Camera-based mapping in search-and-rescue via flying and ground robot teams

Esteves Henriques, Bernardo; Baglioni, Mirko; Jamshidnejad, Anahita

DOI
10.1007/s00138-024-01594-4

Publication date
2024

Document Version
Final published version

Published in
Machine Vision and Applications

Machine Vision and Applications (2024) 35:117
https://doi.org/10.1007/s00138-024-01594-4

RESEARCH

Camera-based mapping in search-and-rescue via flying and ground robot teams

Bernardo Esteves Henriques[1] · **Mirko Baglioni**[1] · **Anahita Jamshidnejad**[1]

Received: 29 January 2024 / Revised: 16 July 2024 / Accepted: 30 July 2024

Abstract
Search and rescue (SaR) is challenging, due to the unknown environmental situation after disasters occur. Robotics has become indispensable for precise mapping of the environment and for locating the victims. Combining flying and ground robots more effectively serves this purpose, due to their complementary features in terms of viewpoint and maneuvering. To this end, a novel, cost-effective framework for mapping unknown environments is introduced that leverages You Only Look Once and video streams transmitted by a ground and a flying robot. The integrated mapping approach is for performing three crucial SaR tasks: localizing the victims, i.e., determining their position in the environment and their body pose, tracking the moving victims, and providing a map of the ground elevation that assists both the ground robot and the SaR crew in navigating the SaR environment. In real-life experiments at the CyberZoo of the Delft University of Technology, the framework proved very effective and precise for all these tasks, particularly in occluded and complex environments.

Keywords Search and rescue robotics · Computer vision · Object detection · Pose estimation · Homography estimation · State estimation · Terrain mapping

1 Introduction

In recent decades, the number and severity of natural disasters have risen dramatically. Between 1991 and 2005, nearly 90% of disaster-related deaths and 98% of people affected by disasters belonged to low-income countries [1]. The survival rate of trapped victims drops from 91% in the first 30 min to 36.7% by the end of the second day [2]. Therefore, it is vital to make search-and-rescue (SaR) operations both affordable and time-efficient. SaR has become increasingly augmented with robotics in the last 20 years [3]. Besides traversing hazardous environments via their sensors, robots scan their surroundings rapidly and autonomously. In this paper, we introduce a novel, cost-effective framework for mapping unknown environments via SaR robots that effectively and efficiently combines the strengths of ground and flying robots when teaming up for SaR missions (Fig. 1).

1.1 Motivations

Table 1 shows a qualitative cost comparison for conventional sensors used in SaR robotics. More expensive sensing approaches, e.g., thermal imaging, may still fail to detect humans in high-temperature environments, e.g., in case of fire [4, 5]. The main disadvantage of radar and LiDAR sensors is their very high costs. In addition, the applications of RGB images are wider, i.e., while radar and LiDAR provide range information that can be used for obstacle avoidance or for mapping, images taken by RGB cameras can be used for computer vision tasks, including detection and tracking of objects and SaR victims, or extraction of information for training machine learning algorithms and neural networks [6–8].

Special sensors, e.g., Doppler-shift sensors for detecting humans based on the motion of their lungs, their heartbeat, or typical Doppler signatures of motion may fail in case of stationary targets or in distinguishing humans from other moving objects [9–11].

Acoustic sensors in SaR applications can be subject to various disturbances and noise from the environment. Flying robots are specially impacted by this issue, because of the noise that their propellers create [12].

✉ Anahita Jamshidnejad
a.jamshidnejad@tudelft.nl

[1] Control and Operations Department, Delft University of Technology, Kluyverweg 1, 2629 HS Delft, The Netherlands

Published online: 20 August 2024

Table 1 A qualitative cost comparison of SaR medium-segment sensors based on [6–8, 13–17]

Sensor type	Cost
RGB cameras, depth cameras	Low
Doppler-shift sensors, acoustic sensors	Low
2D radar, 2D LiDAR, thermal imaging	Medium
3D radar, 3D LiDAR	High

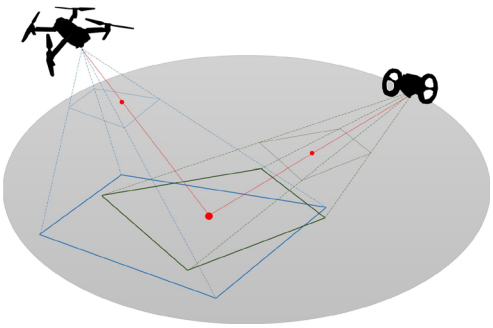

Fig. 1 Combining a flying and a ground robot with their different specifications and area coverage for mapping unknown environments

A main motivation for using vision-based algorithms that rely on inputs from standard RGB cameras is the affordability of these cameras (see Table 1) and the capability of such algorithms to perform well in very challenging SaR conditions, thanks to the recent advancements in image processing. In fact, an improved performance with respect to the state-of-the-art is achieved by incorporating the latest advancements in computer vision into the proposed framework. In particular, You Only Look Once (YOLO), which is used in this paper, has emerged as one of the most promising object detection algorithms and has proven to achieve real-time object detection with high accuracy levels and reduced computational resources.

Our main motivations for teaming up flying and ground robots are the following: On the one hand, with low costs, flying robots provide access to aerial perspectives and are able to swiftly cover expansive areas in a short time. Their imaging quality, however, may be compromised due to the tilting of their cameras while flying. Moreover, they may fail to access and perform properly in very confined spaces and corridors. On the other hand, while ground robots navigate rugged terrains at a slower pace, with a more restricted field of view, they excel in precise imaging, accessing confined spaces, and transporting heavy payloads. Moreover, as it is illustrated in Fig. 2, the distinct viewpoints that are captured by flying and ground robots can properly complement each other.

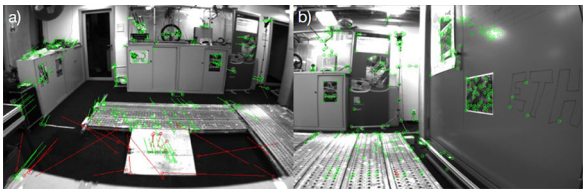

Fig. 2 Different viewpoints for a flying (left) and a ground (right) robot, retrieved from [18]

1.2 Background

The majority of the state-of-the-art literature that considers collaborative teams of flying and ground robots focuses on the problem of navigation of these robots in SaR environments (see, e.g., [18, 19]). An important problem in SaR missions, however, is to map the unknown environment, and to detect the individuals who are in distress and to estimate their position and pose [20].

The only papers that have focused on the simultaneous detection and localization of objects in SaR scenarios, using YOLO, include [21, 22]. Authors in [21] use the Scale-Invariant Feature Transform key point matching algorithm in order to determine the corresponding points in the pictures that have been taken from different viewpoints. Afterwards, using a homography matrix, the coordinates are transformed from one frame to the other. The approach used in [22] differs in determining the corresponding points within distinct frames, where trigonometry is used.

With recent advances in deep learning, victim detection through image processing has gained increased attention. Convolutional neural networks (CNNs) play a vital role in such detection methods [23–25]. In particular, the CNN YOLO has proven to be very effective in object detection [26]. For instance, in [21] pre-trained deep learning models based on YOLO are used, in order to detect objects in a flooded area. Authors in [22] apply YOLOv5 for detecting and localizing the victims in an outdoor SaR environment. In [27], YOLOv4-tiny (a compact version of YOLOv4) is used to detect the victims and their poses.

Moving victims should also be tracked in order to save them in a timely way. Thus, state estimation methods, e.g., the Kalman filter (and more advanced versions including the extended and unscented Kalman filter), have been proposed to estimate the trajectory of moving victims [28, 29]. Another crucial piece of information in SaR is the traversability and elevation of the ground of the terrain. Image processing can be used for such applications as well. In particular, authors in [30] present an approach for extracting the ground characteristics (e.g., the roughness, slope, discontinuity, hardness) from images using neural networks.

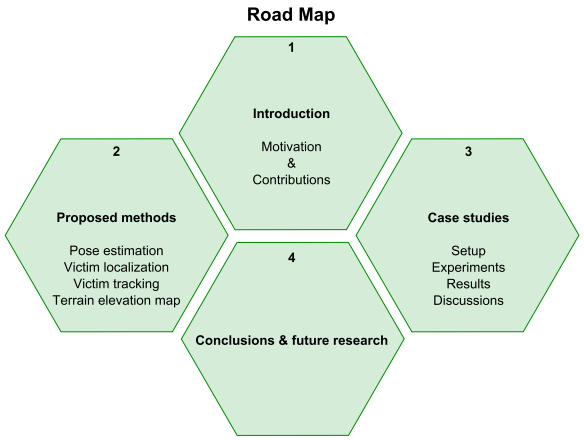

Fig. 3 Road map of the paper

1.3 Main contributions and structure of the paper

We propose new time and cost-efficient approaches for mapping unknown SaR environments, based on the fusion of the knowledge that is deduced from images that are taken separately, via autonomous flying and ground robots. The mapping includes localization of and trajectory tracking for victims, and estimating the ground elevation. The main contributions of this paper are:

1. We extend the pose estimation module of YOLO, in order to estimate the coordinates of the unobserved key points of the body of a victim from an image, based on body proportions and symmetry. Accordingly, YOLO is leveraged to estimate the distance of the victim from the ground robot.
2. YOLO is employed in streams captured from a flying and a ground robot, along with a localization algorithm, in order to map the detected points into real-world coordinates, and to fuse them using a Kalman filter, in order to estimate and track the trajectory of a moving victim.
3. An algorithm is proposed for estimating the elevation of the terrain, using YOLO and homography estimation.

Real-life case studies have been designed and performed in order to validate the three approaches explained above.

The rest of the paper is organized as follows. Section 2 explains the proposed methodologies. Section 3 describes the setup and implementation of the case studies that have been carefully designed to validate the developed approaches using real-world data. Then the results of the case studies are presented and discussed. Finally, Sect. 4 concludes the paper and provides suggestions for relevant future research (Fig. 3).

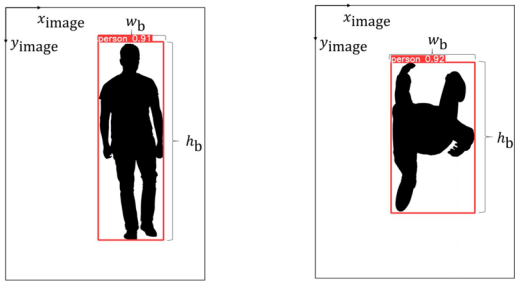

Fig. 4 YOLOv8n output for front (left-hand plot) and top (right-hand plot) views of a victim

2 Proposed methods

This section is structured in four parts, each explaining one SaR task. The first two parts will address the pose estimation and the victim localization. The third part will address the victim tracking approach using a Kalman filter. The last part will address the terrain elevation mapping. These four tasks are achieved by teaming up flying and ground robots.

