections in camera pairs to determine viewpoint overlap. This method is advantageous as it capitalises on the daily activities of people moving through the environment.

During the graph-building process, we monitor the activity in each camera using a *YOLO* detector trained on people. When simultaneous detections occur across cameras, we employ *PersonViT* (Hu et al., 2024), a transformer-based model specifically trained for person identification, to describe each person with a feature vector. If the any two of the feature vectors are within a certain euclidean distance—feature distance—of each other, we conclude that they correspond to the same individual (Fig. 3).

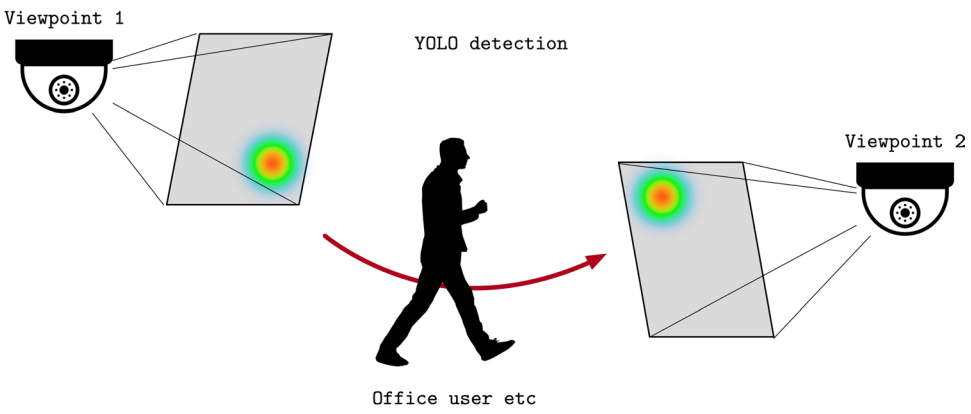


Fig. 3 Tracking individuals transitioning between camera frames by detecting them simultaneously in overlapping frames. This is achieved using YOLOv8 for bounding box detection and PersonViT for feature matching

Upon successfully matching detections, we establish a connection between the pixel location of the bottom middle of the bounding boxes of the detected individual in each camera, referred to as an *overlap point*. This approach allows us to utilise the positions of individuals, who may be significantly taller than the robot in use, to establish connections between regions on the ground plane.

Once pixel-to-pixel overlap points between two cameras have been detected in a cluster, we select the one positioned the farthest from its closest obstacle in either camera as the specific *handover point* between the two cameras. This handover point is saved as an edge in the camera handover graph, connecting the nodes representing the two camera views. Specifically, an edge tells the system that two cameras share an overlap region, and the handover point associated with the edge represents the pixel location the robot should navigate to in one camera for the other to see it and continue control. In this method, a graph of isolated nodes representing the cameras in the system is built up automatically over time to become more connected as people move through the space and areas of overlap are identified. Examples of these handover points can be seen in Fig. 4 where a robot in camera 7 can navigate to the point labelled *8* and be picked up by camera 8 to continue navigation. Further details of this graph creation procedure are detailed in algorithm 1.

Input: N synchronised camera frames $\{I_1, I_2, \ldots, I_N\}$, PersonViT feature-distance cut-off f_c
Output: Graph $G = (V, E)$ where V represents nodes (cameras) and E represents edges between nodes corresponding to handover points between the cameras.

$G \leftarrow$ empty graph with N nodes
while people using the space **do**
  for all $i \in [0, N)$ **do**
    $I_i \leftarrow$ sample camera i
    $\{B_i^1, B_i^2, \ldots, B_i^{P_i}\} \leftarrow \text{YOLO}(I_i)$
    for all $p \in [0, P_i)$ **do**
      $F_i^p \leftarrow \text{PersonViT}(B_i^p)$
    end for
  end for

  for all pairs (B_i^k, B_j^l) where cameras $i \neq j$ **do**
    if $\|F_i^k - F_j^l\|_2 < f_c$ **then**
      Insert E_{ij} into G
    end if
  end for
end while

Algorithm 1 Graph Construction from Camera Frames

(a) Camera 7 (b) Camera 8

Fig. 4 Handover regions are depicted as red markers and labelled with the relative camera ID they lead to. For example, the red '7' in camera 8's view corresponds in physical space to the red '8' in camera 7's view

