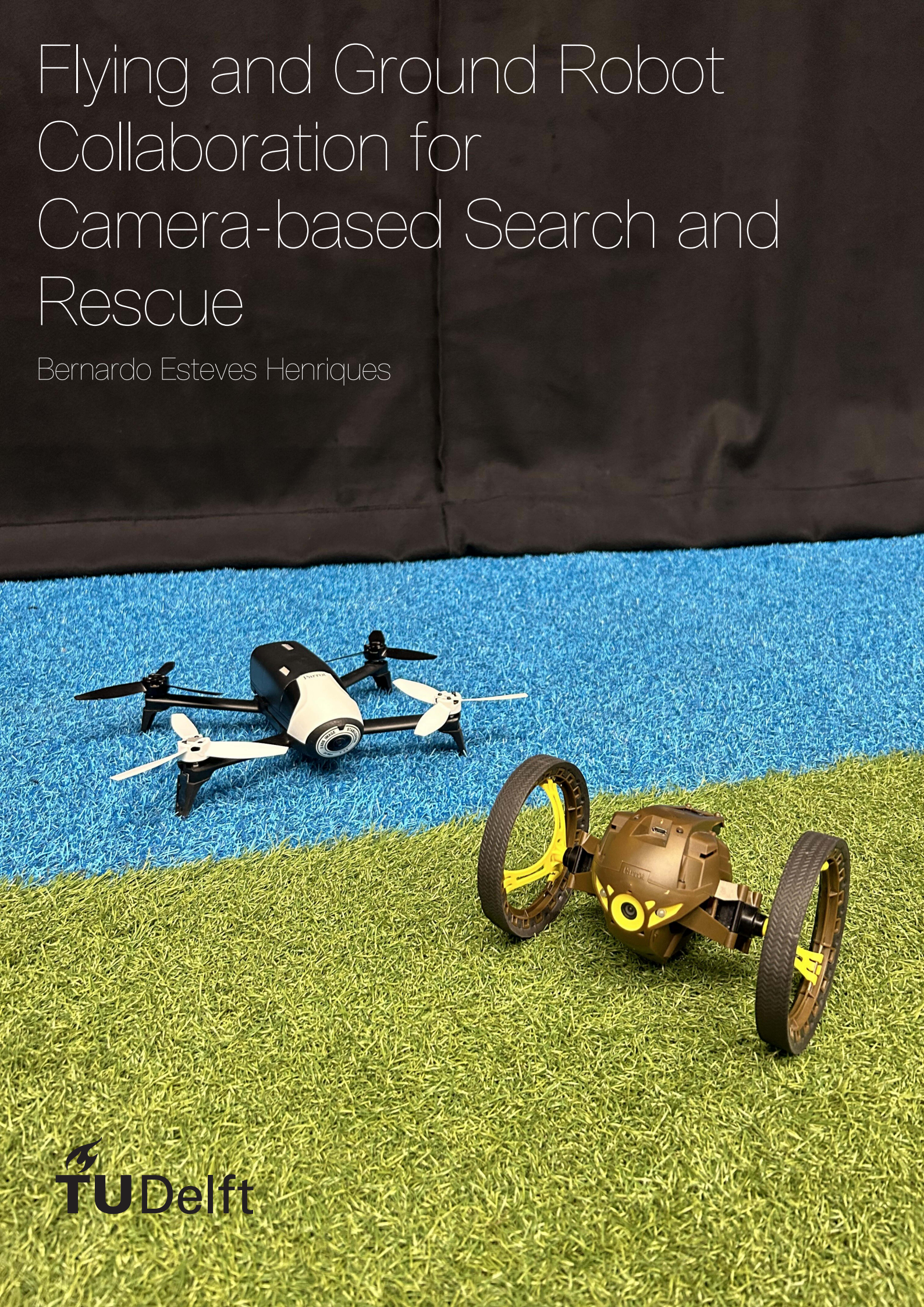
Flying and Ground Robot Collaboration for Camera-based Search and Rescue

Bernardo Esteves Henriques



TU Delft

Flying and Ground Robot Collaboration for Camera-based Search and Rescue

Thesis report

by

Bernardo Esteves Henriques

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on February 5, 2024 at 13:00

Thesis committee:

Dr. Ir. E. van Kampen (chair)
Dr. A. Jamshidnejad
Dr. R. van de Plas

Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	December, 2022 - December, 2023
Student number:	5522978

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Aerospace Engineering · Delft University of Technology

Acknowledgments

They say to leave home is to break your own heart. Looking back, I am lucky to say that, even though I left my home country, I never left home. I want to take the chance to appreciate the people who helped me feel at home wherever I went.

To my supervisors, Dr. Anahita Jamshidnejad and Mirko Baglioni, for their constant guidance and support and for teaching me what it means to be a researcher. To Erik van der Horst, for the long afternoons at the laboratory and for making me realize that experimental challenges are not so daunting.

To my parents, for always believing in me. ”Thank you” will never be enough. To Tomás, for being much more than just a brother. I could not be more proud of you. To my cherished friends, for making the path of life a delightful and fulfilling experience. Our treasured moments together have been the bedrock upon which I have built this and all other achievements. To Teresa, the most supporting partner I could ask for, your unwavering encouragement has been a guiding light throughout this journey, and your presence has made the highs brighter and the lows easier to navigate. You know I could not have done any of this without you.

Even though this thesis marks the end of my academic life, I leave with the certainty that, regardless of what the future holds, I will never stop being a student.

Bernardo Esteves Henriques
Delft, November 2023

Contents

Nomenclature iv

List of Figures v

List of Tables vi

1 Introduction 1
- 1.1 Flying and Ground Robots Collaboration . . . 1
- 1.2 Search and Rescue . . . 1
- 1.3 Research Formulation . . . 2
- 1.4 Report Structure . . . 3

I Scientific Article 4

2 Flying and Ground Robot Collaboration for Camera-based Search and Rescue 5
- 2.1 Introduction . . . 5
- 2.2 Literature Review . . . 6
- 2.3 Main Contributions & Structure of the Paper . . . 7
- 2.4 Methodology . . . 8
- 2.5 Case Studies . . . 12
- 2.6 Results & Discussion . . . 16
- 2.7 Limitations & Topics for Future Work . . . 22
- 2.8 Conclusions . . . 23

II Preliminary Analysis 25

3 Literature Review 26
- 3.1 Introduction . . . 26
- 3.2 Image Processing . . . 28
- 3.3 Communication Protocols . . . 35
- 3.4 Control Strategies and Path Planning . . . 40
- 3.5 Conclusion . . . 52

III Closure 54

4 Conclusion 55
- 4.1 Revisiting Research Formulation . . . 55
- 4.2 Closing Remarks . . . 57

References 62

Nomenclature

List of Abbreviations

ANN Artificial Neural Network

BFS Breadth-First Search

BoVW Bag of Visual Words

CNN Convolutional Neural Network

DFS Depth-First Search

DNN Deep Neural Network

FAST Features from Accelerated Segment Test

FLC Fuzzy Logic Control

HREC Human Research Ethics Committee

IoT Internet of Things

MPC Model Predictive Control

MQTT Message Queuing Telemetry Transport

PCI Physical Couple Interface

PID Proportional-Integral-Derivative

PRM Probabilistic Roadmap

PSNR Peak Signal-to-noise Ratio

R-CNN Region-based Convolutional Neural Network

RMSE Root Mean Square Error

ROS Robotic Operating System

RRT Rapidly-exploring Random Tree

SaR Search and Rescue

SIFT Scale Invariant Feature Transformation

SPT Shortest Path Tree

SSIM Structural Similarity Index Measure

SURF Speeded Up Robust Features

SVM Support Vector Machine

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

WLAN Wireless Local Area Network

WSAN Wireless Sensor and Actuator Network

XMPP Extensible Messaging and Presence Protocol

YOLO You Only Look Once

List of Figures

3.1 a) Traditional computer vision workflow, and b) Deep learning workflow. Retrieved from [18]. 28
3.2 A simple ANN architecture composed of a sigmoid hidden layer and a linear output layer. Weights are represented as w, whereas biases are represented as v. Retrieved from [31]. . 30
3.3 A simple CNN architecture composed of five layers. Retrieved from [32]. 31
3.4 Graphical representation of a CNN. Retrieved from [31]. 31
3.5 Example of change in viewpoints for a) flying robot and b) ground robot. Retrieved from [4]. 33
3.6 Sequential UGV-UAV deployment for unknown environment exploration. Retrieved from [45]. 36
3.7 The UAV attaches a tether to a structure by flying around it. Retrieved from [47]. 37
3.8 The UGV climbs a cliff by winding the tether attached by the UAV. Retrieved from [47]. . . . 37
3.9 Architecture of an FLC-based controller. Retrieved from [31]. 40
3.10 Examples of membership function shapes. Retrieved from [31]. 41
3.11 Architecture of a PID fuzzy controller. Retrieved from [31]. 42
3.12 a) COG and b) MOM defuzzification methods. Retrieved from [31]. 42
3.13 Fundamental MPC scheme. Retrieved from [31]. 45

List of Tables

1.1 General considerations and qualitative cost comparison of medium-segment sensors used in SaR. Own elaboration based on [9]–[15]. 2

3.1 Literature review structure. Own elaboration. 28
3.2 Image processing algorithms comparison. Own elaboration. 32
3.3 Overview of the advantages and disadvantages of FLC. Own elaboration. 44
3.4 Overview of the advantages and disadvantages of MPC. Own elaboration. 46
3.5 Overview of the advantages and disadvantages of search algorithms. Own elaboration. . . 50

1 Introduction

In this work, two captivating and impactful domains are combined: the collaboration of heterogeneous robot teams and the vital field of Search and Rescue (SaR). The impetus for this thesis work is rooted in the tremendous potential of combining flying and ground robots to deliver reliable and cost-effective SaR solutions. There is a strong motivation to bolster SaR operations, particularly in regions most severely affected by climate change, notably the least developed countries. Section 1.1 delves into the promising possibilities of integrating aerial and ground-based robots, showcasing real-world applications across various industries. Section 1.2 sheds light on the demanding, yet indispensable field of SaR, underscoring the urgent need for cost-effective solutions. Section 1.3 outlines the research objectives and the fundamental questions addressed throughout the work. Lastly, in Section 1.4, a structured overview of the report is presented, delineating the content of the subsequent sections.