Our methods are based on YOLOv8n [31], since it significantly speeds up the class detection process and maintains high accuracy. In a single pass through a neural network, YOLO identifies both the bounding boxes and the probabilities for objects to belong to specific given classes. In particular, YOLOv8n (i.e., the nano version model) is chosen, since it is faster than other models. For training, the Common Objects in COntext (COCO) dataset, comprising 330,000 diverse images, has been used. From these images, 200,000 are annotated for object detection, segmentation, and captioning tasks across 80 object categories. Annotations encompass bounding boxes, segmentation masks, and captions, that enhance the precision and versatility of the model.

Figure 4 shows the output of YOLOv8n for the front and top views of a victim. The bounding boxes are drawn around the detected object, and a confidence score from [0, 1] is attributed to each class for that object.

2.1 Pose estimation

This section explains how the pose estimation of the victims works. In order to precisely locate the victims and to estimate their (health/physical) status, awareness about the body pose of the victim is needed. YOLOv8n-Pose has been trained for pose estimation on a COCO dataset, which contains 200,000 images labeled with 17 key body points [31]. Figure 5 shows the output of YOLOv8n-Pose for the front view of a victim. The body key points, given in a pre-defined order, include: nose, eyes, ears, shoulders, elbows, hands, hips (the two side points), knees, and feet. These labels are recorded in a list, which we call list "L". Some of these key

Algorithm 1 Determining the position of all the missing key points

1: L: An ordered list of n^{keypoint} key point labels (i.e., nose, eyes, ears, shoulders, elbows, hands, hips, knees, feet)
2: K: An ordered list, with the same size and order as L, including the positions corresponding to the key points that are detected within a captured image, with $K[i]$ the i^{th} coordinate pair in list K
3: $n^{\text{iteration}}$: Pre-set number of iterations of the algorithm
4: i and k: Index variables
5: estimate_keypoint$(\cdot,\cdot)$: A function that receives as input the list K and the index i corresponding to an element in K that is empty, and assigns as output a pair of coordinates to fill in the empty element $K[i]$

6: **for** $k \leftarrow 1$ **to** $n^{\text{iteration}}$ **do**
7: **for** $i \leftarrow 1$ **to** n^{keypoint} **do**
8: **if** $K[i]$ is **None then**
9: $K[i] \leftarrow$ estimate_keypoint(K, i)
10: **end if**
11: **end for**
12: **end for**

13: **return** K

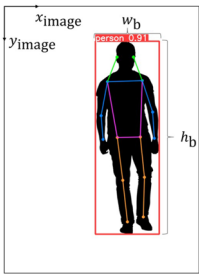

Fig. 5 YOLOv8n-Pose is extended to perform pose estimation, where based on the previous output of the algorithm, the positions of the body key points are estimated

points may not be visible in or detectable from the images that are received from a SaR robot, while their position is important for the estimation of the pose and distance of the victim from the robot. Therefore, YOLOv8n-Pose was extended to locate the missing key points, in order to make the pose estimation module robust to occlusions. Figure 6 showcases the output of the extended model whenever some body key points are occluded.

Algorithms 1 and 2 have been developed to estimate the position (i.e., the x and y coordinates) of the key points that are missing from an image: Algorithms 1 (lines 7–11) loops through the entire list K of the positions of the key points, that has initially been deduced from an image. This loop assesses whether or not any positions are missing within list K. For those key points that have not been detected by YOLOv8 in the image, a function `estimate_keypoint`$(\cdot,\cdot)$ has been developed (the details are given in Algorithm 2) in order to determine the position of that missing key point and to fill in the empty element in list K. Note that since Algorithm 2 runs within Algorithm 1, we have not repeated, in Algorithm 2, the same definitions that are used in both algorithms.

Function `estimate_keypoint`$(\cdot,\cdot)$ (see Algorithm 2) receives K and the index i of the element that is missing from K, and determines the position of the missing key points. The

Algorithm 2 Function `estimate_keypoint`$(\cdot,\cdot)$

Input: K, i
1: $n^{\text{dependency}}[i]$: Number of all sets of dependencies that have been defined for key point $K[i]$
2: R$[i]$: An ordered list composed of $n^{\text{dependency}}[i]$ sets, each including one set of the dependencies of key point $K[i]$
3: W$[i]$: A list composed of $n^{\text{dependency}}[i]$ sets of weights for the dependency key points, where the order of the sets and the order of the elements within each set are the same as in R$[i]$
4: W$[i, j, \ell]$: The weight that is deduced from the ℓ^{th} element within the j^{th} set of W$[i]$
5: D$[i, j, \ell]$: The position of the ℓ^{th} key point within the j^{th} set for list R$[i]$ where this position is deduced from list K
6: $n^{\text{set}}[i, j]$: Number of the key points within the j^{th} set of the dependencies of key point $K[i]$
7: j: An index variable
8: w: An auxiliary variable
9: b: A binary variable

10: $K[i] \leftarrow (0, 0)$, $w \leftarrow 0$, $j \leftarrow 1$
11: **while** $j < n^{\text{dependency}}[i]$ **do**
12: $b \leftarrow$ True
13: **for** $\ell \leftarrow 1$ **to** $n^{\text{set}}[i, j]$ **do**
14: **if** D$[i, j, \ell]$ is None **then**
15: $b \leftarrow$ False
16: **break**
17: **else**
18: $K[i] = K[i] + W[i, j, \ell] \cdot D[i, j, \ell]$
19: $w = w + W[i, j, \ell]$
20: **end if**
21: **end for**
22: **if** b **then**
23: **break**
24: **end if**
25: $j \leftarrow j + 1$
26: **end while**

27: **if** $j = n^{\text{set}}[i, j]$ **then**
28: **return** "CANNOT LOCATE MISSING KEY POINT!!"
29: **else**
30: $K[i] \leftarrow K[i]/\omega$
31: **end if**
32: **return** $K[i]$

algorithm needs the position of the other key points that the position of a missing key point directly depends on. Those key points are called the "dependencies" of the key point. For instance, using the body proportions, the position of the missing key point, the right elbow, can be estimated based on the positions of the dependency key points, the right hand and right shoulder. Alternatively, using the body symmetry the position of the right elbow may be estimated using the positions of the two shoulders and the left elbow. Thus, for each key point, more than one list of dependencies may exist (e.g., in the given example both sets {right hand; right shoulder} and {right shoulder; left shoulder; left elbow} belong to the set of dependencies of Key point "right elbow").

Thus, the dependencies for all the key points in list L (which includes all the n^{keypoint} key point labels) are defined a-priori in n^{keypoint} lists called "R$[r]$", which, for

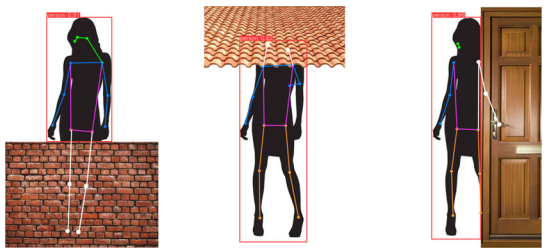

Fig. 6 Extended YOLOv8n-Pose for locating the missing key points (shown in white) based on typical body proportions and symmetry

$r \in \{1, \ldots, n^{\text{keypoint}}\}$, is composed of $n^{\text{dependency}}[r]$ sets of dependencies for key point r in list L. Moreover, a list "W[r]" with a similar number of sets and elements within each set as in R[r] is pre-defined and includes the weights that the algorithm associates with each corresponding dependency key point within R[r] (see lines 1–3 in Algorithm 2). For the given example, suppose that the order of the right elbow in list L is 8. Then, as an example, R[8] = { {right hand; right shoulder}, {right shoulder; left shoulder; left elbow} } and W[8] = { {0.5; 0.5}, {0.2; 0.4; 0.4} }. These weights are used to compute the position of a missing key point, based on a weighted average of the positions of the dependencies of that missing key points.

Note that the procedure of determining the positions of the missing points should better be run in more than only one iteration. This is because the positions that have been determined per iteration for the missing key points may be the positions of the dependency key points of other missing key points that could not be positioned in that iteration. Thus, the next iterations will help to position those key points as well. This is why in line 6 of Algorithm 1, a loop of $n^{\text{iteration}}$ iterations has been designed. In the real-life experiments in the CyberZoo, the number of the iterations, $n^{\text{iteration}}$, was set to 10, which was deemed sufficient to determine all the missing key points.

Note that when a list or parameter is always fixed, we have used a regular font for the corresponding notations, whereas for varying lists and for other variables we have used an italic font.

Remark 1 For unconventional body poses (e.g., when one arm is stretched and the other arm is bent), for symmetrical parts of the body (e.g., for the left and right arms), the algorithm considers the most reliable estimate of the position of one of the two parts (e.g., the estimate for the right arm) from the image. For the other part (in this case the left arm), a virtual symmetrical body part is considered and the coordinates are estimated accordingly (similarly to the right-hand side plot in Fig. 6). Thus, whether or not a human has a conventional pose does not impact the performance of the algorithms.

2.2 Victim localization

In this part of the methodology, determining the position of the victims from the ground and aerial images is explained. Localizing the victims autonomously via robots requires homography and distance estimation, as explained next.