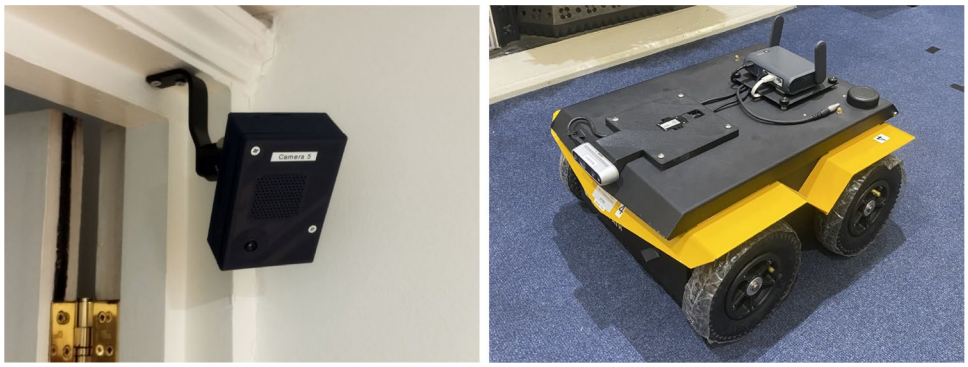
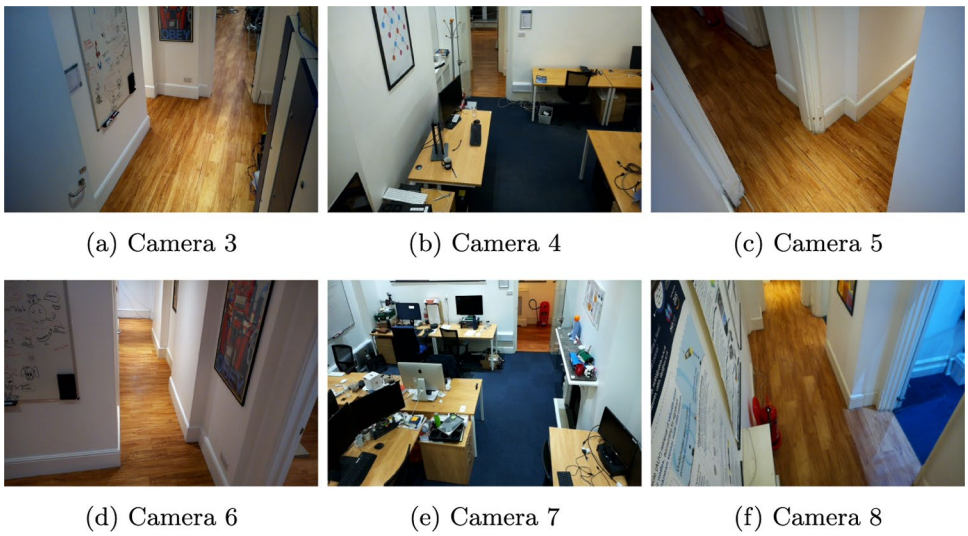


Fig. 5 Hardware employed for the experiments: (left) our custom camera and (right) the Jackal platform



(a) Camera 3 (b) Camera 4 (c) Camera 5

(d) Camera 6 (e) Camera 7 (f) Camera 8

Fig. 6 All six viewpoints as seen by the cameras. Noticeably, they show realistic scenarios with various objects and perspectives. We chose this experimental site to test our autonomous system in scenes with varied obstacles, textures, and lighting, all requiring dependable and adaptable performance

4.4 High-level planner and graph traversal

Through the camera handover graph, we can instruct a high-level planner to select the cameras needed to navigate from the current robot position to a final goal destination. To achieve this, the high-level planner is implemented using Dijkstra's algorithm, which directly operates on the camera graph and returns the shortest path from the original to the destination camera in terms of the number of handovers.

Control is then seamlessly passed from camera to camera at the corresponding overlap region in the planned path through the constellation of viewpoints. Control is performed by the camera's low-level controller and relinquished as soon as the robot approaches the next handover point within a certain distance threshold. At that point, the next camera will detect and drive the robot towards the next handover region or the target position if it is the final camera.

5 Experimental setup

To test our proposed system, we deployed it in an office space composed of two rooms and a corridor roughly 13 m long. We employ for our experiments a Clearpath Jackal,[1] with its sensors taken off and its onboard computers used only to drive the robot through actuator commands (forward speed and rotational speed), which themselves are computed offboard and sent through WiFi. This offboard autonomy stack runs on a single laptop with an Apple M2 Pro SoC. We designed a custom camera based on a Raspberry Pi 4 Model B and a Pi Camera Module 3, connected through Ethernet to the laptop. The robot and camera are shown in Fig. 5.

To provide sufficient autonomous coverage, we set up six remote cameras throughout the building floor, each overlapping with at least one other camera. Figure 6 shows the individual viewpoints for each camera, whereas the full experimental setup is depicted in Fig. 7, together with the extracted camera graph. Notably, the rooms have different textures, clutter, object types, and dimensions, *and* present

[1] https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/.