1.1. Flying and Ground Robots Collaboration

The integration of flying and ground robots presents a promising synergy owing to their distinct yet complementary capabilities. Unmanned Aerial Vehicles (UAVs) offer cost-effective access to aerial perspectives and are renowned for their ability to swiftly cover expansive areas. However, their imaging quality can be compromised due to camera tilting during flight, and they are not suited to accessing confined spaces. On the other hand, Unmanned Ground Vehicles (UGVs) navigate rugged terrains at a comparatively slower pace with a restricted field of view. Nonetheless, they excel in tasks requiring heavy payload transport, access to confined spaces, and precise imaging. Some examples of such synergies in real-life applications are the following:

Agriculture and Precision Farming: The collaboration of UAVs and UGVs is extensively employed in precision agriculture. UAVs swiftly survey vast agricultural lands, capturing high-resolution imagery that helps in monitoring crop health and optimizing resource allocation. Subsequently, UGVs navigate through the terrain to perform targeted actions, such as precise application of fertilizers or pesticides, ensuring optimal yield [1], [2].

Infrastructure Inspection and Maintenance: The collaboration of aerial and ground robots is crucial for inspecting critical infrastructure such as bridges, power lines, and pipelines. UAVs are employed for initial aerial inspection, identifying potential issues. Subsequently, UGVs with specialized sensors and tools navigate to conduct close-up inspections and perform necessary maintenance tasks [3].

Disaster Response and Assessment: During natural disasters, combining UAVs and UGVs facilitates rapid and efficient disaster assessment. UAVs survey disaster-affected regions, providing real-time aerial imagery for damage assessment and rescue planning. UGVs are then deployed to navigate through the affected areas, identifying survivors and assessing infrastructure damage in detail [4]–[6].

1.2. Search and Rescue

Natural disasters are a significant and growing threat to human health and safety, with an increasing frequency and cost of damage. In recent decades, the number and severity of natural disasters have risen dramatically, highlighting the need for effective disaster management [7]. Thus, optimizing search and rescue (SaR) operations has become a highly relevant research topic in both academia and industry due

to their importance in mitigating further life losses and injuries. In the case of the least developed countries, the risk is particularly concerning, as nearly 90% of disaster-related deaths and 98% of people affected by disasters between 1991 and 2005 occurred in these countries [8]. Since these countries lack the budget to adopt the most sophisticated SaR solutions, it is vital to make SaR operations not only more reliable but also more affordable.

Ideally, a SaR solution should strike a delicate balance between being cost-effective and accurate. To shed light on the cost implications, Table 1.1 contains a qualitative comparison of the cost implications associated with medium-segment sensors used in SaR applications, such as LiDAR, radar, depth cameras, RGB cameras, thermal imaging, and acoustic sensors.

Table 1.1: General considerations and qualitative cost comparison of medium-segment sensors used in SaR. Own elaboration based on [9]–[15].

Sensor Type	General Considerations	Cost
RGB Cameras	Provides standard RGB imaging	Low
Depth Cameras	Provides depth information, suitable for mapping and navigation	Low
Acoustic Sensors	Commonly used for underwater search and mapping	Low
2D Radar	Detects objects and their velocities, effective in various weather conditions	Medium
2D LiDAR	High accuracy, detailed mapping, suitable for precision applications	Medium
Thermal Imaging	Detects heat signatures, crucial for low visibility conditions	Medium
3D Radar	Same as 2D Radar, but providing information in all three axes	High
3D LiDAR	Same as 2D LiDAR, but providing information in all three axes	High

From the table, it becomes evident that setups relying on RGB cameras, depth cameras, and acoustic sensors emerge as the lowest costly. Algorithms relying on standard RGB imagery appear to present particularly high room for improvement due to recent advancements in image processing. Therefore, the proposed research endeavors to delve into the attainable accuracy of a simple UGV-UAV collaborative team in performing typical SaR tasks, with both robots leveraging exclusively standard RGB imagery.

1.3. Research Formulation

The research will address the challenge of performing conventional SaR tasks, such as victim detection, victim tracking, and elevation mapping, using a simple setup relying on RGB cameras and leveraging the complementary capabilities of the heterogeneous robots. In an attempt to achieve improved performance compared to past solutions, some of the latest advancements in computer vision are incorporated into the framework. In recent years, You Only Look Once (YOLO) has emerged as one of the most promising object detection algorithms. It has proved to achieve real-time object detection while maintaining high accuracy and requiring less computational resources than in several applications. Thus, the research objective that is central to this work is formulated as follows:

Research Objective

Investigate the attainable accuracy of a collaborating flying and ground robot team equipped with RGB cameras and employing YOLO in performing conventional SaR tasks.

The research objective is split into several research questions that will be answered in this thesis. Each of the research questions relates to one of the SaR tasks. The first research question connects with the need for an unconventional approach to capturing depth information without relying on a depth camera, LiDAR, or radar. The central idea is to leverage the reference sizes for a typical human and their image size to perform depth estimation through camera equations. Nevertheless, in cluttered environments with occlusions or situations where the human is not standing in a conventional pose, this can result in a significantly faulted estimation. The motivation for the first research is to use pose estimation as a means

to overcome these obstacles. Thus, research was conducted with human participants after approval from the Human Research Ethics Committee (HREC) of the Delft University of Technology (letter of approval no. 3457, issued on September 26, 2023).

Research Question 1

Is leveraging pose estimation techniques and camera equations a robust and accurate approach to performing human depth estimation in photographic images?

The second research question refers to the task of tracking the SaR victims based on their previous motion patterns and measurements from both robots. Therefore, unobstructed and cluttered scenarios representing different trajectories are recreated to assess the improvements of a joint operation compared to a standalone deployment for each of the robots.

Research Question 2

How much improvement does a UGV-UAV collaboration offer in performing object tracking compared to an individual operation within realistic SaR scenarios?

The third research question addresses the potential to also provide information regarding the elevation of the terrain. This is particularly helpful to the SaR human agents in planning their mission. While this information can be obtained through distance measurement sensors such as LiDAR or radar, the distinct viewpoints of the UGV and UAV can also be used to estimate it.

Research Question 3

To what extent can a collaborative UGV-UAV system generate a dependable and comprehensive elevation map of diverse terrains?

The research challenges follow logically from one to the other and cover a spectrum of technical and operational aspects to indicate how advantageous heterogeneous teams of robots are in disaster response. Successful answers to these questions can lead to the development of more cost-effective SaR robotic systems that can operate in demanding environments, ultimately improving the effectiveness of disaster response efforts.

1.4. Report Structure

The report details the outcomes of the research endeavor structured around the defined objectives and subsequent research questions. Chapter 2 encapsulates the primary findings, elucidating methodologies, experimental case studies, and the results obtained, as well as limitations of the work, suggestions for future work, and conclusions. Chapter 3 provides a literature review, synthesizing past research and identifying research gaps, some of which led to the research formulation. The conclusions of the work are outlined in Chapter 4 by revisiting the research questions and addressing them. The study illuminates significant insights, paving the way for future investigations in the field.

Part I

Scientific Article

Flying and Ground Robot Collaboration for Camera-based Search and Rescue

Bernardo Esteves Henriques
Control & Operations Department (TU Delft), Delft, the Netherlands

***Abstract*—Search and Rescue (SaR) missions present challenges due to the complexity of the disaster scenarios and the urgency to rescue the victims. Most life losses and injuries occur in developing countries where the budget is scarce. Robotics has become indispensable for rapidly locating disaster victims. Combining flying and ground robots more effectively serves this purpose due to their complementary features in terms of viewpoint. To this end, a financially cost-effective framework to perform typical SaR tasks is presented. The method leverages You Only Look Once (YOLO) and video streams transmitted by an Unmanned Ground Vehicle (UGV) and an Unmanned Aerial Vehicle (UAV). Three sets of experiments were conducted at the Cyber Zoo of the Delft University of Technology. In exploiting pose estimation to perform human depth estimation, the experiments unveiled the susceptibility of the proposed approaches to variations in poses. In tracking moving object trajectories, the collaboration was found to be particularly advantageous in wide-area cluttered trajectories as opposed to narrow-area unobstructed trajectories where the deployment of one robot suffices. In mapping terrain elevation, relative errors dropped significantly with the collaboration of the UGV and the UAV. Moving forward, refining algorithms, enhancing collaborative functionalities, and devising adaptable strategies tailored to diverse SaR scenarios will be pivotal.**

***Index Terms*—Search and Rescue Robotics, Computer Vision, Object Detection, Pose Estimation, Homography Estimation, State Estimation, Terrain Mapping.**

ACRONYMS

CNN Convolutional Neural Network
COCO Common Objects in COntext
DLT Direct Linear Transform
EKF Extended Kalman Filter
EMA Exponential Moving Average
FOV Field of View
GPS Global Positioning System
HREC Human Research Ethics Commitee
KF Kalman Filter
RANSAC RANdom SAmple Consensus
RMSE Root Mean Square Error
SaR Search and Rescue
SIFT Scale-Invariant Feature Transform
UAV Unmanned Aerial Vehicle
UGV Unmanned Ground Vehicle
UKF Unscented Kalman Filter
YOLO You Only Look Once