Homography estimation

An indispensable method for mapping image pixels into real-world coordinates, especially when dealing with planar surfaces, is homography estimation. In our research, homography estimation serves as a valuable auxiliary method that facilitates the transformation of 2D image points into their corresponding real-world coordinates relative to the robot platforms. The homography transformation of 2D points is performed using a 3× 3 homography matrix $\mathbf{H}$, given by:

$$\mathbf{H} = \begin{bmatrix} \tilde{h}_{11} & \tilde{h}_{12} & \tilde{h}_{13} \\ \tilde{h}_{21} & \tilde{h}_{22} & \tilde{h}_{23} \\ \tilde{h}_{31} & \tilde{h}_{32} & 1 \end{bmatrix}$$

which implies that 8 parameters (3 rotational, 3 translational, 2 scalars) should be identified, requiring a system of 8 equations. These equations are generated by choosing 4 reference points, for which the corresponding x and y coordinates in both the camera view and the augmented view are known. Consider a set $P = \{(x_i, y_i) \text{ for } i = 1, \ldots, n^{\mathbf{H}}\}$ of $n^{\mathbf{H}}$ points co-planar in the camera view, and the corresponding set in the augmented view, i.e., $P'' = \{(x_i'', y_i'') \text{ for } i = 1, \ldots, n^{\mathbf{H}}\}$. The points within set P may be selected arbitrarily. However, considering points that are more distant from one another will help with the precision of the approach that will be explained next. This is because by selecting points that are farther from one another, a larger portion of the image frame is considered in the calibration. Furthermore, such a selection helps the algorithm to be more robust to errors in the estimation of one or a few of the reference points. Thus, the most efficient approach is to select the four corners of the image frame. For translation of the positions of these points into their corresponding real-world positions, a reference object may be placed at a known position with respect to the global reference frame, such that the object appears in the corner of the image. This procedure can be done in a laboratory before deployment in a real SaR setting because the calibration refers to the robot and to the orientation of its camera, and not to any external factors. Therefore, changing the x and y coordinates of the robot position or the yaw angle of the robot will not affect the calibration. Only a change in the tilt angle would require a different calibration of the system.

The following direct linear transform is used to obtain 2 linear equations per point:

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i'' & -y_i x_i'' & -x_i'' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i'' & -y_i y_i'' & -y_i'' \end{bmatrix}.$$

$$\begin{bmatrix}\tilde{h}_{11} & \tilde{h}_{12} & \dots & \tilde{h}_{32} & 1\end{bmatrix}^\top = \mathbf{0} \quad (1)$$

Alternatively, with more than 4 reference points the estimation of the homography matrix $\mathbf{H}$ transforms into an optimization problem that should determine an optimal homography matrix $\mathbf{H}^*$ that minimizes a defined cost function, which typically includes the algebraic sum, across all reference points, of the geometric distances between the projected points $(\tilde{x}_i'', \tilde{y}_i'')$ obtained using the candidate $\mathbf{H}$, and the actual corresponding points (x_i'', y_i'') in the map view.

Given $\mathbf{H}$, a point (x_i, y_i) in the camera view is transformed into (x_i'', y_i'') in the augmented view, using the following equation from [21]:

$$\begin{bmatrix}x_i''/\lambda & y_i''/\lambda & \lambda\end{bmatrix}^\top = \mathbf{H}\begin{bmatrix}x_i & y_i & 1\end{bmatrix}^\top \quad (2)$$

with λ a scaling factor that ensures the transformed points maintain their relative positions after the transformation.

Distance estimation

Knowing the distance of a detected victim from the ground robot is crucial in SaR. Under a simplified pinhole camera model, the distance d of a detected victim from the camera of the ground robot, is given via:

$$d = \frac{s^{\text{real}} l^{\text{focal}}}{s^{\text{image}}} \quad (3)$$

with s^{real} the real-world size of the victim (where knowing the position of the victim allows the robot to have a more realistic estimation of s^{real} for the victim), l^{focal} the focal length of the camera, and s^{image} the size of the victim in the image. It is assumed that the camera is calibrated. Note that l^{focal} is usually provided by the manufacturer, or is alternatively computed from the height H^{image} and width W^{image} of the image in pixels, and the horizontal FOV_h and vertical FOV_v fields of view of the camera (see Fig. 7). We have:

$$l^{\text{focal}} = \frac{H^{\text{image}}}{2\tan\left(\frac{\text{FOV}_\text{v}}{2}\right)} = \frac{W^{\text{image}}}{2\tan\left(\frac{\text{FOV}_\text{h}}{2}\right)}$$

Two sources of position measurement are obtained from the ground and the flying robot (see Fig. 8). Each measurement is computed by vector summation of the position of the robot and the position of the victim relative to that robot. The position of the robots is generally measured via a position-determination system, e.g., via GPS. The position of the victim relative to each robot can be computed using (2) and (3).

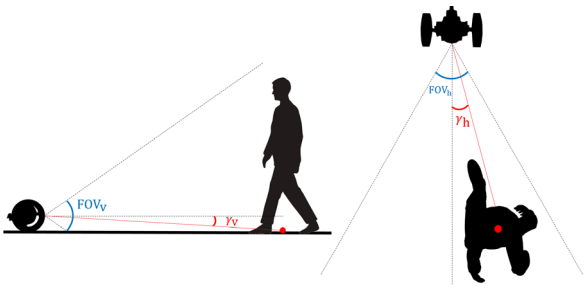

Fig. 7 The vertical (left) and the horizontal (right) FOVs of a camera, where γ represents the angle between the camera and the victim, with a subscript 'v' and 'h' for, respectively, the vertical and the horizontal cases

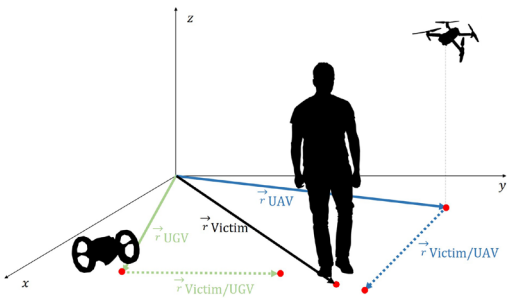

Fig. 8 Victim location per time step: Vector sum of the (flying or ground) robot position, and the relative position of the victim and the robot

2.3 Victim tracking

In this part of the methodology, we explain in detail how the trajectory of a moving victim is tracked through the images that are taken by the ground and aerial robots.

State estimation is a fundamental requirement in several SaR contexts since it enables one to make inferences about the states of a dynamic system based on noisy sensor measurements. Our primary objective of state estimation is to infer the trajectory of a moving victim after being detected. A Kalman filter, using a model of the victim motion, is employed to fuse the measurements of the two robots. Such models of the victim motion in disaster situations are available in the literature (see, e.g., [32]). By incorporating knowledge of how the victim's trajectory is expected to evolve over time, the Kalman filter makes predictions about the future states of the victim and generates a continuous and coherent trajectory according to the model. The state estimation approach is explained next.

State transition model

The state transition model describes how the states of a system evolve over time. In the discrete time, the evolution of the state vector, $\boldsymbol{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^\top$, including the position and the velocity of the victim in a 2D space at time step k, is via the following state-space equation, which is a general form of a dynamic model for the victim motion that has

been simplified via linearization and excluding the influence of external forces:

$$\boldsymbol{x}_{k+1} = \mathbf{F}\boldsymbol{x}_k \tag{4}$$

with $\mathbf{F}$ the state transition matrix. In real life, the actual state $\boldsymbol{x}_k$ may be unavailable, and thus estimated sequentially via (4). A hat symbol is used to represent the estimated states.

Observation model

The observation model relates the noisy or uncertain measurements $\boldsymbol{x}_k^{\mathrm{m}} = [x_k^{\mathrm{m}}, y_k^{\mathrm{m}}, \dot{x}_k^{\mathrm{m}}, \dot{y}_k^{\mathrm{m}}]^\top$ obtained from the sensors of the robots to the actual states of the system per time step via:

$$\boldsymbol{x}_k^{\mathrm{m}} = \mathcal{O}\boldsymbol{x}_k \tag{5}$$

with $\mathcal{O}$the observation matrix. The Kalman filter operates in two main steps, prediction and update, explained next. The superscript $^-$ for a variable indicates the variable before being updated.

Prediction step

At time step k, the most recent estimated state $\hat{\boldsymbol{x}}_k^-$ is updated (details are given next) to $\hat{\boldsymbol{x}}_k$ and is then used by the Kalman filter to estimate the future state of the system, i.e.:

$$\hat{\boldsymbol{x}}_{k+1}^- = \mathbf{F}\hat{\boldsymbol{x}}_k \tag{6}$$

The error covariance matrix $\mathbf{P}_{k+1}^-$ (which indicates how uncertain the state prediction is) is estimated by propagating the current updated error covariance matrix, $\mathbf{P}_k$, through the state transition model and by adding the noise matrix, $\mathbf{Q}$, i.e.:

$$\mathbf{P}_{k+1}^- = \mathbf{F}\mathbf{P}_k\mathbf{F}^\top + \mathbf{Q} \tag{7}$$

Update step

At time step $k+1$, when new measurements are obtained, the state predicted at time step k for time step $k+1$ is updated to improve the certainty. First, the Kalman gain $\mathbf{K}_{k+1}$ is obtained, based on the error covariance prediction matrix, $\mathbf{P}_{k+1}^-$, the observation matrix, $\mathcal{O}$, and the measurement noise covariance matrix, $\mathbf{R}$, via:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^-\mathcal{O}^\top(\mathcal{O}P_{k+1}^-\mathcal{O}^\top + \mathbf{R})^{-1} \tag{8}$$

The Kalman gain reflects the relative importance of the prediction and the measurements.

Next, the state $\hat{\boldsymbol{x}}_{k+1}^-$ predicted at time step k via (6) is updated using the Kalman gain and the difference of the actual measurements obtained from the sensors at time step $k+1$ and the predicted value for the measurements using $\hat{\boldsymbol{x}}_{k+1}^-$ and (5), i.e.:

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_{k+1}^- + \mathbf{K}_{k+1}(\boldsymbol{x}_{k+1}^{\mathrm{m}} - \mathcal{O}\hat{\boldsymbol{x}}_{k+1}^-) \tag{9}$$

Similarly, the error covariance matrix is updated at time step $k+1$ to incorporate the reduced uncertainty. We have:

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathcal{O})\mathbf{P}_{k+1}^- \tag{10}$$

Smoothening filter

In order to refine and smoothen the output of the Kalman filter, an exponential moving average filter is employed that assigns exponentially decreasing weights to older estimations. This allows the filter to adapt more quickly to recent changes in the measurements, while incorporating past information. The smoothening for time step k is performed via:

$$\hat{\boldsymbol{x}}_k^{\mathrm{EMA}} = \alpha\hat{\boldsymbol{x}}_k + (1-\alpha)\hat{\boldsymbol{x}}_{k-1}^{\mathrm{EMA}} \tag{11}$$

where $\hat{\boldsymbol{x}}_k^{\mathrm{EMA}}$ is the smoothened state estimate and α is the smoothening factor of the exponential moving average filter. Larger values for α result in higher weights for recent changes.

2.4 Terrain elevation map

In the last part of the methodology, we explain how the ground and aerial robots will collaboratively determine the ground elevation of the terrain.