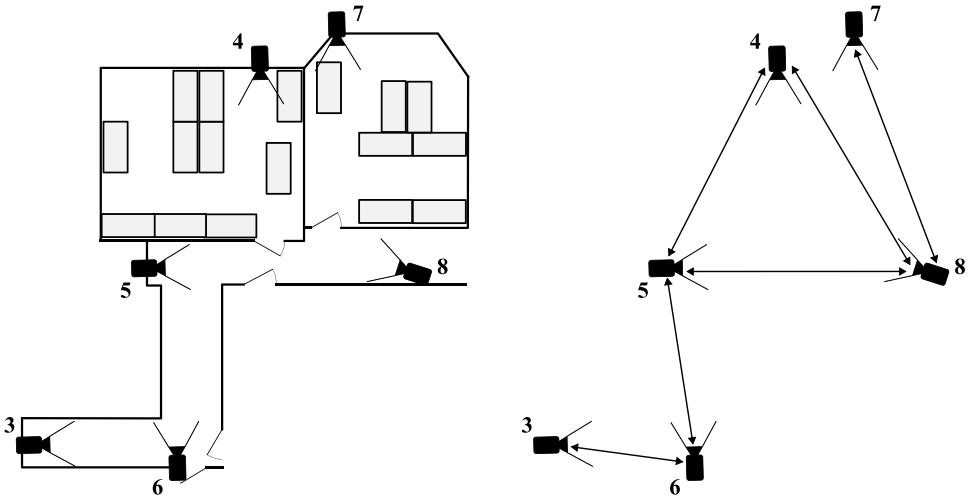


Fig. 7 Experimental setup (left) and camera handover graph (right). We installed six cameras in an office space, each with an overlapping field of view with at least another. These overlapping regions are shown in the handover graph, where each possible handover is depicted as an arrow

different perspectives, making this deployment scenario realistic.

We evaluate the system in two steps. Firstly, the graph creation mechanism described in Sect. 4.3 is run for ten minutes with three people using the office space. The resulting graph is then compared to the ground truth generated by inspection, and the robustness of the method investigated with a separate dataset of four different people traversing the environment for 10 min.

Having created a reliable camera graph, we evaluate the system by driving amongst the cameras for five runs, covering roughly 40 min. To do this, we generated seven random destination points covering the full graph, with the points in cameras `{7, 5, 3, 7, 4, 8, 6}`, which we repeated continuously.

For the traversals, we evaluated the system's ability to complete the path without hitting obstacles or getting stuck, as well as the time taken compared to a human, who drives the robot at the same maximum speed but looks directly at it to control its direction of travel.

Similarly, we tested the proposed system against a Simultaneous Localisation and Mapping (SLAM) baseline. For this experiment, we utilised the same Jackal, but equipped with a 2D LiDAR scanner, and relied on the open-sourced `nav2`[2] SLAM pipeline. We opted for a 2D scanner due to our requirement for cheap operations and, as such, for a more fair evaluation, and for the simplicity of the `nav2` library. Unfortunately, since the SLAM system was struggling localising the robot in the long corridor between cameras `5` and `6` due to its limited geometric features, we tested the approach in the subset created by cameras `{4, 5, 7, 8}`.

In each destination camera, we randomly selected a location in the drivable region as the destination point in that

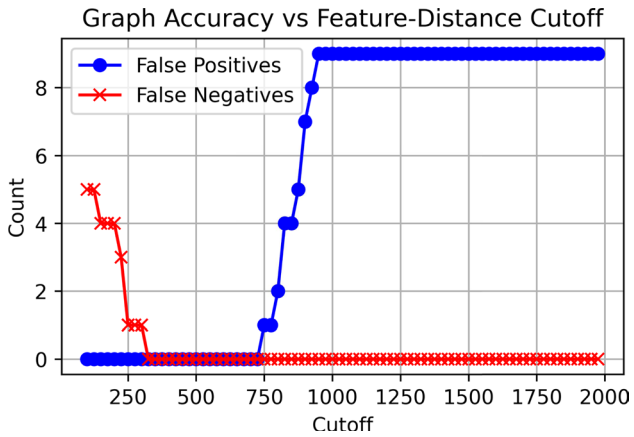


Fig. 8 Effect of feature distance cut-off on camera graph accuracy. We show the relationship between feature distance cut-off and graph accuracy against the ground truth. The red line indicates the number of edges present in the ground truth graph that were not detected by the proposed method, while the blue line represents the number of edges detected by the method that were not in the ground truth. The camera graph was constructed using a threshold of 5 handover points per cluster to be considered a positive edge

camera frame. We then placed the robot at this position, marked the outline of the robot on the floor and recorded the centre of the robot's bounding box. We required the robot to stop with its centre within this marked region to count a traversal as successful, measured as within five pixels from the recorded bounding box centre.

6 Experimental results

First, Sect. 6.1 analyses the accuracy of the camera graph construction before Sect. 6.2 investigates the results of the autonomous robot navigation.

[2] https://nav2.org/.