NOMENCLATURE

α	Smoothing factor
γ	Angle to SaR victim
ω	Angular velocity
d	Depth
d_{ground}	Ground distance
e	Elevation
$\mathbf{F}$	Transition matrix
FOV_h	Horizontal FOV
FOV_v	Vertical FOV
$\mathbf{H}$	Homography matrix
$\mathbf{H}^*$	Optimal homography matrix
h_b	Bounding box height
k	Current time step
$\mathbf{K}$	Kalman gain matrix
l_{focal}	Focal length
$\mathcal{O}$	Observation matrix
$\mathbf{P}$	Covariance matrix
$\mathbf{Q}$	Process noise matrix
r	Radius
$\mathbf{R}$	Measurement noise covariance matrix
s_{image}	Image size
s_{real}	Real-world size
$\boldsymbol{U}$	Control input matrix
$\boldsymbol{v}$	Measurement noise vector
w_b	Bounding box width
$\boldsymbol{x}$	System states
$\hat{\boldsymbol{x}}k^{\text{EMA}}$	Predicted system states via EMA
$\hat{\boldsymbol{x}}k^{\text{KF}}$	Predicted system states via KF
$\boldsymbol{z}$	Measurement vector

I. INTRODUCTION

Natural disasters are a significant and growing threat to human health and safety, with an increasing frequency and cost of damage. In recent decades, the number and severity of natural disasters have risen dramatically, highlighting the need for effective disaster management [1]. Thus, optimizing search and rescue (SaR) operations has become an urgent and timely research topic in academia and industry due to their importance in mitigating further life losses and injuries. In the case of the least developed countries, the risk is particularly concerning, as nearly 90% of disaster-related deaths and 98%

of people affected by disasters between 1991 and 2005 occurred in these countries [2]. Therefore, it is vital to make SaR operations not only more reliable, but also more affordable.

In the past twenty years, SaR has become increasingly augmented with robotics [3]. Besides being able to traverse hazardous environments, robots have the capability of mapping their surroundings in a fast and automated way. This competence makes them vital in disaster response effectiveness since survival rates drop steeply in the aftermath of a catastrophe. For instance, the survival rate of a very strong earthquake can drop from 91% in the first thirty minutes to 36.7% at the end of the second day [4]. Therefore, even if the robots do not perform the actual rescue themselves, providing a reliable map of the location of the victims speeds up the mission planning and increases the chances of success. Table I displays qualitative cost comparison of conventional sensors included in robotic SaR setups.

TABLE I
SAR MEDIUM-SEGMENT SENSOR QUALITATIVE COST COMPARISON [5]–[11].

Sensor	Cost
RGB Cameras	Low
Depth Cameras	Low
Acoustic Sensors	Low
2D Radar	Medium
2D LiDAR	Medium
Thermal Imaging	Medium
3D Radar	High
3D LiDAR	High

Even though the different sensors have different principles and functionalities, this research focuses solely on the usage of RGB cameras as it is a promising sensor type to deliver a cost-effective SaR solution. Furthermore, combining the advantages of flying and ground robots for mapping the location of the victims in SaR is promising due to the complementary capabilities of these robots. Unmanned Aerial Vehicles (UAVs) provide affordable access to aerial data and are known for their ability to scan large areas in a short time. However, their camera is subject to tilting motions that often result in reduced image quality. Unmanned Ground Vehicles (UGVs) are typically slower at covering large areas due to rough terrain and a smaller angle of view. Nevertheless, they are better suited to carrying heavy weights, accessing confined spaces, and taking focused photographs. The research is thus motivated by the potential to combine heterogeneous robots in SaR applications and the urgent need for accessible SaR solutions. A graphical representation of a heterogeneous robot setup is provided in Figure 1.

Throughout the research, the setup performs three different typical SaR tasks, namely SaR victim localization, trajectory tracking, and elevation estimation. An assessment is then provided regarding the accuracy of the proposed setup in performing the aforementioned tasks.

II. LITERATURE REVIEW

Before advancing to the contributions of the paper, it is essential to provide the background review that led to the research questions. In the current section, a summary of the literature review performed is given. This encompasses several considerations crucial to the domain of SaR, as well as background theory for a better understanding of the paper.

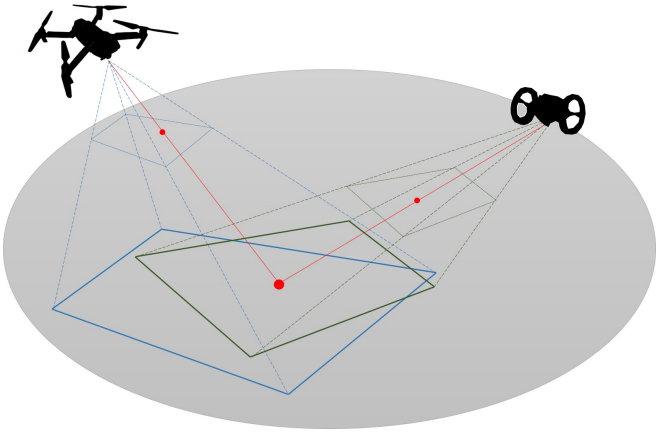


Fig. 1. Illustration of a possible UGV-UAV setup in which the two robots have their own specifications and thus cover different areas.

Prior work has focused on solving the navigation problem for collaborative teams of flying and ground robots (see e.g. [12], [13]). Figure 2 showcases an example of the distinct viewpoints captured by flying and ground robots and hints at how they can complement each other. On the one hand, a SaR victim under a roof might not be detected by a flying robot, but be detected by a ground robot. On the other hand, the opposite can happen to a SaR victim behind a wall. Therefore, it is likely that the heterogeneous robots working together can detect more victims than any of the robots working alone.

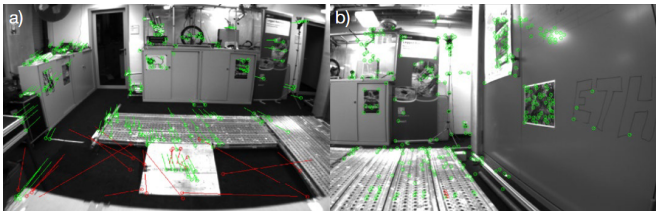


Fig. 2. Example of change in viewpoints for a flying robot (left) and a ground robot (right). Retrieved from [12].

Some studies have focused on processing thermal imaging to detect humans in SaR scenarios (see e.g. [14], [15]). Even though the results are auspicious in specific scenarios, they have not proved to be helpful in environments where high temperatures are reached, as in the case of fire. If the environment is emitting higher radiation than the heat emanating from a human body, the human is not detected. There are also studies that inspect wave properties, such as by Doppler-shift sensors, to detect humans based on the motion of their lungs, their heartbeat, or typical Doppler signatures of walking humans (see [16]–[18]). Nevertheless, these sensors are primarily designed to detect motion. In SaR scenarios, it is essential to identify both moving and stationary targets. Furthermore, distinguishing human targets from other moving objects solely based on Doppler readings can be challenging

and can lead to false positives. It becomes evident that, due to the complexity of SaR scenarios, it is vital to include visual inspection besides other sensors when challenging conditions are met.

Latest advances in deep learning have attracted attention to human detection through image processing tasks. Deep learning models can automatically extract relevant features from raw data and detect objects even in the presence of challenging conditions such as low lighting, occlusions, and cluttered backgrounds. Their ability to handle high-dimensional data, such as videos, makes them advantageous compared to traditional computer vision methods. Convolutional Neural Networks (CNNs) play a vital role in modern object detection due to their ability to exploit the spatial structure of images and to learn translation-invariant representations of objects. Instead of treating each pixel in an image as a separate input, convolutional layers are used to learn spatial relationships between pixels through kernels, enabling CNNs to recognize more complex features. Several studies (see e.g. [19]–[21]) have shown encouraging results in detecting people in videos in the context of SaR using CNNs and its variants.

YOLO (You Only Look Once), existing as part of the CNN family, has gained prominence as an effective object detection algorithm in recent years [22]. As it uses a single neural network to detect and classify objects, the training and deployment process is simplified. It can achieve real-time object detection while maintaining high accuracy and requiring less computational resources than other CNN variants. The YOLO algorithm has already proved its effectiveness in detecting SaR victims or disaster-affected objects. In [23], pre-trained deep learning models based on YOLO are used to detect a flooded area, which is then tracked and mapped using homography estimation (further elaboration is provided in the next paragraph). In [24], YOLOv5 is deployed to detect and localize humans in an outdoor SaR environment, whereas in [25] YOLOv4-tiny, a compact version of YOLOv4, is employed to detect victims and to estimate whether they are standing, laying vertically or laying horizontally.

In the context of SaR, it is vital to detect individuals in distress while also estimating their location and mapping it accordingly. To the best of available knowledge, the only works that focused on simultaneously detecting and localizing objects in SaR scenarios while using YOLO were discussed in [23] and [24]. In the case of [23], the Scale-Invariant Feature Transform (SIFT) key point matching algorithm is deployed to find corresponding points in pictures taken from different viewpoints. After determining the correspondences, a homography matrix is computed to transform coordinates from one frame to the other. Homography estimation is a technique used in computer vision to find the relationship between two images of the same scene but captured from different viewpoints. As a prerequisite, it is necessary to know the coordinates of the corresponding points in both frames. This can be challenging in cases where the images have limited visual features, occlusions, or significant changes of viewpoint. Nevertheless, homography estimation provides a way to model planar transformations accurately, is known to be computationally efficient, and allows for real-time processing. The approach used in [24] differs from the one in [23] in the determination of the corresponding points within distinct frames. More specifically, instead of using SIFT to make correspondences, trigonometry is used.