First, the flying robot carries and drops a standard object with a known shape and size on a point where the terrain elevation is to be estimated. Then, the distance of the ground robot from this object is sent via the flying robot (by leveraging homography estimation and trigonometry) to the ground robot. Alternatively, the ground robot estimates its relative distance to the object using (3). This distance, together with the coordinates of the ground robot, is used by the robot to determine the expected position (blue point in Fig. 9) of the object, assuming a flat ground. Then, from the image the camera of the ground robot captures, the real point (shown in red in Fig. 9) that the object touches the ground. The pixel distance between these two points is used for the estimation of the ground elevation.

When the object sits close to the ground robot or at a high elevation, the object is perceived as smaller, and thus, the distance is perceived as larger than it is (see Fig. 10). This impacts the estimated terrain elevation. To mitigate this, depending on the geometry of the object, it is possible to take its width as a reference, since the width is only distorted in extreme scenarios.

Remark 2 The approach introduced in this section for terrain elevation mapping is mainly meant to be performed randomly, on a large scale, within the first stages of mapping the environment. In other words, a group of drones will be deployed to drop standard objects, not necessarily tennis balls, but, for instance, more flexible objects with possibly an

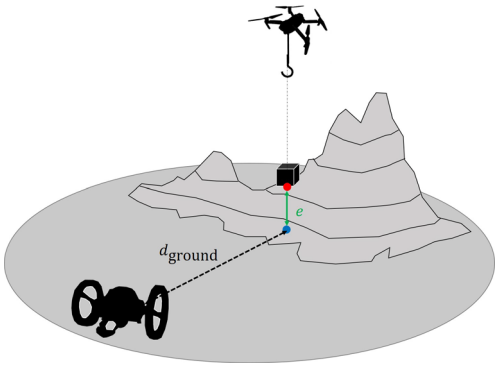

Fig. 9 In determining the terrain elevation, the flying robot drops a standard object with a known size in the desired place. The ground robot then receives via the flying robot its distance to the dropped object, or else estimates this distance using (3). This information is used by the ground robot to determine the point (shown in blue) that the object would touch the ground if the ground were flat, using (3). Then via its camera, the ground robot captures sight of the point (shown in red) that the object actually touches the ground. The pixel distance between these two points is used to compute the elevation of the ground (shown in green) (color figure online)

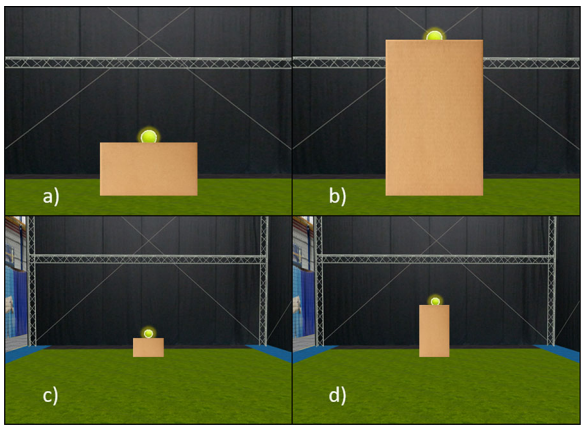

Fig. 10 When the object sits close to the ground robot, as in **a** and **c**, or is at a high elevation, as in **b** and **d**, partial occlusion is more significant. In the image **a** $d_{\text{ground}} = 1$ m, $e = 25$ cm, **b** $d_{\text{ground}} = 1$ m, $e = 75$ cm, **c** $d_{\text{ground}} = 3$ m, $e = 25$ cm, **d** $d_{\text{ground}} = 3$ m, $e = 75$ cm

adhesive texture that are more suitable for simplifying their placement on a non-smooth ground. After a large number of such objects have been distributed around the environment, the ground robots will be deployed to detect these objects and the corresponding elevation of the ground underneath them, as a part of mapping the unknown environment. Therefore, no precision in locating these objects via the flying robots in the environment is in general required. In case such precision is needed for some reasons, as it was mentioned, an adhesive material may be used to place the object more precisely. In such a case, the movements and altitude of the flying robots should also more precisely be controlled.

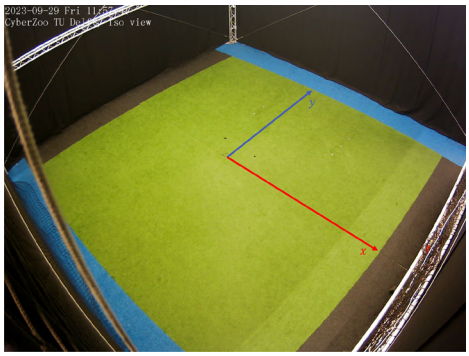

Fig. 11 CyberZoo including the reference frame, where the origin corresponds to the center of the laboratory field

3 Case studies

This section explains the setup, implementation, and results of real-life experiments that were conducted at the CyberZoo of the Delft University of Technology, in order to validate our proposed approaches. In the experiments, challenging scenes that encompass occlusions and the temporary unavailability of the sensors have also been generated.

3.1 Setup of the experiments

The experiments were conducted at the CyberZoo of the Delft University of Technology (see Fig. 11), a research and test laboratory that embeds a $10 \times 10\,\text{m}^2$ synthetic turf surrounded by safety nets, for protecting the participants and robots during the experiments. Furthermore, the experimental facilities at the CyberZoo are equipped with twelve high-tech cameras and Motive, a software platform designed to control motion capture systems for various tracking applications. By placing markers asymmetrically in a rigid body, it is possible to track them using Motive to obtain their position, velocity, and 3D orientation (pitch, roll, and yaw).

The drone used in the experiments was a Parrot Bebop 2 (see the right-hand side photo in Fig. 12). Parrot Bebop 2 is a small quadcopter that measures 382 mm in front, 328 mm on either side and 89 mm in height. It weighs 500 g and has a 2700 mAh battery. Depending on the circumstances, the drone can fly continuously for up to 25 min on this battery power. The 14 MP front camera on Parrot Bebop 2 can record 1080p video at 30 fps. The drone is also equipped with a bottom camera used to estimate the velocity of the drone. This built-in camera is not suited for video recording. Thus, a 14 MP bottom camera, which can also record 1080p video at 30 fps was attached to the bottom of the drone. The drone has its own WiFi network, allowing the drone to connect to other devices. It boasts a dual-core processor operating at 500 MHz per core, orchestrating seamless flight control and navigation. Complemented by a quad-core GPU, the drone efficiently

Fig. 12 Parrot Jumping Sumo (left) and Parrot Bebop 2 (right)

processes video and image data and ensures smooth and high-quality visuals.

As for the ground robot, the Parrot Jumping Sumo (see the left-hand side photo in Fig. 12), which is a compact, wheeled ground robot, was used. Displaying a movable base with two independently driven wheels, Parrot Jumping Sumo measures 185 mm in length, 150 mm in width, and 110 mm in height. Weighs 180 g, and features a 550 mAh battery that grants an operational time of up to 20 min. The robot can perform horizontal and vertical jumps up to 80 cm, and incorporates a wide-angle camera providing 480 p at 15 fps. Just like Parrot Bebop 2, Parrot Jumping Sumo is equipped with its own WiFi network, allowing it to connect to other devices. Parrot Jumping Sumo is powered by an ARM Cortex A9 processor running at a clock speed of around 1 GHz, facilitating swift and responsive execution of commands and sensor data processing, enabling agile movements and interactions.

Overall, the autonomy of the robots, the sufficient quality of their cameras, and their affordability, ready-to-use nature, and user-friendly interfaces make them suitable for this research. The processing of the images and videos was performed via an external computer that was equipped with an Intel Core i7-1185G7 processor, part of the 11th generation, operating at a base frequency of 3.0 GHz and reaching up to 4.8 GHz with Turbo Boost capability. It boasted 16 GB of LPDDR4x RAM and integrated Intel Iris Xe Graphics. The storage was facilitated by a 512 GB PCIe NVMe SSD. The operating system utilized was Ubuntu 20.04.6 LTS. The source codes were composed using Python, owing to its compatibility with external code, versatility, and robust libraries.

3.2 Experiments for distance estimation via ground images

A collection of images of various body poses from volunteer participants were collected. In order to represent a general population, the participants were selected to have diverse physical characteristics, e.g., different heights, and shoulder-to-shoulder and shoulder-to-elbow distances. The experiments included 24 participants of 12 different nationalities. The reference values used for the height, shoulder-to-shoulder distance, and shoulder-to-elbow distance were selected based on [33] and are given in Table 2.

The corresponding measurements were also gathered from the participants, in order to gain awareness of how

Table 2 Measurements and reference values for the participants

Measurement	Reference [m]	μ [m]	σ [m]
Height	1.7000	1.7538	0.0732
Shoulder-to-shoulder	0.3800	0.4075	0.0398
Shoulder-to-elbow	0.3000	0.3054	0.0218

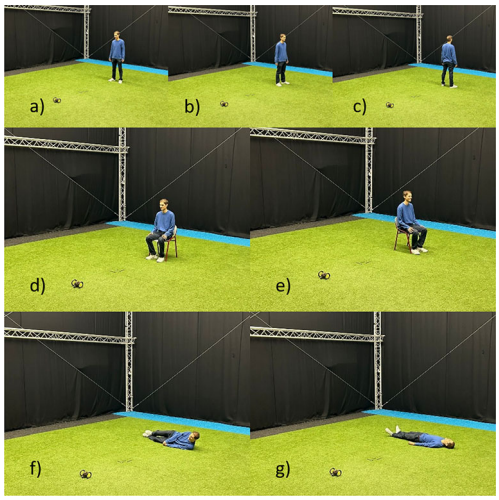

Fig. 13 The participants were asked to take specific poses for the ground vehicle camera. The images were captured systematically from varying distances of 1.5 m and 3 m. The sequence of the poses encompasses **a** standing facing the camera, **b** standing in profile, **c** standing back to the camera, **d** sitting facing the camera, **e** sitting in profile, **f** lying on the side facing the camera, and **g** lying face-up

closely they matched the reference values. Larger deviations from the reference values in the participant sample are expected to yield larger errors. The average value, μ, and standard deviation, σ, for these measurements are also given in Table 2. For height, shoulder-to-shoulder distance, and shoulder-to-elbow distance, the average values are, respectively, 3.16%, 7.24%, and 1.80% larger than their reference values, whereas the corresponding standard deviations are, respectively, 4.17%, 9.77%, and 7.14% of their average values. Thus, the measurements from the participants matched the reference values relatively closely, with the shoulder-to-shoulder distance the least reliable measurement in terms of both the average value and the standard deviation. While the average value for the shoulder-to-elbow distance was closer to the reference value than that for the height, the worst-case measurement was still less reliable than that of the height due to its increased standard deviation. On grounds of the challenge of determining these distances precisely in a conventional SaR scenario, the algorithm privileges using the height, shoulder-to-shoulder distance, and shoulder-to-elbow distances owing to their decreasing absolute values.