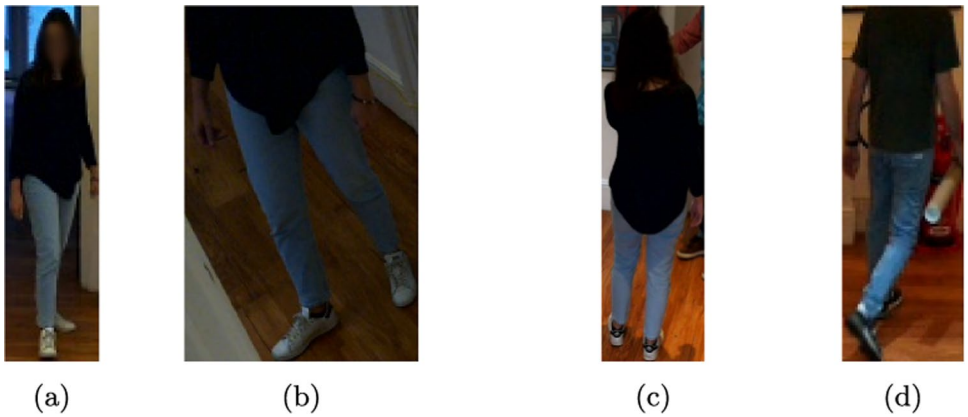


Fig. 9 Person matches at either end of the cut-off basin. Figures **a** and **b** show the person match needed to correctly identify the last edge in the graph. Figures **c** and **d** show the first incorrect person match, leading to an erroneous edge being added to the graph

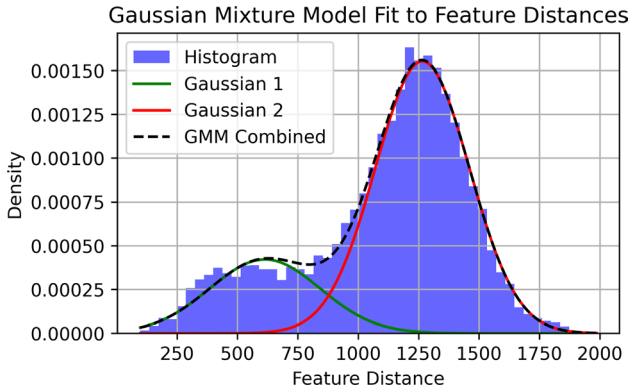


Fig. 10 Histogram of feature distances from patches surrounding people, illustrating the distribution of variations in the appearance of people using the workspaces in our dataset

6.1 Camera network generation via person tracking

To test our proposed method of graph creation detailed in Sect. 4.3, a group of three people were recorded using the office space for ten minutes by all of the cameras. This dataset consisted of people sitting and working at desks and walking through rooms and corridors. Further, when moving through the corridors, the subjects walked both on their own and in pairs as well as crossing paths. Pairs of handover points are then created using *PersonViT*. To remove spurious detections, we use a threshold of five overlap points before a cluster is considered to produce a graph edge.

Figure 8 shows how the accuracy of the graph against the manually created ground truth. We report the false positives and negatives at different feature distance cut-offs of the individual signature obtained through *PersonViT*. Here, the false-positive count defines how many edges were in the ground-truth graph but were not detected by our method and the false positives, instead, the number of edges detected using our method that were not in the ground truth. The plot shows a clear region between 325 and 700, where the cut-off obtains a perfect graph. Edges are missed for lower cut-off values, while erroneous additional edges are included for higher cut-off values. The maximum observed feature distance between two people is 1987, which gives a wide margin of error for tuning this cut-off parameter while still allowing for perfect graph construction.

We have inspected the image pairs at the margins of the cut-off basin in Fig. 9. The first pair shows the two images with the largest feature distance that needed to be correctly matched before the graph was perfectly reconstructed—i.e. the left-hand edge of the cut-off basin. This pair of images has an extreme crop, with one camera viewing the whole person and the other only seeing the bottom half. This suggests that the cut-off for the feature distance used to construct the graph must be large enough to account for large differences in crop experienced by some overlap regions, which artificially inflate the image pairs' feature distance due, e.g., to stark differences in perspective. Conversely, the second pair of images shows the first match that leads to an incorrect edge—i.e. the right-hand edge of the cut-off basin. These two images are, in fact, of separate people, but they are dressed very similarly, and it is clear to the observer that these images could be confused to be the same person.

Further insight on the performance of the person matching by *PersonViT* can be gained from Fig. 10, where the feature distance between every individual comparison has been visualised in a histogram. Since the comparisons will either be two separate views of the same person or of two different people, the data is expected to be bimodal, with clusters corresponding to positive matches of the same person or negative non-matches of different people. The data respect this hypothesis, and we fit a Gaussian Mixture Model (GMM) to validate it. However, the distribution better fits the mode with larger feature distances—i.e. the non-matches. This aligns with the previous analysis, as many of the positive person matches have substantially different crops (see Fig. 9a and b), which will artificially increase the distance between the *PersonViT* features of the matches. This has the effect of skewing the first mode to have a large right-hand tail that blends into the second's left-hand tail.