When localizing humans in SaR scenarios, estimating their pose can be key. In cluttered environments where only some body parts of a human are visible, pose estimation can be extended to provide an estimate of where the invisible key points are situated in the image. Apart from being valuable for depth estimation, knowing whether a victim is standing or lying on the floor can hint at their health state. It can also be useful to determine when a victim is pointing in a certain direction, for instance, as it can indicate an area of interest. In [26], the relative position between 3D key points is used for human action recognition.

For the SaR team, it is not only valuable to receive information about the position of the victims at each moment in time but also tracking information which includes their speed to estimate where they will be in the following time frames. For this purpose, state estimation can be performed to fuse position measurements from different sensors, as well as to apply models that describe the typical motion conducted by humans. The Kalman Filter (KF) is the most used state estimation technique in such scenarios due to its remarkable ability to combine noisy sensor measurements and dynamic models seamlessly, providing accurate and efficient real-time state estimates. This technique is applied in [27] to track the position of multiple humans, even when they are momentarily occluded. If the motion of the object to be tracked is highly non-linear, using more sophisticated filters, such as the Extended Kalman Filter (EKF) or the Unscented Kalman Filter (UKF), can yield better results. In [28], a comparison is drawn between using a KF and a UKF in tracking a human head, proving that the UKF has a better performance at visual curve tracking, namely in surveillance tasks, at the expense of increasing computational complexity.

Another convenient piece of information for SaR teams is the traversability and elevation of the terrain. In [29], a technique for extracting the ground characteristics (including the roughness, slope, discontinuity, and hardness) from images using neural networks, in order to determine traversability is presented. While the extraction of ground characteristics from images using neural networks can certainly provide valuable information about terrain traversability, the effectiveness of this approach can be significantly enhanced when combining the distinct viewpoints of UGVs and UAVs. In addition to getting more coverage of the terrain, the collaboration of the robots is also less likely to be misled by optical illusions.

III. Main Contributions & Structure of the Paper

The main contributions of the research stem from the following three sources:

1) The pose estimation module is extended to estimate the location of the key points that are missing from an image based on body proportions and symmetry. YOLO is leveraged to perform depth predictions based on reference human body measurements. A case study involving participants is conducted.

2) YOLO is employed in streams from a UGV and a UAV, along with a localization algorithm, to map the detections into real-world coordinates. The measurements are fused through a KF and are tested in diverse scenarios through performance metrics.
3) An algorithm is created to estimate the elevation of the terrain based on YOLO and homography estimation. It is tested via several lab experiments for different elevations and distances from the camera.

The rest of the paper is organized as follows: Section IV addresses the proposed research methodologies in detail. This encompasses the developed and adopted techniques to detect, localize, and track humans, as well as to map the terrain elevation. Subsequently, Section V refers to case studies used to validate and support the developed approaches and algorithms using real-world data, as well as to obtain experimental results. As such, it addresses the equipment used, along with the experimental setup. Afterward, Section VI presents the results obtained based on the case studies and discusses them in light of the research questions. Reflections on the limitations of this work and possible topics for future research are provided in Section VII. Lastly, the main conclusions drawn from this study are given in Section VIII.

IV. Methodology

This section describes the methods developed for performing human detection and pose estimation, human localization, human tracking, and elevation map generation. Even though these methods can be extended for teams of multiple flying and ground robots, the study is focused on a team composed of a single UAV and a single UGV.

A. Human detection and pose estimation

Due to its ability to significantly speed up the class detection process while maintaining high accuracy, YOLO is the chosen computer vision algorithm for this research. This groundbreaking algorithm can simultaneously identify bounding boxes and class probabilities for objects in a single pass through a neural network. Having been the focal point of extensive research endeavors, it has undergone several iterations and improvements since its launch in 2015. In this research, YOLOv8, the most recent version of YOLO, is employed for both detection and pose estimation tasks. In particular, the YOLOv8n model representing the nano version was chosen due to being lightweight and faster than other YOLOv8 models. YOLOv8n was trained on the Common Objects in COntext (COCO) dataset comprising 330,000 diverse images [30]. Among these, 200,000 images are annotated for object detection, segmentation, and captioning tasks across 80 object categories, including common and specific objects. Annotations encompass bounding boxes, segmentation masks, and captions, enhancing the model's precision and versatility. Figure 3 depicts an example output of the YOLOv8n model for a front and top view of a human. Bounding boxes are drawn around detected classes, and a confidence score from 0 to 1 is attributed to each class. The bigger the confidence score, the more confident the algorithm is that the object in the picture matches the class. It is possible to define a confidence threshold for the detections. The algorithm does not flag a detection if the confidence score is lower than the confidence threshold.

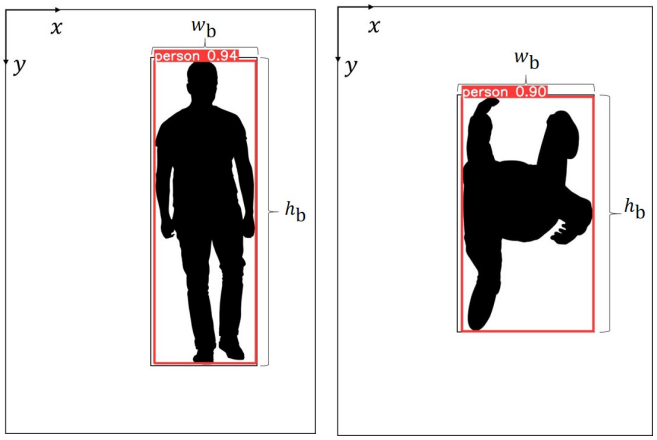


Fig. 3. Output of the YOLOv8n model for the front view (left) and top view (right) of a human.

In order to obtain pose information, an alternative model named YOLOv8n-Pose is employed whenever such information is required. This model is trained on a specialized version of the COCO dataset, designed for pose estimation tasks. This dataset contains 200,000 images labeled with key points for pose estimation tasks [30]. The dataset supports 17 key points for human figures, facilitating detailed pose estimation. Figure 4 depicts an example output of the YOLOv8n-Pose model for a front view of a human.

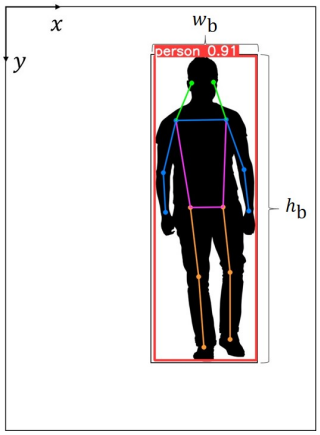


Fig. 4. YOLOv8n-Pose is extended to also perform pose estimation. Expanding on the previous output, it also estimates the positions of key points that represent body parts.

These key points are given in a pre-defined body part order whenever it is possible to estimate their location: nose, eyes, ears, shoulders, elbows, hands, hips, knees, and feet. Since the depth estimation module might rely on the location of a key point that is missing in a particular scenario, in this work, YOLOv8n-Pose was extended by the authors to also estimate the location of missing key points. This makes pose estimation more robust to occlusions. Figure 5 showcases the output of

the extended model in more challenging scenarios where some body parts are occluded.

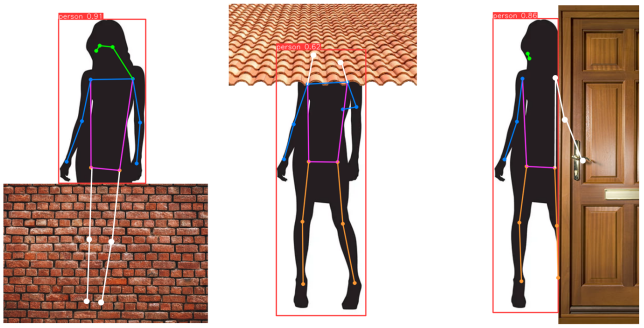


Fig. 5. During the study, YOLOv8n-Pose was further extended to estimate the location of missing key points based on typical body proportions and symmetry (represented in white). Examples of the output of the algorithm are given for a SaR victim behind a wall (left), under a roof (middle), and behind a door (right).

Building upon the output of the regular YOLOv8n-Pose, the algorithm loops through the entire key points vector to find key points that are missing and to estimate their location if the location of its dependencies is available. The procedural form of the algorithm is given below. In the experiments, n was set to 10 as this value is sufficient to determine all missing key points.

Algorithm 1 Complete key points

K: List of key points
N: Names of key points
n: Number of iterations
i: Index variable
$N \leftarrow$ get_keypoint_name()
for iteration $\leftarrow 1$ **to** n **do**
  for $i \leftarrow 0$ **to** length of K **do**
    if $K[i]$ is **None then**
      $K[i] \leftarrow$ estimate_keypoint(K, N, i)
    end if
  end for
end for
return K

The dependencies are the key points required to estimate the location of a particular key point and are defined a priori. As an example, the location of the right elbow can be estimated using the right hand and right shoulder, through body proportions, or using the location of the shoulders and the left elbow, through body symmetry. The high-level layout of the algorithm is given below.

B. Human localization

In the realm of computer vision and robotics, the accurate mapping of image pixels to real-world coordinates plays a pivotal role in enabling machines to perceive and interact with their surroundings.