Each participant was asked to take 7 different poses for the camera (see Fig. 13): standing facing the camera, stand-

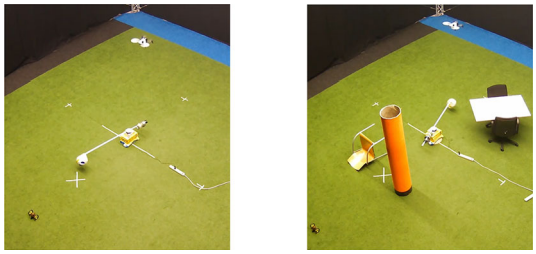

Fig. 14 The motorized rotating stand makes the volleyball move in a circle with constant angular velocity $\omega = 6$ rpm. During the experiments, the flying and ground robots captured video streams of the movement of the volleyball in an unobstructed environment (left) and in a cluttered environment (right). The radius of the trajectory of the volleyball was adjustable

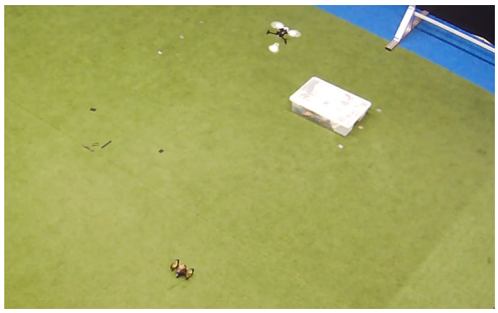

Fig. 15 The flying robot carried and dropped a tennis ball where the elevation of the terrain was to be estimated. The ground robot positioned itself in an adequate perspective, in order to estimate the elevation

ing in profile, standing back to the camera, sitting facing the camera, sitting in profile, lying on the side facing the camera, and lying face-up. These poses have been repeated for distances of 1.5 m and 3 m from the ground robot. The experiments thus generated 14 images per participant and 336 images in total. All images were post-processed using the extended YOLOv8n-Pose, to predict the bounding boxes and the location of the key points. The key points are augmented by leveraging typical body proportions and body symmetry as described in Sect. 2.1. This ensured that all key points were estimated even when some body parts were not detected or captured by the camera. The distance between the key points were then inserted into (3) to determine the distance of the participant from the ground robot in pixels, and to compare this with a reference real-world value for that distance. As different poses distort different key point distances, the camera frame distance substituted in the equation depended on the pose. When the participant was standing or lying on the floor, the distance between the feet and the head was compared against a reference value for the human height. When the participant was sitting facing the camera or in profile, the shoulder-to-shoulder distance or the shoulder-to-elbow distance was compared to the reference values, respectively.

3.3 Experiments for tracking a moving object

The main aim of this set of experiments was to track the movement of a real-world object by leveraging the video streams of the ground and flying robots. Due to the limited space at the experimental facility and the need to compare the obtained movement trajectory with a ground truth, a precise trajectory had to be replicated. Therefore, a volleyball was taken as the moving object to be tracked. The volleyball was attached to a rod, tied to a 360-degree motorized rotating stand. By moving the rod, it was possible to manipulate the radius of the trajectory. The motorized rotating stand allowed for adjustable angles of rotation, directions of rotation, and angular velocities. During the experiments, different obstacles were placed around the trajectory of the volleyball to occlude both the aerial and the ground views of the motion (see Fig. 14).

In all the experiments, the center of rotation was placed at the center of the CyberZoo, i.e., at $(x, y) = (0, 0)$. The ground robot was kept static at $(0, -3)$, and the flying robot hovered with an altitude of 3 m, as closely as possible above the center of the CyberZoo. It was crucial to keep the altitude of the flying robot approximately constant, so that the homography matrix $\mathbf{H}$ does not regularly need to be calibrated. Correcting a frame manually took 10 s on average. The videos that were being processed had a length of around 30 s. The frame rate was 15 fps for the ground robot and 30 fps for the aerial robot. This editing step was done before feeding the videos to the algorithm.

Remark 3 In order to reduce the computational errors in the estimation of the trajectory of the victim that are sourced from the relative motion of the robots or from the tilting motion of the aerial robot, a GPS may be used on the robots. With a GPS, the position of the robots with respect to a fixed reference point (e.g., the center of rotation of the volleyball in the case studies) is obtained precisely and the relative distance of the robot and the moving target is corrected accordingly. This, however, may not be feasible in complex SaR applications. Moreover, using GPS on various robots increases the costs of SaR robotics (which contradicts the main goal of this paper, i.e., to propose a framework that also suits low-income countries). Therefore, an alternative option, that was also used in our case studies, is to apply a video editing software, e.g., the daVinci Resolve, which allows editing the frames and to ensure that the center of rotation of the volleyball is placed in the center of each frame.

Throughout the experiments, the angular velocity and the direction of the rotation were kept constant at 6 rpm counterclockwise. Five main settings were tested: $r = 0.75$ m, $r = 1.00$ m, and $r = 1.25$ m in an unobstructed environment, $r = 1.25$ m in a cluttered environment for the flying robot only, and $r = 1.25$ m in a totally cluttered environment.

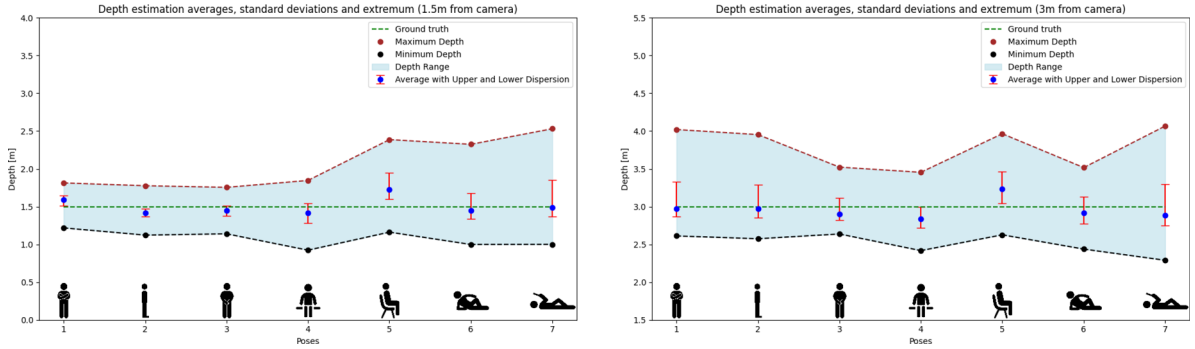

Fig. 16 Average values, standard deviations, and extremum for estimation of the distance for 1.5 m (left) and 3 m (right) ground truth distances

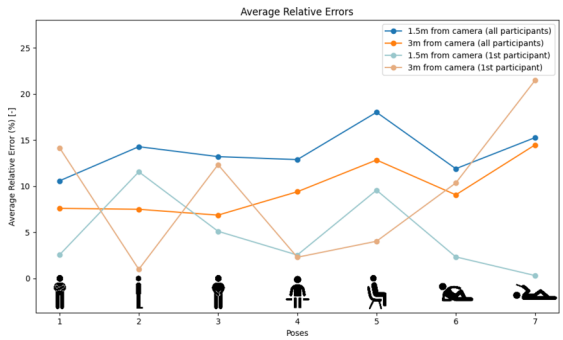

Fig. 17 Average relative errors for a 1.5 m and 3 m distance to the ground robot across all the participants and for an arbitrarily chosen participant

For each experiment, both video streams were post-processed frame by frame and off-board on an external laptop, using YOLOv8n. For the ground robot, (2) and (3) were employed to determine the position of the volleyball in the reference frame depicted in Fig. 11. For the flying robot, (2) was sufficient to determine the position of the volleyball. Since the ground and flying robots recorded videos at, respectively, 15 fps and 30 fps, the frames were aligned before inputting the measured locations to the Kalman filter. The Kalman filter was updated at 30 fps, i.e., without any measurements from the ground robot in half of the updates, due to the halved frame capture rate. The estimated trajectory was compared with the ground truth per setting.

3.4 Experiments for estimation of the ground elevation

In these experiments, the flying robot carried and dropped a signaling object where the elevation of the terrain was to be determined. A remote air-dropping system was incorporated into the setup and a thrower was attached to the bottom of the flying robot. Whenever the object needed to be released, a remote control was manually operated. Even though the maximum payload of the airdrop system is 750 g, weights above 150 g seriously jeopardize flight stability. Therefore, a tennis ball weighing 58 g was selected as the signaling object and boxes with varying heights were used to model the terrain elevation (see Fig. 15). The ground robot was positioned at different distances from the signaling object and 12 images were captured via the camera of the ground robot for distances 1, 2, 3, 4 m and for elevations 25, 50, 75 cm. These images were then processed off-board, considering 2 options: First, the ground robot estimated the terrain elevation on its own, as explained in Sect. 2.4. Second, the ground vehicle received its distance from the tennis ball via the flying robot.

3.5 Results and discussions

Next, we present and discuss the results of our experiments.

3.5.1 Results for distance estimation via ground images

After batch-processing the images, the average, maximum, and minimum distances, along with their upper and lower dispersion, were obtained. Only 5 of 336 images (i.e., less than 1.5%), did not result in detecting a human above a confidence threshold of 20%. For the rest, the results were divided based on the body pose and the distance from the camera (see Fig. 16).

From the plots, the distance to the camera appeared to have the largest impact when the participant was standing. When the person was closer to the camera, the results were promising given the proximity to the ground truth, the small upper and lower dispersion, and the non-substantial errors in the extremum. For larger distances, however, these metrics degraded. When the participant was sitting or lying on the floor, the pattern seemed to be consistent, regardless of the distance of the participant to the ground camera. A possible explanation is the lens distortion (i.e., the optical anomalies in camera lenses that result in deviations from ideal light projection), since this effect is not considered in modeling a simplified pinhole camera.