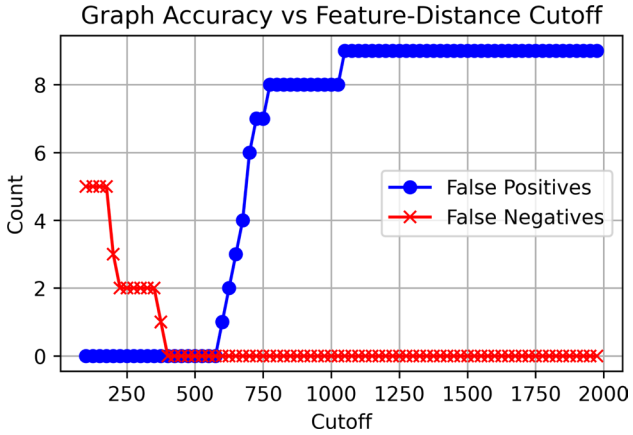


Fig. 11 Effect of feature distance cut-off on camera graph accuracy over the test dataset. The red line indicates the number of edges present in the ground truth graph that our method did not detect, and the blue line represents the edges detected by our method that were not in the ground truth. Notably, a region where both lines reach zero on this new dataset demonstrates that the method generalises well, achieving perfect alignment with the ground truth within this range and—critically—replicating the behaviour seen in the first dataset Fig. 8

Since graph building is part of the wider context of an autonomous robotic system, the failure modes should be carefully considered. Most notably, the existence of an erroneous edge in the camera graph is far worse than the absence of an edge that should exist. This is because an erroneous edge could lead to a collision of the robot if the system tries to hand over control to a camera that cannot see the robot due to an overlap assumption. Conversely, the absence of an edge has no greater effect than reducing the graph's connectivity and, in the worst-case scenario, losing the ability to reach certain regions. For this reason, we should heavily favour the sensitivity of our model to correctly predict person matches over specificity. Therefore, assuming that the rightmost mode, with a mean of 1265.30 and std. deviation of 195.08, represents non-matches and using a 0.01% chance of incorrectly classifying images of two different people as a match, gives a cut-off point of $1265.30 - 3.891 \times 195.08 = 506.2$. Referring back to Fig. 8, this cut-off value is positioned right in the middle of the cut-off basin and produces a perfect graph reconstruction.

To test the generalisation ability of our model, we recorded a separate dataset and reconstructed the graph with the same cut-off value. Due to practical limitations, we were forced to use the same camera locations as in our previous dataset but employed four new people, previously unseen by the system, to gather the data; moreover, the recording took place at a different time of day with intentionally different lighting conditions present. All other conditions for recording and analysing the dataset remained the same.

Figure 11 again shows how the accuracy of graph reconstruction varied with cut-off distance and, again, a cut-off value of 506.2, as previously calculated, produces a perfect graph reconstruction. Further, Fig. 12 shows the same two pairs of images as in Fig. 9 but for the new dataset. Observation of these images sheds light on why the cut-off basin is smaller in this new dataset. The more challenging lighting conditions make the first pair harder to match, and the similarity in appearance of two of the subjects makes the second pair harder to differentiate. However, even though the cut-off basin is slightly smaller, it is still centred around our previously determined cut-off point and therefore leads to a robust construction of the camera graph.

This shows that the Euclidean distance between *Person-ViT* feature vectors is a robust measure to determine if a person is being viewed from multiple cameras and can reliably produce perfect constructions of camera overlap graphs.

A final notable point of this approach is its efficiency against our previous work (Robinson et al., 2023). In fact, both the prior random-walk method and the proposed human-centric method achieved correct graph reconstruction; however, the effort required to build the graph differs significantly: the random walk involved recording approximately 20 min of data for each camera (approx. 2 h total); by contrast, the human-tracking method required only 15 min overall to create the graph. Whilst our current implementation still relies on random walks for training (and, thus, providing the camera graph for free), we anticipate that