1) Homography estimation: One indispensable technique in achieving a mapping from the image pixels to real-world coordinates, especially when dealing with planar surfaces, is homography estimation. In the context of this work, homography

Algorithm 2 Estimate key point

K: List of key points
N: Key points and dependencies
i: Index of current key point
k: Current key point
d: Dependencies of k
w: Weights for k
I: Indices of required key points
W: Selected weights
$i \leftarrow 0$
while $i <$ len(N) **do**
  $k \leftarrow N[i]$
  $d \leftarrow k[\text{"}dependencies\text{"}]$
  $w \leftarrow k[\text{"}weights\text{"}]$
  $deps\_satisfied \leftarrow$ True
  for $j \leftarrow 0$ **to** len(d) **do**
    if not all$(K[kp[\text{"}idx\text{"}]]$ not None for dep in $d[j]$ for kp in K if $kp[\text{"}name\text{"}] == dep)$ **then**
      $deps\_satisfied \leftarrow$ False
      break
    end if
  end for
  if $deps\_satisfied$ **then**
    break
  end if
  $i \leftarrow i + 1$
end while
if $i =$ len(N) **then**
  return err("Cannot estimate key points.")
end if
$I \leftarrow$ [indices of required key points]
$W \leftarrow w$
return comb(K, I, W)

estimation serves as a valuable auxiliary method, facilitating the transformation of 2D image points into their corresponding real-world coordinates relative to robotic platforms. The choice of employing homography estimation stems from its ability to handle perspective transformations, making it particularly well-suited for scenarios where images captured by cameras mounted on robots need to be translated into a coherent real-world reference frame. However, homography estimation relies on certain key assumptions, including the coplanarity of reference points and the absence of lens distortion, which are essential considerations when applying this method in practical applications.

The homography transformation of 2D points is performed using a 3 ×3 matrix, commonly represented as $\mathbf{H}$ and known as the homography matrix (see (1)). In (1), we have $\tilde{h}_{ij} = h_{ij}/h_{33}$ for $i, j = 1, 2, 3$, where h_{33} can be set equal to 1 without losing the generality because of the homogenous property of the projection.

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = h_{33} \begin{bmatrix} \tilde{h}_{11} & \tilde{h}_{12} & \tilde{h}_{13} \\ \tilde{h}_{21} & \tilde{h}_{22} & \tilde{h}_{23} \\ \tilde{h}_{31} & \tilde{h}_{32} & 1 \end{bmatrix} \quad (1)$$

Note that (1) shows that there are eight unknowns in the matrix that represent eight (three rotational, three translational, and two scalars) degrees of freedom. The eight equations needed to compute these eight unknowns are constructed by selecting four reference points for which the corresponding x-coordinates and y-coordinates in both camera view and augmented view are known. Suppose a set of N points $P_1 = \{(x_i, y_i)$ for $i = 1, 2, \ldots, N\}$ that are coplanar in the camera view. Also, consider the set of these points when they have been represented in the augmented view, i.e., $P_2 = \{(x_i^{''}, y_i^{''})$ for $i = 1, 2, \ldots, N\}$. For each point, (2) is used to obtain two linear equations by applying an approach known as Direct Linear Transform (DLT):

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i'' & -y_i x_i'' & -x_i'' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i'' & -y_i y_i'' & -y_i'' \end{bmatrix} \begin{bmatrix} \tilde{h}_{11} \\ \tilde{h}_{12} \\ \vdots \\ \tilde{h}_{32} \\ \tilde{h}_{33} = 1 \end{bmatrix} = \mathbf{0} \tag{2}$$

When dealing with more than four reference points, computing $\mathbf{H}$ transforms into an optimization problem. The objective is to determine an optimal homography matrix $\mathbf{H}^*$ that minimizes a defined cost function. The cost function typically involves measuring geometric distances, like the Euclidean distance, between the projected points $(\tilde{x}_i'', \tilde{y}_i'')$ obtained using $\mathbf{H}$ and the actual corresponding points (x_i'', y_i'') in the map view. The optimal homography matrix $\mathbf{H}^*$ minimizes the algebraic sum of the geometric distances across all reference points. However, when dealing with outliers in the reference points, like non-coplanar or erroneous points, the accuracy of estimating matrix $\mathbf{H}$ is compromised. To address this, the RANdom SAmple Consensus (RANSAC) algorithm is employed. RANSAC randomly selects subsets of reference points, computes $\mathbf{H}$ using the DLT method, and classifies points as inliers or outliers based on the cost function (e.g., geometric distance). It iteratively refines $\mathbf{H}$ using only the inlier subset, ensuring a robust estimation of $\mathbf{H}$ even in the presence of outliers.

Given $\mathbf{H}$, P_1 in camera view can be transformed into P_2 in augmented view using the following equation from [23]:

$$\begin{bmatrix} x_i^{''}/\lambda \\ y_i^{''}/\lambda \\ \lambda \end{bmatrix} = H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3}$$

where λ is a scaling factor that is introduced in homography transformations. It represents the transformation's scale factor and ensures that the transformed points maintain their relative positions after the homography transformation.

2) Depth estimation: Another vital approach to estimating real-world coordinates from 2D points in photographic images is depth estimation. Under a simplified pinhole camera model, it is possible to deduct straightforward equations that relate camera specifications to distances. The camera is assumed to be calibrated, and the parameters such as focal length, reference object size, and field of view are known or can be accurately measured.

Note that (4) allows us to estimate the depth d of an object from its real-world size s_{real}, the focal length of the camera l_{focal}, and the size of the object in the image s_{image}:

$$d = \frac{s_{\text{real}} l_{\text{focal}}}{s_{\text{image}}} \tag{4}$$

Also note that (4) can be rearranged to calculate s_{real} of an object based on s_{image} and d:

$$s_{\text{real}} = \frac{d s_{\text{image}}}{l_{\text{focal}}} \tag{5}$$

where l_{focal} of the camera lens is usually provided by camera manufacturers. Alternatively, l_{focal} can be computed based on the image dimensions and the field of view (FOV) of the camera. An illustration of the vertical and horizontal FOV of a camera is given in Figure 6.

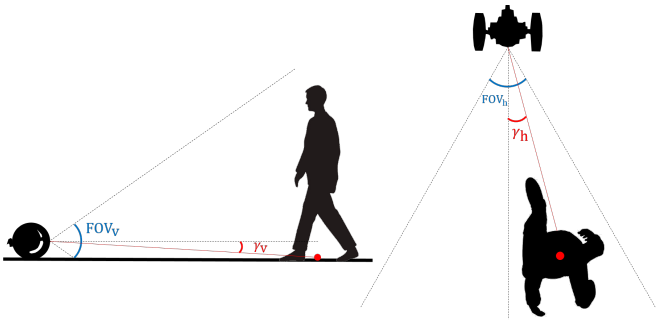


Fig. 6. A sketch of the vertical v (left) and the horizontal h (right) FOV of a camera, where γ represents the angle between the camera and the SaR victim.

These values can be substituted in (6) or (7) to determine l_{focal} of a camera, in case it is not provided. In these equations, H_{image} and W_{image} represent, respectively, the height and the width of the image in pixels:

$$l_{\text{focal}} = \frac{H_{\text{image}}}{2 \tan\left(\frac{\text{FOV}_\text{v}}{2}\right)} \tag{6}$$

$$l_{\text{focal}} = \frac{W_{\text{image}}}{2 \tan\left(\frac{\text{FOV}_\text{h}}{2}\right)} \tag{7}$$

C. Human tracking

State estimation is a fundamental requirement in several SaR contexts, as it enables one to make inferences about the states of a dynamic system based on noisy sensor measurements. The primary objective of state estimation in this study is to infer the trajectory of a moving SaR victim.

The two sources of position measurement are obtained from the distinct robotic platforms: a ground robot and a flying robot. Each position 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 can generally be measured with the support of a position-determination system, such as the Global Positioning System (GPS). The position of the victim relative to each robot can be computed using (3) and (4). A visual representation of the

procedure is available in Figure 7. In the present study, the position of the robots is assumed to be known at all times with no error.

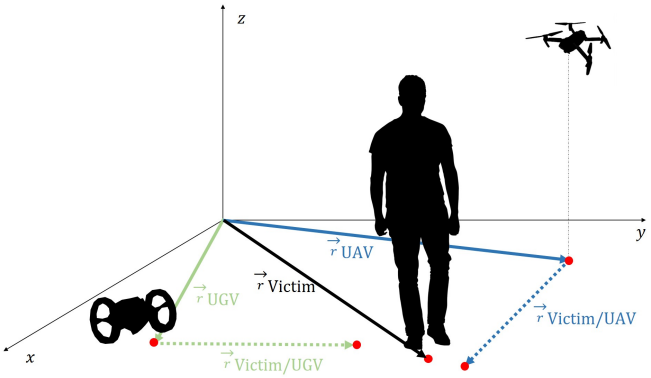


Fig. 7. There are two sources of measurement for the location of the victim, one per robot. The estimated location at each time step is the vector sum of the position of the robot and the position of the victim relative to the robot.

A KF is then employed to fuse the measurements with a motion model of a human agent. By incorporating knowledge of how the system is expected to evolve over time, the filter can make predictions about future states, even when no new measurements are available. Furthermore, a KF ensures that the estimated states evolve realistically according to the model, providing a continuous and coherent trajectory. However, formulating a motion model for humans in SaR scenarios is extremely challenging due to the unpredictability of their movement. Therefore, a simplistic linear motion model has been adopted, which assumes that agents have no acceleration if there are no measurements available. The formulation of the state estimator is divided into the state transition model, observation model, as well as the prediction, update, and filtering steps, as follows:

*1) **State Transition Model**:* The State Transition Model describes how the state of a system evolves over time. The state vector, denoted as $\boldsymbol{x}_k$, represents the position and velocity in a 2D space at time step k:

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} \tag{8}$$

This vector encapsulates both the current position and velocity in the x and y directions, given that the superscript $\dot{}$ represents the time derivative. The State Transition Model predicts the next state, $\boldsymbol{x}_{k+1}$, based on the current state $\boldsymbol{x}_k$:

$$\boldsymbol{x}_{k+1} = \mathbf{F}\boldsymbol{x}_k \tag{9}$$

where $\mathbf{F}$ represents the state transition matrix, which characterizes how the system evolves over time. It is important to note that in this specific system model, there is no utilization of a control input matrix $\boldsymbol{U}$, meaning that state evolution from a time step to the next is solely determined by the inherent dynamics defined by $\mathbf{F}$ and no external control inputs $\boldsymbol{u}_k$ are incorporated.