The worst cases appeared when the distance was estimated in excess, rather than in defect. This is compatible

Table 3 Performance metrics for different setups and trajectories (G: Ground robot alone; F: Flying robot alone; G & F: Collaboration of both robots)

r (m)	RMSE/r (%) [−]			Extreme deviation [m]			Area ratio (%) [−]		
	G	F	G & F	G	F	G & F	G	F	G & F
0.75	**77.218**	89.492	88.090	1.5811	**1.5600**	**1.5600**	39.769	**97.895**	96.557
1.0	93.440	**79.492**	85.697	2.1608	**1.7575**	1.9561	64.832	86.568	**93.551**
1.25	95.028	**62.681**	79.992	3.4031	**1.7921**	2.3525	55.183	26.980	**65.447**
1.25-obstacles	**95.028**	100.946	98.035	3.4031	3.6800	**2.4621**	55.183	37.896	**78.164**

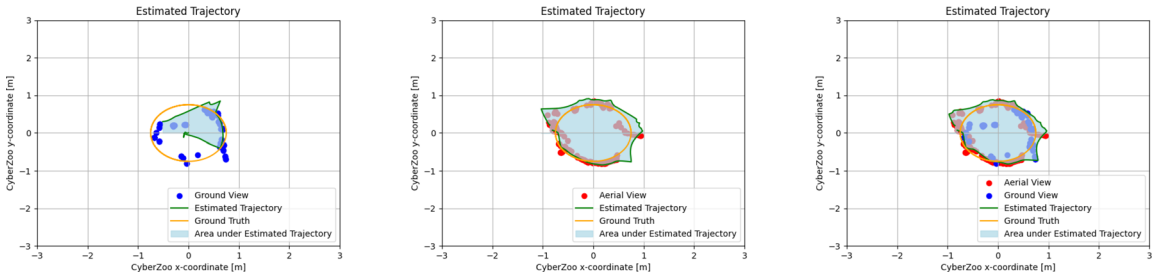

Fig. 18 Estimated object trajectories for $r = 0.75$ m, with ground robot alone (left), flying robot alone (middle), and collaboration of both robots (right)

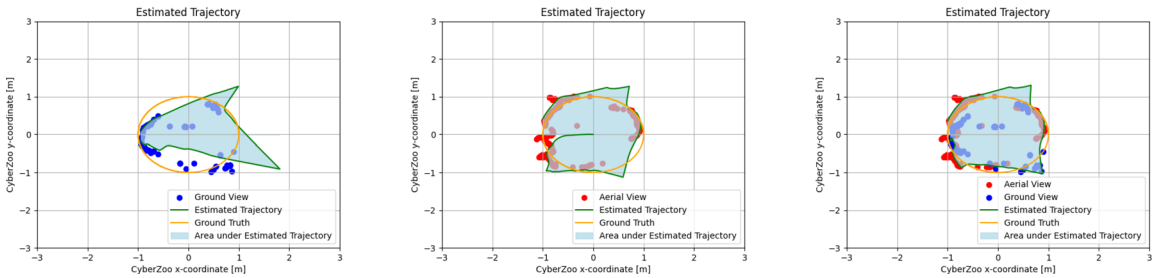

Fig. 19 Estimated object trajectories for $r = 1.0$ m, with ground robot alone (left), flying robot alone (middle), and collaboration of both robots (right)

with the fact that the participants were, on average, taller than the reference values, which increases the estimated distance. Therefore, the reference values should be adapted depending on the average height of the participating people.

Remark 4 When the height of the victims is considered as the reference value for determining their distance from the ground robot, errors may occur especially whenever the height of a victim varies significantly from the values that have been used in the calibration. In fact, the distance of a person with a significantly larger height may by mistake be estimated smaller than it really is and vice versa. In order to mitigate such errors, instead of the height, the relative distance between other key points of the body that is not subject to significant variations among different people may be considered, e.g., the relative distance of the eyes or other key points in the face.

The standing poses appeared to be, on average, the least prone to errors. While their error pattern changed signifi-

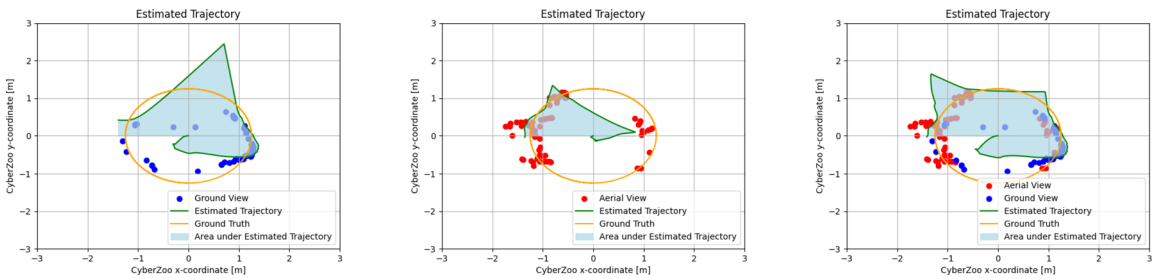

Fig. 20 Estimated object trajectories for $r = 1.25$ m, with ground robot alone (left), flying robot alone (middle), and collaboration of both robots (right)

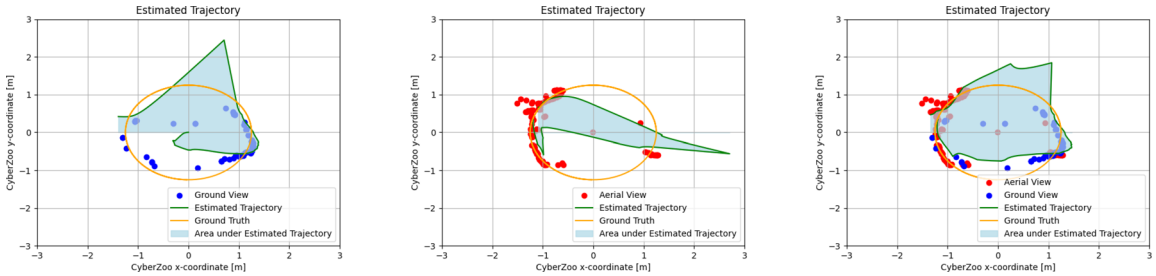

Fig. 21 Estimated object trajectories for $r = 1.25$ m when obstacles were placed in the sight of the aerial robot, with ground robot alone (left), flying robot alone (middle), and collaboration of both robots (right)

cantly with the distance from the camera, the average distance remained close to the ground truth. Sitting and lying poses appeared to be more challenging. In fact, whenever the participants have a wider range of motion and poses where a smaller measurement is taken to compare against the reference (e.g., shoulder-to-shoulder distance as opposed to the height) are more prone to uncertainties. For instance, the participants stood still in a similar fashion while standing. However, while sitting, they placed their legs more openly or closely, and curved their back to different extents. The same goes for lying positions, where the participants tilted their bodies and stretched their arms and legs to different degrees.

The average relative errors of the estimated distances per pose across all participants and for an arbitrarily chosen participant are given in Fig. 17. As expected, the curves kept the same pattern except for the standing poses. Moreover, the sitting and lying poses generally displayed larger relative errors, than the standing poses. The slight discrepancy in the pattern for the standing poses is because the faces of some participants standing 1.5 m from the camera were not captured in the image. Thus, the position of their faces was estimated by the algorithm, giving rise to larger errors. Nevertheless, sitting in profile and lying face-up still appeared to be the most challenging cases.

The average across all poses for the relative errors is 13.73% and 9.67% for ground truth distances of, respectively, 1.5 m and 3 m. This 4.06% discrepancy was because the participants did not always place themselves in the exact expected distance from the ground robot. These slight displacements of a few centimeters have a larger impact on the average relative errors when the distance is smaller. Assuming that this is the only external source of the error, the misplacement amounts on average to 12.18 cm. This implies that the actual relative error resulting from the algorithm itself ranges from around 4% to 10%.

Indeed, most of the relative errors lie between $< 1\%$ to 12%. Moreover, the relative errors appear to have no or negligible correlation with the distance from the camera. For the arbitrarily chosen participant, the relative error is suspected to arise from the physical discrepancies of the participant, as well as the slight deviations from the desired pose. The

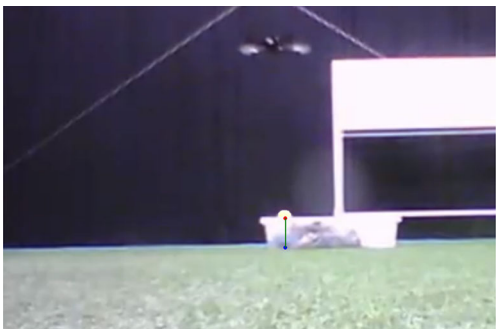

Fig. 22 The algorithm draws the red point where the tennis ball touches the ground, the blue point where the tennis ball would touch the ground if the ground was flat, and the line connecting these points. The reduced quality of the image is because the ground robot captured it while moving

participant had a height of 1.83 m, shoulder-to-shoulder distance of 40 cm, and a shoulder-to-elbow distance of 31 cm, which are, respectively, 7.65%, 5.26%, and 3.33% above the reference values. Therefore, poses that rely on the height of the participant, i.e., poses 1, 2, 3, 6, 7, are more prone to errors compared to other poses.

Finally, the average relative errors were shown to be distance-independent, which means that the absolute errors varied linearly with the distance from the camera. The relative errors varied from $< 1\%$ to $> 20\%$, depending on the pose and the physical characteristics of the participant. In 10 runs of the algorithm, the 336 images needed 83.38 s to 93.43 s to be processed, averaging to 0.248 s to 0.278 s per image.