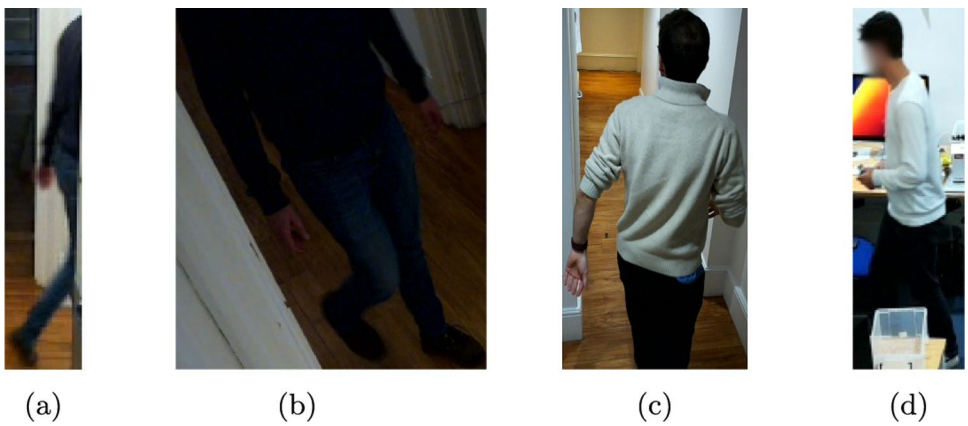


Fig. 12 Person matches at either end of the cut-off basin for the alternative dataset. Figures **a** and **b** show the person match needed to correctly identify the last edge in the graph. Figures **c** and **d** show the first incorrect person match, leading to an erroneous edge being added to the graph

Table 1 Time spent on each traversal in seconds

Run	7→5	5→3	3→7	7→4	4→8	8→6	6→7
1	55.5 (0.7)	48.5 (0.2)	115.6 (3.9)	72.8 (0.5)	37.5 (1.0)	47.2 (2.3)	87.6 (3.0)
2	56.0 (0.8)	48.9 (0.2)	112.4 (3.8)	74.9 (0.8)	35.7 (1.0)	47.0 (2.3)	93.0 (3.0)
3	56.5 (0.6)	48.9 (0.2)	109.4 (3.8)	75.1 (0.8)	33.9 (1.2)	47.5 (2.0)	92.3 (3.2)
4	56.2 (0.6)	51.6 (0.2)	106.1 (3.9)	73.7 (0.9)	36.7 (1.1)	47.8 (1.9)	90.9 (3.0)
5	54.8 (0.8)	50.9 (0.2)	107.8 (3.5)	73.4 (1.1)	33.8 (0.8)	47.5 (1.8)	87.6 (3.1)
Human	51.9	47.1	97.4	66.0	31.7	45.6	79.7

The values stated are the duration over which the robot was moving with values in parentheses being the total time spend planning while stationary. Therefore, the total time taken for each traversal would be the sum of these two values. The bottom row shows the time taken for an expert human operator to complete the traversal via manual remote control

Table 2 Time spent on each traversal in seconds, including all planning time, for both a traditional SLAM approach and our system. Values presented for each traversal are averages over successive runs

Run	7→5	7→4	4→8
`nav2`	55.7	69.6	35.7
Ours	56.5	74.8	36.5

future extensions with alternative detection methods might eliminate this requirement, making the efficiency gain of the human-centric approach very pronounced.

6.2 Autonomous robot navigation and control

Table 1 shows the time taken for each traversal that made up our 40 min continuous operation. Each row shows one whole “loop" of the environment with five loops completed during the experiment. The main times shown exclude the time the robot spent stationary planning paths during each traversal, with these extra durations being shown in parentheses for completeness. The bottom row shows the result of an expert human operator manually controlling the robot through the same traversals while directly observing it, limited to the same maximum velocity as the autonomous system. When comparing the times of our system and the human operator, the system’s efficiency can be seen–the autonomous control was, averaged over all runs and traversals, only 10.4% slower than the human controlled path (13.2% when planning time is also considered).

Finally, we compare our system with a SLAM-based approach implemented using the `nav2` navigation package. We employed the same evaluation procedure, but restricted it to the subgraph of cameras `4, 5, 7, 8`, since the SLAM method struggled to maintain localisation in the narrow corridor between cameras `5` and `6`. The results, shown in Table 2, indicate that our method performs comparably to classical SLAM, completing the traversal only 4% slower on average with planning time included. While we acknowledge that the navigation package used as our SLAM benchmark is a relatively simple example, this comparison highlights a strength of our system when compared to traditional methods. Without sacrificing speed or efficiency in navigation, the static nature of our sensors allows us to practically eliminate the difficulties of maintaining an accurate localisation in long and featureless spaces, such as the long corridor in this example, as the sensor needs only to detect the robot itself.

7 Conclusion and future work

We have demonstrated the effectiveness of learning camera handover points and the connectivity of a network of viewpoints from the natural use of a workspace by people, which holds great promise for facilitating online, incremental installation, and lifelong management of offboard control from constellations of remote viewpoints. Our experiments, involving a significant number of cameras, have showcased the system’s robust autonomy, underlining its potential for real-world deployment.

Specifically, we perfectly recovered a network of overlapping fields of views between 6 cameras. We showed that the method for doing so was generalisable across times of day and differing participants in the workspace. Importantly, the system for doing so was insensitive to tuning, providing a wide “basin” of perfect performance.