*2) **Observation Model**:* The Observation Model relates the measurements obtained from sensors to the underlying state of the system. The measurement vector, denoted as $\boldsymbol{z}$, comprises position measurements:

$$\boldsymbol{z}_k = \begin{bmatrix} x_k^{\text{measured}} \\ y_k^{\text{measured}} \\ \dot{x}_k^{\text{measured}} \\ \dot{y}_k^{\text{measured}} \end{bmatrix} \tag{10}$$

These measurements are subject to noise and uncertainties. The Observation Model establishes a connection between the state vector, $\boldsymbol{x}_k$, and the noisy measurements, $\boldsymbol{z}_k$:

$$\boldsymbol{z}_k = \mathcal{O}\boldsymbol{x}_k \tag{11}$$

where $\mathcal{O}$ represents the observation matrix, which defines how the state relates to the measurements. A measurement noise vector $\boldsymbol{v}_k$ may also be incorporated in this formulation.

*3) **Prediction Step**:* The KF operates in two main steps: prediction and update. The prediction step estimates the future state of the system based on the current state and any available control inputs. In the following equations, the superscript $^-$ represents the variable before the update step is performed.

a) State Prediction: The predicted state, $\hat{\boldsymbol{x}}_{k+1}^-$, is calculated using the State Transition Model:

$$\hat{\boldsymbol{x}}_{k+1}^- = \mathbf{F}\hat{\boldsymbol{x}}_k \tag{12}$$

This step provides an initial estimate of the future state of the system.

b) Error Covariance Prediction: The error covariance matrix, $\mathbf{P}_{k+1}^-$, estimates how uncertain the state prediction is. It is determined by propagating the current error covariance, $\mathbf{P}_k$, through the State Transition Model and by adding the process noise, $\mathbf{Q}$:

$$\mathbf{P}_{k+1}^- = \mathbf{F}\mathbf{P}_k\mathbf{F}^T + \mathbf{Q} \tag{13}$$

where $\mathbf{P}_{k+1}^-$ quantifies the uncertainty associated with the state prediction.

*4) **Update Step**:* The update step of the KF refines the state estimate based on new measurements, in order to reduce the uncertainty and improve the accuracy.

a) Kalman Gain Calculation: The Kalman gain, $\mathbf{K}_k$, determines how much weight to give to the measurements in the state update at time step k. It depends on the error covariance prediction, $\mathbf{P}_k^-$, the observation matrix, $\mathcal{O}$, and the measurement noise covariance, $\mathbf{R}$:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathcal{O}^T(\mathcal{O}\mathbf{P}_k^-\mathcal{O}^T + \mathbf{R})^{-1} \tag{14}$$

The Kalman gain reflects the relative importance of the prediction and measurement information.

b) State Update: The state estimate, $\hat{\boldsymbol{x}}_k^{\text{KF}}$, is updated using the Kalman gain and the difference between the actual measurements, $\boldsymbol{z}_k$, and the predicted measurements, $\mathcal{O}\hat{\boldsymbol{x}}_k^-$:

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \mathbf{K}_k(\boldsymbol{z}_k - \mathcal{O}\hat{\boldsymbol{x}}_k^-) \tag{15}$$

This step combines the prediction and the measurement information to yield a more accurate state estimate.

c) *Error Covariance Update*: The error covariance matrix, $\mathbf{P}_k$, is updated to reflect the reduced uncertainty after incorporating the measurements:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathcal{O})\mathbf{P}_k^- \tag{16}$$

This matrix quantifies the remaining uncertainty in the state estimate, accounting for the influence of the measurements.

*5) **Filtering Step**:* To further refine and smoothen the output obtained from the KF, an Exponential Moving Average (EMA) filter is employed. This additional step helps to provide a more stable representation of the estimated system state by giving the output state variables from the KF as input to the EMA. Unlike a simple moving average, which gives equal weights to all data points in the averaging window, EMA assigns exponentially decreasing weights to older data points. This characteristic allows the EMA to react more quickly to recent changes in the data while still incorporating information from the past.

The state prediction is filtered using the following relationship, where $\hat{\boldsymbol{x}}_k^{\text{KF}}$ represents the state as predicted by the KF and $\hat{\boldsymbol{x}}_k^{\text{EMA}}$ represents the state as predicted by the EMA:

$$\hat{\boldsymbol{x}}_k^{\text{EMA}} = \alpha\hat{\boldsymbol{x}}_k^{\text{KF}} + (1-\alpha)\hat{\boldsymbol{x}}_{k-1}^{\text{EMA}} \tag{17}$$

where α represents the smoothing factor of the EMA filter. A higher value of α gives more weight to recent data points, making the EMA more responsive to recent changes. Conversely, a lower α value results in a smoother EMA curve.

D. Elevation map generation

The proposed approach to determining terrain elevation is divided into three steps illustrated in Figure 8.

Firstly, the UGV needs to receive information about its straight-line distance d_{ground} to the target object on the ground. The UAV might be able to provide this information by leveraging homography estimation and trigonometry if both the UGV and the target object are in its FOV. If not, the UGV has to estimate this distance, making use of depth estimation that is formulated via (4). After obtaining this knowledge, the UGV proceeds to calculate the expected position of the specified point in the camera frame under the assumption of perfectly flat ground. With the expected position in hand, the UGV then undertakes a critical comparison. It contrasts the anticipated position, based on the assumption of a flat terrain, with the actual perceived position of the touchdown point as observed from its ground-level perspective. The outcome of this comparison is used to obtain the elevation e, which essentially represents the vertical deviation of the terrain from the hypothetical flat surface, via the formulation in (5). By systematically mapping this e at various points across the terrain, the UGV generates a comprehensive elevation map of the area.

Applying this methodology is particularly challenging when the object sits close to the UGV or at a high elevation, as it is partially occluded, and thus affects the depth perception. By perceiving the object as smaller, the algorithm tends to deem it farther away, which decreases the estimated elevation. A visual depiction of this phenomenon is given in Figure 9.

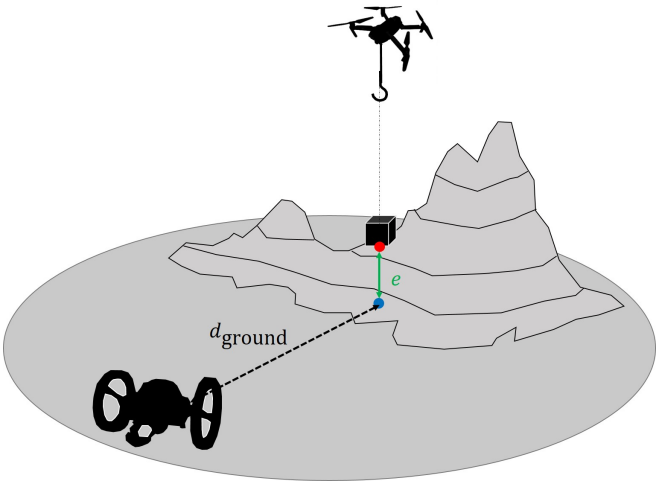


Fig. 8. In determining terrain elevation, the UAV drops an object where the elevation is to be determined. The UGV receives information about its ground distance d_{ground} to the UAV or determines the distance itself using depth estimation given by (4). This information is used to determine the point where the object would touch the ground if the ground were flat (this estimated point is represented in blue). Through the camera, the UGV catches sight of the point where the object actually touches the ground (represented in red). The pixel distance between these two points is used to compute the elevation e of the ground (represented in green) using (5).

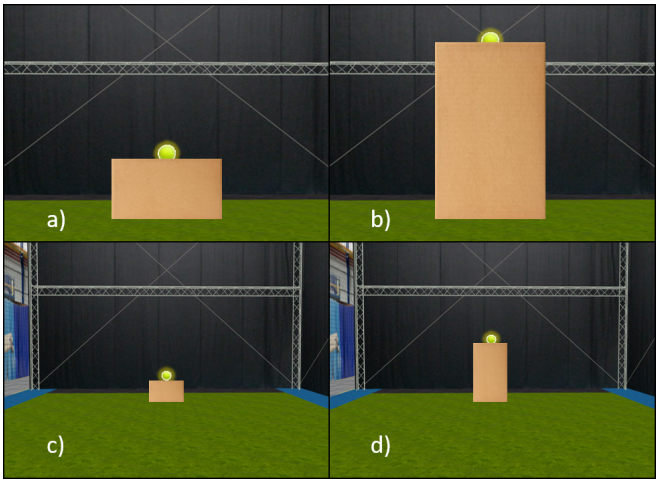


Fig. 9. When the object sits close to the UGV, as in a) and b) or is at a high elevation, as in b) and d), partial occlusion is more significant. In the image a) $d_{\text{ground}} = 1$ m, $e = 25$ cm, b) $d_{\text{ground}} = 1$ m, $e = 75$ cm, c) $d_{\text{ground}} = 3$ m, $e = 25$ cm, d) $d_{\text{ground}} = 3$ m, $e = 75$ cm.