3.5.2 Results for tracking a moving object

The volleyball trajectories obtained after analyzing the videos were compared with the ground truth, based on various performance metrics, including the root mean square error (RMSE), extreme deviation, and area ratio. The RMSE shows how close the estimated volleyball trajectories are on average to the ground truth trajectories. Extreme deviation spots where these estimates and the ground truth stray the

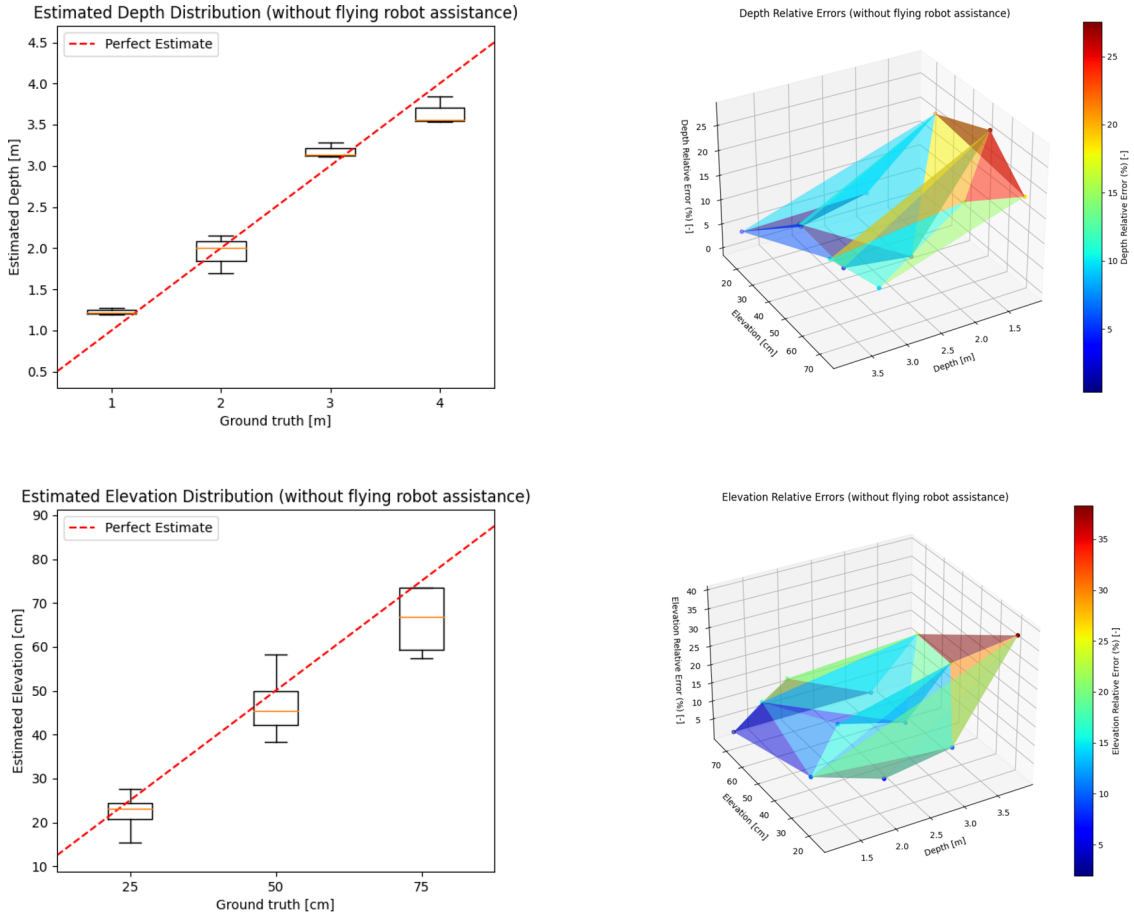

Fig. 23 Distance distribution (top, left) and relative errors (top, right), elevation distribution (bottom, left) and relative errors (bottom, right), solely by the ground robot

farthest. The area ratio reflects the percentage of the area enclosed by the ground truth that is also covered by the estimated trajectory, thus the match between the overall shapes. Considering the three metrics provides a more comprehensive insight about how accurate the estimations are. Using a volleyball instead of a human increased the difficulty of the tracking task for the flying robot, due to the smaller size and a less distinctive pattern of movement. Thus, in the presence of a cluttered environment with shadows, the flying robot had difficulty detecting the volleyball. The analysis was performed for a turning radius of $r = 0.75, 1.0, 1.25$ m with unobstructed views for both the flying and the ground robots, and for $r = 1.25$ m with obstacles placed for only for the flying robot.

The performance metrics for cases with only the ground robot, only the flying robot, and both robots are shown in Table 3, where the best performance per metric and scenario are shown with bold fonts. Moreover, Figs. 18, 19, 20 and 21 illustrate the estimated trajectories for $r = 0.75, 1, 1.25$ m and for $r = 1.25$ m with obstacles placed for the flying robot only. As expected, the metrics were degraded with increasing the radius of the trajectory in cluttered environments.

From Table 3, the RMSE for the 4 scenarios with the collaborating robot lies between the RMSE when the robots are used solely. However, not the same robot always performed the best. In fact, the standalone performance of these robots depended on their measurements, making it impractical to predict beforehand robots should be applied to an unknown scenario. Therefore, using a collaborative team of a ground and a flying robot is the most promising for handling an unknown scenario.

Regarding the extreme deviations from the ground truth, the flying robot produced the best results, whenever its view was unobstructed. This is because the camera of the flying robot has a larger fps, capturing points closer to each other, which prevents large errors by the Kalman filter. Nevertheless, when the view of the flying robot was obstructed, due to the missing measurements the estimated trajectory deviated significantly from its ground truth. Therefore, for an unknown SaR scenario, including the ground robot next to

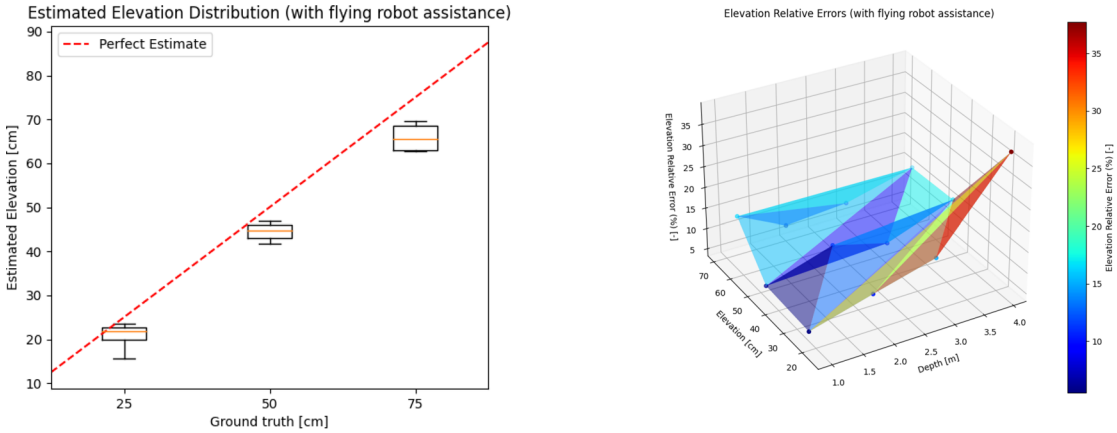

Fig. 24 Elevation distribution (left) and relative errors (right), assuming that the flying robot provides a precise estimation of the distance for the ground robot

the flying robot can significantly improve the estimations for tracking a moving object.

For the area ratio, the team of both flying and ground robots stood out the most. The figures show that the estimated trajectories remained close circular shapes. For $r = 1.25$ m with an unobstructed view, the metric degraded since none of the robots detected the volleyball in the first instants. Thus, the volleyball was assumed to be at the center of the CyberZoo, impacting the final shape of the estimated trajectory of the volleyball. These findings are supported by the results presented in Sect. 3.

In 10 runs of a 10 s video for the ground view, it took between 19.41 s and 22.30 s to obtain the trajectories. Similarly, it took between 20.16 s and 22.68 s to process the aerial videos. For the team of robots, the procedure took between 36.68 s and 42.19 s. This is the time that the algorithm takes to process the edited videos, excluding the time that is needed to edit them. The editing has been done in advance, as addressed in Sect. 3.3.

3.5.3 Results for estimation of the ground elevation

Unless specified, the algorithm assumed by default that the flying robot did not provide assistance in the elevation estimation. An example of the output of the algorithm is shown in Fig. 22. In this frame, the ground distance of the point, for which the elevation should be estimated, from the ground robot is 2 m and the elevation is 20 cm. The algorithm estimates a distance of 2.091 m and an elevation of 18.13 cm, indicating relative errors of, respectively, 4.55% and 9.35%.

Figure 23 illustrates the distribution of the measurements and relative errors for different distance and elevation values, when the ground robot had to estimate the elevation alone. From the top left plot, the distribution of the distance estimated by the ground robot follows the perfect estimate closely. For smaller (larger) distances, the algorithm slightly overestimates (underestimates) the distance. This may be related to lens distortion, reduced resolution for larger distances, or the need for more precise calibration for larger distances. The curves for the relative errors show inverse trends, i.e., while the relative errors for the distances tend to increase with decreasing the distance, the relative errors for the terrain elevation tend to increase with increasing the distance, and more specifically, with decreasing the elevation. Both of these trends are according to the expectation. For smaller distances, any deviation from reality has a higher impact on the relative error of the estimated distance. Since the estimate for the elevation depends on the estimate for the distance, the errors for the elevation are a consequence of both error sources. Even though for larger distances, the relative errors for the distance typically decrease, the absolute errors slightly increase as shown in the figure. This error propagates and has a higher impact on the elevation relative error when the elevation is small. As shown in the 3D graphs, the relative errors for the distance range from around 2% to 25%, whereas the elevation relative errors range from around 0% to 35%.

Figure 24 shows the elevation distribution and relative errors for the same scenarios as before, this time with the assistance of the flying robot in providing the distance values. Although the median values for the estimates of the terrain elevation did not change, the dispersion of the elevations around the mean value was significantly reduced. In this layout, the average relative errors were also notoriously mitigated. Except for one data point, the relative errors were lower than 13% in all instances. The data point corresponding to an elevation of 25 cm and a distance of 4 m appears to be an outlier, where nearly no error was mitigated. This may be because the homography matrix responsible for mapping the distance into real-world coordinates was calibrated using reference points up to distances of 3 m.

Finally, in 10 runs, the processing of the images took on average between 0.54 s and 0.62 s.

4 Conclusions and future research

We leveraged the complementary capabilities of flying and ground robots, in order to deliver affordable, effective solutions based on image processing for unknown SaR environments. In particular, the following SaR tasks were considered: estimating the location and distance of a human from the ground robot based on the images captured by the robot, tracking the trajectory of a moving object by applying data fusion and a Kalman filter, and estimating the terrain elevation. In order to ensure a simplistic setup with low hardware costs, these tasks are performed via visual depiction only, through images and videos.

We performed real-life experiments in order to validate the proposed approaches, which are founded on You Only Look Once (YOLO). The experiments proved the efficiency of the proposed methods in the accurate estimation of the position of humans in various poses from their images. In fact, the average relative errors remained mainly below 10%. In tracking the trajectory of a moving object, the advantage of deploying a collaborative team of a flying and a ground robot was evident, especially in properly re-generating the overall shape of the trajectory. Furthermore, this setup is more fault-safe in keeping a low root mean square error and in preventing large deviations from the ground truth, especially in cluttered environments. In fact, for the widest trajectory simulated and in the presence of obstacles to the aerial view, compared to the best-performing robot, the collaboration of the robots resulted in a reduction of 28% in the extreme deviation from the true trajectory, and an increase of 42% in the area covered ratio. Finally, the estimation of the terrain elevation was prone to two primary sources of error: error in the estimation of the distance and error in the subsequent elevation estimation. The distance estimation inaccuracies stem from simplified camera models and low image resolution, which impact object detection, compounded by occasional object occlusion. The elevation errors result from poor homography calibration in specific regions, which is exacerbated by the double application of the simplified camera model. Without the assistance of the flying robot in providing the distance of the human from the ground robot, the relative errors in estimation of the terrain elevation were up to 35% in some cases. This error was reduced to less than 13% when the flying robot estimated the distance for the ground robot.