We further showed good autonomous capability across the network of cameras, where we repeatedly autonomously traversed the entire productive site including multiple connected workspaces which feature realistic clutter and hazards which the robot navigated around robustly. This is an important step towards not having to calibrate numerous cameras in a large workspace—saving significant time, reducing errors, minimising maintenance, and enhancing efficiency, making setup and ongoing operation much simpler and cost-effective.

Nevertheless, this approach is reliant on the positioning of the fixed cameras, which, if done manually by a human operator, might not cover the full working space effectively nor have ideal overlapping areas. For this reason, we will work on automated positioning of cameras to minimise the number of cameras needed for optimal robotic performance.

Moreover, while our person identification approach is robust for the presented cases, we expect it to face challenges

in more complex scenarios involving crowds or larger-scale sensor networks. Although works such as (Marinakis et al., 2005; Marinakis & Dudek, 2008) are not directly applicable to our setting—since they focus on inferring network topology only, rather than also overlap regions—we believe that statistical strategies inspired by these methods could enhance our system in such challenging conditions.

Similarly, in the future, we will explore selecting amongst overlapping viewpoints depending on robot controllability—where some typical motions are more or less observable depending on perspective—and tackle dynamic obstacles in the environment. Indeed, although initial work has been done in Buoso et al. (2025) exploiting a Visual-Language Model, this remains a challenging problem that prevents applicability in common operational spaces shared with human operators.

Acknowledgements We thank Pratik Somaya and Jared Bellingham for their help in configuring the experimental setup. We also thank Arianna Angelico, Luigi Campanaro, Matthew Holland, Nick Elkington and Tristan Johnston-Wood for their help recording the dataset used to create and test the camera graphs.

Author contributions All authors contributed to the conception and design of the initial work as well as reviewing the final manuscript. L.R., D.D.M. and M.G. wrote the initial draft and analysed and interpreted the experimental results discussed in Sects. 6.2 and 6.3. L.R. and D.D.M. were responsible for safely conducting and gathering data from all experiments as well as the analysis and interpretation of experimental results discussed in Sect. 6.1. L.R. and D.D. M. authored the software used to conduct experiments for this manuscript.

Funding The authors are affiliated with the University of Oxford, UK. Matthew Gadd and Paul Newman are currently also affiliated with Oxa Ltd. The work was jointly supported by the EPSRC Programme Grant "From Sensing to Collaboration" (EP/V000748/1) and the EPSRC Impact Acceleration Account (IAA) "Robotics Inversion".

Data availability Data generated from this work cannot be shared publicly to protect study participant privacy.

Declarations

Competing interests The authors declare no competing interests.

References

Batalin, M. A., Sukhatme, G. S., & Hattig, M. (2004). Mobile robot navigation using a sensor network. In *IEEE international conference on robotics and automation*, 2004. Proceedings. ICRA'04. 2004, vol. 1, (pp. 636–641). IEEE.

Buoso, D., Robinson, L., Averta, G., Torr, P., Franzmeyer, T., & De Martini, D. (2025). Select2plan: Training-free icl-based planning through vqa and memory retrieval. *IEEE Robotics and Automation Letters*.

Corke, P. (2017). Robotics, Vision and Control. Springer Tracts in Advanced Robotics, vol. 118. Springer. https://doi.org/10.1007/978-3-319-54413-7

Dolgov, D., Thrun, S., Montemerlo, M., & Diebel, J. (2008). Practical search techniques in path planning for autonomous driving. *Ann Arbor, 1001*(48105), 18–80.

Elsheikh, E. A., El-Bardini, M., & Fkirin, M. (2016). Practical design of a path following for a non-holonomic mobile robot based on a decentralized fuzzy logic controller and multiple cameras. *Arabian Journal for Science and Engineering, 41*, 3215–3229.

Esterle, L., Lewis, P. R., Yao, X., & Rinner, B. (2014). Socio-economic vision graph generation and handover in distributed smart camera networks. *ACM Transactions on Sensor Networks, 10*(2), 20–12024. https://doi.org/10.1145/2530001

Gheissari, N., Sebastian, T. B., & Hartley, R. (2006). Person reidentification using spatiotemporal appearance. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2, (pp. 1528–1535). IEEE.

Gray, D., & Tao, H. (2008). Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *Computer Vision–ECCV 2008: 10th European conference on computer vision*, Marseille, France, October 12-18, 2008, Proceedings, Part I 10, (pp. 262–275). Springer.