To mitigate this effect, depending on the geometry of the object, it is possible to take the width of the object as a reference, as the width is only distorted in extreme scenarios.

V. Case Studies

This section describes the experimental setup employed during the research. Firstly, an overview of the laboratory

setting and the specifications of the robots are given, along with general considerations. In order to fulfill the research objectives, an experiment is conducted to estimate the depth of humans in photographs making use of their poses. The experiment, data collection process, and post-processing of the images are thus thoroughly described. Afterward, case studies encompassing tracking an object moving in circular trajectories with different radii in and out of the presence of obstacles were performed. The last task consists of making the UAV transport and drop an object where the elevation of the terrain needs to be determined. The UGV proceeds to determine the elevation after post-processing the collected video streams. In the experiments, challenging scenes that encompass occlusions and the temporary unavailability of the sensors are recreated.

A. General Experimental Setup

The experimental sessions have been conducted at the Cyber Zoo of the Delft University of Technology. Cyber Zoo (see Figure 10) is a research and test laboratory in the Faculty of Aerospace Engineering that embeds a 10 m $\times$ 10 m synthetic turf surrounded by safety nets to protect participants and robots during the experiments. Furthermore, the experimental facilities at the Cyber Zoo 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).

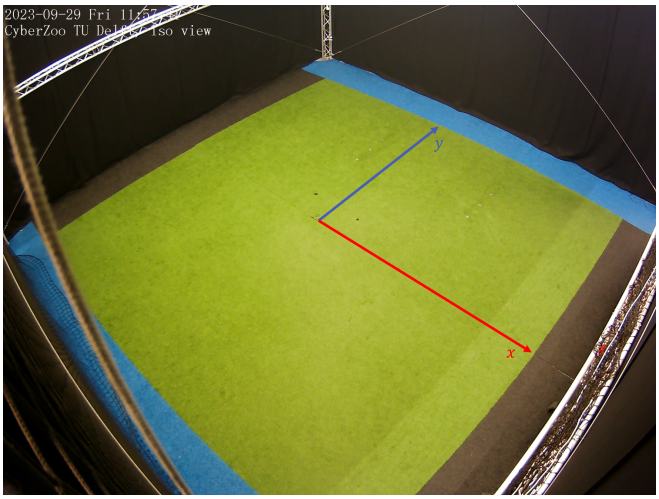


Fig. 10. Cyber Zoo including the reference frame. The origin corresponds to the center of the laboratory.

The drone used in the experiments was a Parrot Bebop 2 (see the right-hand side plot in Figure 11). 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 minutes on this battery power. The 14 MP front camera on Parrot Bebop 2 can record 1080p video at 30 fps. It is also equipped with a bottom camera used to estimate its velocity. This in-built camera is not suited to video recording, but another 14 MP bottom camera, which can also record 1080p video at 30 fps, has been attached to the bottom of the drone. The drone also has its own WiFi network, allowing it 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, this drone efficiently processes captured video and image data, ensuring smooth and high-quality visuals.

As for the ground robot, the Parrot Jumping Sumo was used. It is a compact, wheeled ground robot, forming part of the MiniDrones line by Parrot. Displaying a moveable base with two independently driven wheels, it measures 185 mm in length, 150 mm in width, and 110 mm in height. Weighing 180 g, Parrot Jumping Sumo features a 550 mAh battery, granting an operational time of up to 20 minutes. It can perform horizontal and vertical jumps up to 80 cm. The robot incorporates a wide-angle camera providing 480p 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 its agile movements and interactions.

The Parrot Jumping and the Sumo Parrot Bebop 2 drone are depicted in Figure 11.

Fig. 11. Parrot Jumping Sumo (left) and Parrot Bebop 2 (right).

Overall, the autonomy of the robots, the sufficient quality of their cameras, their affordability, their ready-to-use nature, and their user-friendly interfaces make them suitable for this research.

The processing of the photographs and videos was performed using an external computer. The external computer employed in this study 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. Storage was facilitated by a 512 GB PCIe NVMe SSD. The operating system utilized was Ubuntu 20.04.6 LTS. All the source code for the project was composed using Python, owing to its compatibility with external code, versatility, and robust libraries.

B. Experimental Human Depth Estimation

The primary objective of the experiments is to leverage pose estimation techniques to enhance the precision and to increase the robustness of human depth estimation. To achieve this goal,

a collection of static photographic representations of humans assuming diverse poses is deemed essential. Participants for this study were sourced from the personal network of the researchers. During the process of recruitment, an effort has been made to recruit participants with diverse physical characteristics, such as different heights, shoulder-to-shoulder and shoulder-to-elbow distances. This is crucial to ensure that the participants are somewhat representative of the generality of the population. As the experiments involve humans, they were conducted after approval from the Human Research Ethics Committee (HREC) of the Delft University of Technology (letter of approval no. 3457, issued on September 26, 2023). The experiment was performed on 24 participants of 12 different nationalities. The reference values used for height, shoulder-to-shoulder distance, and shoulder-to-elbow distance of a typical human were 1.70 m, 38 cm, and 30 cm in that order, based on [31], an article that contains anthropometric data for the US population between 2011 and 2014. The measurements are divided by gender, and in the research, the average of the male and female measurements was taken and rounded to the nearest integer cm. The participants were measured to obtain an idea of how closely they matched the reference values. Naturally, it is expected that a larger deviation from the reference values in the sample of participants yields a higher error for the algorithm that takes the reference values as guidelines. The measurements and reference values are displayed in Table II, where μ represents the average value and σ represents the standard deviation.

TABLE II
PARTICIPANTS MEASUREMENTS AND REFERENCE VALUES.

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

As can be seen from the table, μ for height, shoulder-to-shoulder distance, and shoulder-to-elbow distance are respectively, 3.16%, 7.24%, and 1.80% higher than the reference, whereas σ is 4.17%, 9.77%, and 7.14% of the value of μ in that order. Thus, the measurements from the participants match the reference relatively closely, with the shoulder-to-shoulder distance the least reliable measurement in terms of both μ and σ. While μ for shoulder-to-elbow distance is closer to the reference than that for the height, the worst-case measurement is still less reliable than that of the height due to its increased σ. On grounds of the challenge of determining these distances precisely in a conventional SaR scenario, the algorithm privileges using height, shoulder-to-shoulder distance, and shoulder-to-elbow distances owing to their decreasing absolute values.

During the experiments, each participant was asked to pose in 7 different poses for the camera (see Figure 12): standing facing the camera, standing 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 and 3 meters from the UGV.

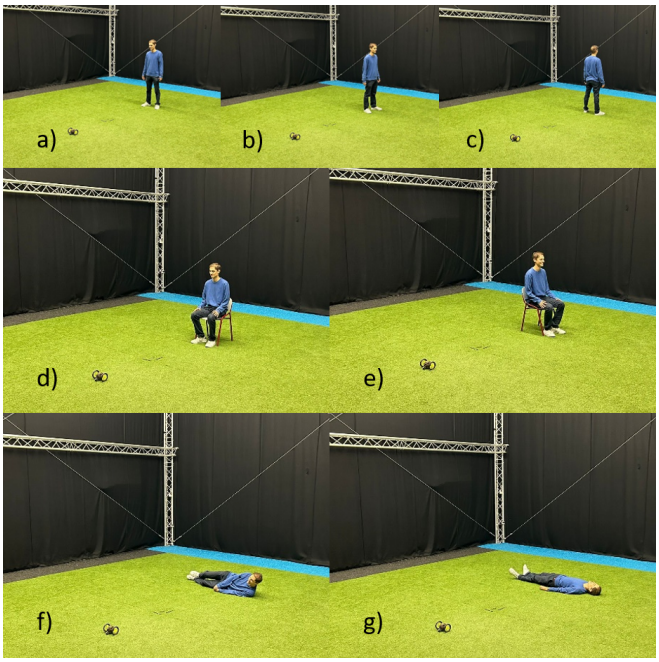


Fig. 12. The participants were instructed to adopt specific poses for the UGV camera. The photographs were captured systematically from varying distances of 1.5 m and 3 m. The sequence of poses encompasses a) standing facing the camera, b) standing in profile, c) standing back to the camera, d) sitting facing the camera, e) sitting in profile, f) lying on the side facing the camera, and g) lying face-up.

The experiments thus generated 14 photographs per participant and 336 photographs in total. All photographs are post-processed using the YOLOv8n-Pose extended model 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 Subsection IV-A. This ensures that all key points are estimated even when some body parts are not detected or are not captured by the camera. Distances between key points are then inserted into (4) to determine the depth. The equation compares a measured distance in the camera frame in pixels against a reference real-world value for that distance. As different poses distort different key point distances, the camera frame distance substituted in the equation depends on the pose. When the participant is standing or lying on the floor, the distance between the feet and the head is compared against a reference value for the human height. When the participant is sitting facing the camera or in profile, the shoulder-to-shoulder distance or the shoulder-to-elbow distance is compared against the reference values, respectively.

C. Experimental Object Tracking

The main purpose of this experiment is to track the movement of a real-world object by leveraging the video streams of both a UGV and a UAV. In a real-world SaR scenario, the object of interest to be tracked is most commonly a SaR victim. Nevertheless, due to the limited space at the experimental facility and due to the need to compare the obtained trajectory against the ground truth for the movement,

a small and precise trajectory had to be replicated. For this purpose, a volleyball was chosen elected as the object to be tracked. The volleyball was attached to a rod that was tied to a 360-degree motorized rotating stand. By moving the rod, the radius r of the trajectory can be manipulated. The motorized rotating stand allows for adjustable angles of rotation, directions of rotation, and angular velocity ω. During the experiment, different obstacles were placed around the trajectory to occlude both the aerial and the ground view of the volleyball. A visual depiction of the experiment is provided in Figure 13.