In summary, these results represent notable progress in leveraging visual data and robotic capabilities for SaR. Since calibrating the homography matrix is expected to improve with the number of reference points, which slows down the computations, it is crucial to investigate the optimal number of reference points that provides a trade-off between accuracy and computational burden. Furthermore, the results hint that implementing a simplified pinhole camera model may result in significant errors. Thus, including a more realistic camera model that captures optical phenomena, e.g., lens distortion and parallax, is worth looking into. As an extension to the current work, the proposed algorithm for locating the victims should also be validated for non-conventional poses of the victims that are much different from the 7 poses that have been shown in Fig. 16 and that are not common in the COCO dataset. Moreover, it will be advantageous to make the flying robot autonomously capable of identifying the regions of interest for scanning, in collaboration with the ground robot. This may require further interaction and exchange of data/information between the two robots. Finally, validating our proposed approaches in various real-life SaR scenarios, where a combination of pose estimation, victim localization, victim tracking, and terrain elevation mapping should be conducted, is a topic for further future work.

Author Contributions Author B. Esteves Henriques performed the conceptualization, designed and performed the experiments, contributed to methodology, analysis and validity assessment of the results, and wrote the first draft of the paper. Author M. Baglioni contributed to methodology, analysis and validity assessment of the results, supervision, and reviewing the final draft of the paper. Author A. Jamshidnejad contributed to conceptualization, methodology, analysis and validity assessment of the results, editing the final draft of the paper, project administration and supervision, and funding acquisition.

Funding This research has been supported jointly by the NWO Talent Program Veni project "Autonomous drones flocking for search-and-rescue" (18120), which has been financed by the Netherlands Organisation for Scientific Research (NWO) and by the TU Delft AI Labs & Talent programme.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no Conflict of interest.

Ethical approval This work involved human subjects in its research. Approval of all ethical and experimental procedures and protocols was granted by the Human Research Ethics Committee of TU Delft under Approval 3457, issued on September 26, 2023.

Consent for publication Not applicable.

References

1. Zorn, M.: Natural Disasters and Less Developed Countries, pp. 59–78. Springer, Cham (2018)
2. Qi, J., Song, D., Shang, H., Wang, N., Hua, C., Wu, C., Qi, X., Han, J.: Search and rescue rotary-wing UAV and its application to the Lushan Ms 7.0 earthquake. J. Field Robot. **33**(3), 290–321 (2016)
3. Murphy, R.R., Tadokoro, S., Kleiner, A.: Disaster Robotics, pp. 1577–1604. Springer, Heidelberg (2016)
4. Rodin, C.D., Lima, L.N., Alcantara Andrade, F.A., Haddad, D.B., Johansen, T.A., Storvold, R.: Object classification in thermal images using convolutional neural networks for search and rescue missions with unmanned aerial systems. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE, Rio de Janeiro, Brazil (2018)
5. McGee, J., Mathew, S.J., Gonzalez, F.: Unmanned aerial vehicle and artificial intelligence for thermal target detection in search and rescue applications. In: 2020 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 883–891. IEEE, Athens, Greece (2020)
6. Zelten, J.P.: Digital photography and the dynamics of technology innovation. PhD thesis, Massachusetts Institute of Technology, Boston, MA, USA (February 2002)
7. Hummel, S., Hudak, A., Uebler, E.H., Falkowski, M., Megown, K.: A comparison of accuracy and cost of LiDAR versus stand exam data for landscape management on the Malheur National Forest. J. For. **109**, 267–273 (2011)
8. Zhaohua, L., Bochao, G.: Radar sensors in automatic driving cars. In: 2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT), pp. 239–242. IEEE, Nanchang, China (2020)
9. Falconer, D.G., Ficklin, R.W., Konolige, K.G.: Robot-mounted through-wall radar for detecting, locating, and identifying building occupants. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings, vol. 2, pp. 1868–1875. IEEE, San Francisco, CA, USA (2000)
10. Geisheimer, J.L., Marshall, W.S., Greneker, E.: A continuous-wave (CW) radar for gait analysis. In: Conference Record of Thirty-Fifth Asilomar Conference on Signals. Systems and Computers, vol. 1, pp. 834–838. IEEE, Pacific Grove, CA, USA (2001)
11. Zhou, Q., Liu, J., Host-Madsen, A., Boric-Lubecke, O., Lubecke, V.: Detection of multiple heartbeats using Doppler radar. In: 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 2, p. . IEEE, Toulouse, France (2006)
12. Thavasi, P.T., Suriyakala, C.: Sensors and tracking methods used in wireless sensor network based unmanned search and rescue system-a review. Procedia Eng. **38**, 1935–1945 (2012)
13. Condotta, I.C.F.S., Brown-Brandl, T.M., Pitla, S.K., Stinn, J.P., Silva-Miranda, K.O.: Evaluation of low-cost depth cameras for agricultural applications. Comput. Electron. Agric. **173**, 105394 (2020)
14. Hill, A.P., Prince, P., Snaddon, J.L., Doncaster, C.P., Rogers, A.: Audiomoth: a low-cost acoustic device for monitoring biodiversity and the environment. HardwareX **6**, 00073 (2019)
15. Titi, H.H.: Feasibility Study for a Freeway Corridor Infrastructure Health Monitoring (HM) Instrumentation Testbed. Wisconsin DOT Research & Library Unit, Madison (2012)
16. Van Nam, D., Gon-Woo, K.: Solid-state LiDAR based-SLAM: a concise review and application. In: 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 302–305. IEEE, Jeju Island, Korea (South) (2021)
17. Beňo, P., Duchoň, F., Hubinský, P., Dekan, M., Tölgyessy, M., Dobiši, M.: RGBD mapping solution for low-cost robot. Mach. Vis. Appl. **33**, 21 (2022)
18. Fankhauser, P., Bloesch, M., Krüsi, P., Diethelm, R., Wermelinger, M., Schneider, T., Dymczyk, M., Hutter, M., Siegwart, R.: Collaborative navigation for flying and walking robots. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2859–2866. IEEE, Daejeon, Korea (South) (2016)
19. Delmerico, J., Mueggler, E., Nitsch, J., Scaramuzza, D.: Active autonomous aerial exploration for ground robot path planning. IEEE Robot. Autom. Lett. **2**(2), 664–671 (2017)
20. Adel Musallam, M., Baptista, R., Al Ismaeil, K., Aouada, D.: Temporal 3D human pose estimation for action recognition from arbitrary viewpoints. In: 2019 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 253–258. IEEE, Las Vegas, NV, USA (2019)
21. Nath, N.D., Cheng, C.-S., Behzadan, A.H.: Drone mapping of damage information in GPS-denied disaster sites. Adv. Eng. Inform. **51**, 101450 (2022)
22. Xing, L., Fan, X., Dong, Y., Xiong, Z., Xing, L., Yang, Y., Bai, H., Zhou, C.: Multi-UAV cooperative system for search and rescue based on YOLOv5. Int. J. Disaster Risk Reduct. **76**, 102972 (2022)
23. Bejiga, M.B., Zeggada, A., Melgani, F.: Convolutional neural networks for near real-time object detection from UAV imagery in avalanche search and rescue operations. In: 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 693–696. IEEE, Beijing, China (2016)
24. Feraru, V.A., Andersen, R.E., Boukas, E.: Towards an autonomous UAV-based system to assist search and rescue operations in man overboard incidents. In: 2020 IEEE International Symposium on Safety. Security, and Rescue Robotics (SSRR), pp. 57–64. IEEE, Abu Dhabi, United Arab Emirates (2020)
25. Martinez-Alpiste, I., Golcarenarenji, G., Wang, Q., Alcaraz-Calero, J.M.: Search and rescue operation using UAVs: a case study. Expert Syst. Appl. **178**, 114937 (2021)
26. Jiang, P., Ergu, D., Liu, F., Cai, Y., Ma, B.: A review of Yolo algorithm developments. Procedia Comput. Sci. **199**, 1066–1073 (2022)
27. Putra, Y.I., Alasiry, A.H., Darmawan, A., Oktavianto, H., Darmawan, Z.M.E.: Camera-based object detection and identification using YOLO method for Indonesian search and rescue robot competition. In: 2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), pp. 508–513. IEEE, Yogyakarta, Indonesia (2022)
28. Vasuhi, S., Vijayakumar, M., Vaidehi, V.: Real time multiple human tracking using Kalman filter. In: 2015 3rd International Conference on Signal Processing. Communication and Networking (ICSCN), pp. 1–6. IEEE, Chennai, India (2015)
29. Li, P., Zhang, T., Ma, B.: Unscented Kalman filter for visual curve tracking. Image Vis. Comput. **22**(2), 157–164 (2004)
30. Howard, A., Seraji, H.: Vision-based terrain characterization and traversability assessment. J. Robot. Syst. **18**(10), 577–587 (2001)
31. Jocher, G., Chaurasia, A., Qiu, J.: YOLO by Ultralytics (2023). https://github.com/ultralytics/ultralytics
32. Korhonen, T., Hostikka, S.: Fire dynamics simulator with evacuation: FDS+Evac (version 5). VTT Technical Research Centre of Finland (2014)
33. Fryar, C.D., Gu, Q., Ogden, C.L., Flegal, K.M.: Anthropometric Reference Data for Children and Adults; United States, 2011–2014. Vital and Health statistics. Series 3, Data from the National Health and Nutrition Examination Survey, vol. 39 (2016)

Bernardo Esteves Henriques received the MSc degree from the Delft University of Technology, the Netherlands, in 2024. He is currently an Aerospace Engineer at Cargolux Airlines International S.A., Luxembourg. His research interests include computer vision and intelligent robotics.

Mirko Baglioni received the MSc degree from University of Pisa, Italy, in 2018. He is currently a PhD candidate at the Faculty of Aerospace Engineering, Delft University of Technology, the Netherlands. His research is focused on optimal and AI-based control with applications to search-and-rescue robotics.

Anahita Jamshidnejad received the PhD cum laude degree from the Delft University of Technology, the Netherlands, in 2017. She is currently Assistant Professor at the Control and Operations Department, Delft University of Technology. Her main research interests include autonomous perception and control for search and rescue robots, as well as integrated model predictive control and AI methods.