Hu, B., Wang, X., & Liu, W. (2024). Personvit: Large-scale self-supervised vision transformer for person re-identification. arXiv preprint arXiv:2408.05398

Javed, O., Shafique, K., Rasheed, Z., & Shah, M. (2008). Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views. *Computer Vision and Image Understanding, 109*(2), 146–162. https://doi.org/10.1016/j.cviu.2007.01.003

Kruse, E., & Wahl, F. M. (1998). Camera-based monitoring system for mobile robot guidance. In *Proceedings. 1998 IEEE/RSJ international conference on intelligent robots and systems. Innovations in theory, practice and applications* (Cat. No. 98CH36190), vol. 2, (pp. 1248–1253), IEEE.

Li, J., Xu, J., Zhong, F., Kong, X., Qiao, Y., & Wang, Y. (2020). Pose-assisted multi-camera collaboration for active object tracking. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, (pp. 759–766).

Marinakis, D., Dudek, G., & Fleet, D. J. (2005). Learning sensor network topology through monte carlo expectation maximization. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, (pp. 4581–4587). IEEE.

Marinakis, D., & Dudek, G. (2008). Occam's razor applied to network topology inference. *IEEE Transactions on Robotics, 24*(2), 293–306.

Poduri, S., & Sukhatme, G. S. (2004). Constrained coverage for mobile sensor networks. In *IEEE International conference on robotics and automation, proceedings*. ICRA'04. 2004, vol. 1, (pp. 165–171). IEEE.

Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K.V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P., &

Feichtenhofer, C. (2024). SAM 2: Segment Anything in Images and Videos arXiv:2408.00714

Rekleitis, I., & Dudek, G. (2005). Automated calibration of a camera sensor network. In *2005 IEEE/RSJ international conference on intelligent robots and systems*, (pp. 3384–3389). IEEE.

Rekleitis, I., Meger, D., & Dudek, G. (2006). Simultaneous planning, localization, and mapping in a camera sensor network. *Robotics and Autonomous Systems, 54*(11), 921–932.

Robinson, L., De Martini, D., Gadd, M., & Newman, P. (2023). Visual servoing on wheels: Robust robot orientation estimation in remote viewpoint control. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Robinson, L., Gadd, M., Newman, P., & Martini, D. D. (2023). Robot-relay: Building-wide, calibration-less visual servoing with learned sensor handover networks. In *International symposium on experimental robotics*, (pp. 129–140). Springer.

Ruiz, N., Chong, E., & Rehg, J. M. (2018). Fine-grained head pose estimation without keypoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, (pp. 2074–2083).

Shah, S. H. H., Steinnes, O.-M.H., Gustafsson, E. G., & Hameed, I. A. (2021). Multi-agent robot system to monitor and enforce physical distancing constraints in large areas to combat covid-19 and future pandemics. *Applied Sciences, 11*(16), 7200.

Shalom, A. B., & Rofe, A. (2022). Systems and methods for managing multiple autonomous vehicles. Google Patents. US Patent 11520344.

Shalom, A. B., & Rofe, A. (2022). Systems and methods for managing multiple autonomous vehicles. Google Patents. US Patent 11520344.

Simoens, P., Dragone, M., & Saffiotti, A. (2018). The internet of robotic things: A review of concept, added value and applications. *International Journal of Advanced Robotic Systems, 15*(1), 1729881418759424.

Varghese, R., & Sambath, M. (2024). Yolov8: A novel object detection algorithm with enhanced performance and robustness. In *2024 international conference on advances in data engineering and intelligent computing systems (ADICS)*, (pp. 1–6). https://doi.org/10.1109/ADICS58448.2024.10533619

Whitaker, T. J., Cunningham, S.-J., & Bobda, C. (2020). Decentralised indoor smart camera mapping and hierarchical navigation for autonomous ground vehicles. IET Computer Vision.

Ye, M., Shen, J., Lin, G., Xiang, T., Shao, L., & Hoi, S. C. H. (2022). Deep learning for person re-identification: A survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(6), 2872–2893. https://doi.org/10.1109/TPAMI.2021.3054775

Luke Robinson is a DPhil (PhD) candidate at the Oxford Robotics Institute having previously received an MEng degree in Engineering Science from the University of Oxford.

Matthew Gadd is a Staff Software Engineer at Oxa and Stipendiary Lecturer in Engineering Science at St Hilda's College and previously worked as a Senior Research Associate at the Oxford Robotics Institute. He holds a DPhil (PhD) in Engineering Science from the University of Oxford and a BSc in Mechatronics Engineering from the University of Cape Town.

Paul Newman is the Director of the Oxford Robotics Institute, and Founder of Oxa Ltd. He is also the BP Professor of Information Engineering at the University of Oxford.

Daniele De Martini (Member, IEEE) received a BSc degree in mechanical engineering from the University of Pavia, an MSc degree in mechatronic engineering from the Politecnico di Torino, and a PhD degree in robotics from the University of Pavia. He is currently an Associate Professor in robotics with the University of Oxford and a Tutorial Fellow in engineering science with Keble College.