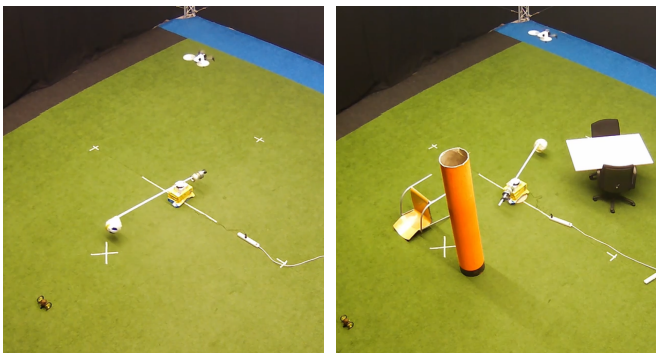


Fig. 13. The motorized rotating stand makes the volleyball move in a circle with constant angular velocity $\omega = 6$ rpm. During the experiments, the UGV and the UAV capture video streams of the movement of the ball, both in an unobstructed environment (left) and in a cluttered environment (right). The radius r of the trajectory is adjustable.

In all the experiments, the center of rotation is placed at the center of the Cyber Zoo $(x, y) = (0, 0)$. The UGV is kept static at $(x, y) = (0, -3)$, and the UAV hovers at a 3 m altitude as close as possible to the center of the Cyber Zoo $(x, y) = (0, 0)$. It is crucial to keep the altitude of the UAV approximately constant so that the homography matrix $\mathbf{H}$ does not require calibration after every frame. Since integrating GPS data is out of the scope of the project, DaVinci Resolve, a video editing software, was used to manually place the center of rotation in the center of the camera frame for each frame. By doing so, it is simulated that the tilting motion of the UAV in the x and y directions is compensated for and does not constitute an additional source of error. In a real-life situation, obtaining a similar compensating effect can be achieved using the real-time position of the UAV. Throughout the experiments, ω and the direction of the rotation were kept constant at 6 rpm counterclockwise. Five main settings were tested: $r = 0.75$ m, $r = 1.0$ m, and $r = 1.25$ m in an unobstructed environment, $r = 1.25$ m in a cluttered environment for the UAV only (i.e., the UGV view is unobstructed) and $r = 1.25$ m in a totally cluttered environment.

For each experiment, both video streams are post-processed frame by frame and off-board in an external laptop using the YOLOv8n model. In the case of the UGV, (3) and (4) are employed to determine the position of the volleyball in the reference frame depicted in Figure 10. As for the UAV, (3) is sufficient to determine the position of the volleyball. Since the UGV records videos at 15 fps and the UAV records videos at 30 fps, the frames are aligned before inputting the measured locations to the KF. Afterward, the measurements are taken as x^{measured} and y^{measured} in (10). The KF is updated at 30 fps, meaning that there are no measurements from the UGV in half of the instances, due to the halved frame capture rate. The estimated trajectory is then compared with the ground truth for each setting, making use of different performance metrics.

D. Experimental Elevation Map Generation

In the last experimental session, the UAV carries and drops a signaling object where the e is to be determined. In a real-world SaR scenario, this object is dropped in the vicinity of a victim to obtain information about the elevation and consequent reachability of the specific location. For this purpose, a remote air-dropping system was incorporated into the setup. During the experiments, a thrower is attached to the bottom of the UAV. Whenever the object needs to be released, a remote control is manually operated. Even though the maximum payload of the airdrop system is 750 g, any object weighing more than 150 g seriously jeopardizes flight stability. For this reason, a tennis ball weighing 58 g was selected as the signaling object. In the course of the experiment, the tennis ball was dropped on top of boxes with varying heights. A visualization of the experiment is given in Figure 14.

Fig. 14. The UAV carries and drops the tennis ball where the elevation of the terrain should be estimated. The UGV positions itself in an adequate perspective to estimate the elevation.

In order to assess the performance of the algorithm when the UGV is at different distances d_{ground} from the signaling object and when the elevation e changes, 12 photographs were captured using the camera of the UGV for $d_{\text{ground}} = 1, 2, 3, 4$ m and for $e = 25, 50, 75$ cm. These images were then processed off-board, simulating two different scenarios. In the first scenario, the UGV estimates the elevation on its own. Thus, it makes use of the YOLOv8n model and (4) to estimate d_{ground} and uses this estimation and (3) to determine the point where the signaling object would touch the ground if it was flat. It proceeds to compute the pixel distance between this point and the actual point where the ball touches the ground and applies (5) to determine the elevation. In a second scenario, it is assumed that the UAV computes d_{ground} from the sky, given its privileged viewpoint. To achieve this

computation, the YOLOv8n model needs to be further trained to also detect the Parrot Jumping Sumo, which is not the case in its default version. After leveraging (3) to transform both the signaling object and the UGV from the camera frame to the Cyber Zoo reference frame, d_{ground} is estimated using simple trigonometry. Nevertheless, given that additional training of the YOLOv8n model is out of the scope of this research and that the error in the computation of d_{ground} from the aerial perspective is expected to be negligible, it is assumed that the UGV receives the true value for d_{ground} with the assistance of the UAV. After this step, the UGV performs the subsequent computation steps, as in the case without assistance.

VI. Results & Discussion

The current section covers the results obtained from the proposed methodology and case studies, along with a discussion entailing the major findings.

A. Human Depth Estimation

After batch-processing all photographs, the average, maximum, and minimum depth, along with their upper and lower dispersion, were obtained. Out of the 336 photographs, in only 5, no humans were detected above a confidence threshold of 20%, amounting to a miss rate of 1.49%. For the remaining 98.51% of photographs, the results are divided by pose and distance from the camera and are exhibited in Figure 15.

From the plots, a few observations can be drawn. Firstly, the distance to the camera appears to have the biggest impact when the participant is standing. When the person is closer to the camera, the algorithm yields promising results given the proximity to the ground truth, the small upper and lower dispersion, and the non-substantial errors in the extremum. This is not the case for a larger distance from the camera, where these metrics degrade significantly. In cases where the participant is sitting or lying on the floor, the pattern seems to be consistent regardless of the distance to the camera. A possible explanation for this phenomenon is lens distortion. When assuming a simplified pinhole camera model, lens distortion is not accounted for. Lens distortion refers to optical anomalies in camera lenses causing deviations from ideal light projection. Standing positions, being farther from the image center, amplify distortion effects. Consequently, this impacts depth perception more than sitting or lying poses, exacerbating inaccuracies.

Secondly, the worst-case scenarios are predominantly situations where depth is estimated in excess rather than in defect. This is compatible with the observation that the measurements of participants are, on average higher than the reference values increasing the depth estimate. While the shortest participants are only slightly shorter than the reference, the tallest participants are significantly taller than the reference. Thus, this effect is expected to reverse for short ethnic groups and aggravate for tall ethnic groups. It is, therefore, advisable to adjust the reference values depending on the location where the SaR mission needs to be performed.

Thirdly, standing poses appear to be the least prone to errors on average. While it is true that their error pattern changes significantly with the distance from the camera, the average depths remain close to the ground truth. Sitting and lying poses appear to be more challenging in this domain. Two causes were identified for this phenomenon: poses where a smaller measurement is taken to compare against the reference are more prone to uncertainties (e.g. shoulder-to-shoulder distance as opposed to height), as well as poses where participants have a wider range of motion. When asked to stand, all participants stood still in a similar fashion. However, when sitting, they placed their legs more openly or closely and curved their back to different extents. The same goes for lying poses, where they tilted their bodies and stretched their arms and legs to different degrees.

To obtain a visual perspective on the average relative errors, a graph of the relative errors for each pose and distance to the camera across all participants and for one arbitrary participant is given in Figure 16.

Looking at the curves for all participants, it is observed that, as expected, the curves keep the same pattern except for the standing poses. It is also verified that sitting and lying poses generally display a bigger relative error than the standing poses. The slight discrepancy in the pattern for standing poses owes to the fact that when some of the participants were standing 1.5 m from the camera, their faces were not caught by the camera. Thus, the position of their faces was estimated using body proportions, giving rise to a new source of error. Nevertheless, sitting in profile and lying face-up still appeared to be the most challenging. The average across all poses for the average relative errors is 13.73% and 9.67% for camera distances of 1.5 m and 3 m, respectively. The explanation for this 4.06% discrepancy lies in the fact that participants did not always place themselves in the exact expected distance from the UGV. These slight displacements of a few centimeters have a higher impact on the average relative errors when the distance is smaller. Assuming that this is the only experimental random error unrelated to the algorithm, misplacement amounts to 12.18 cm on average. This seems reasonable and means that the actual relative error from the algorithm alone ranges from around 4% to 10%. Indeed, most of the relative errors sit between $<$1% to 12%. It is also noticeable that the relative error appears to have minimum to no correlation with the distance from the camera. In the case of the arbitrarily chosen participant, the relative error is suspected to arise from the physical discrepancies of the participant, as well as slight deviations from the conventional pose. This participant has a height of 1.83 m, shoulder-to-shoulder distance of 40 cm, and shoulder-to-elbow distance of 31 cm, which represents increases of 7.65%, 5.26%, and 3.33% with respect to the reference value, respectively. Therefore, poses relying on height, such as poses 1, 2, 3, 6, and 7, are more prone to errors compared to poses 4 and 5 for this specific participant.

As a final note, the average relative errors seem to be distance-independent, which means that the absolute errors vary linearly with the distance from the UGV. Relative errors can be as low as $<$1% but also as high as $>$20%, depending on the pose and the physical characteristics of the SaR victim. Therefore, depending on the requirements of the mission,

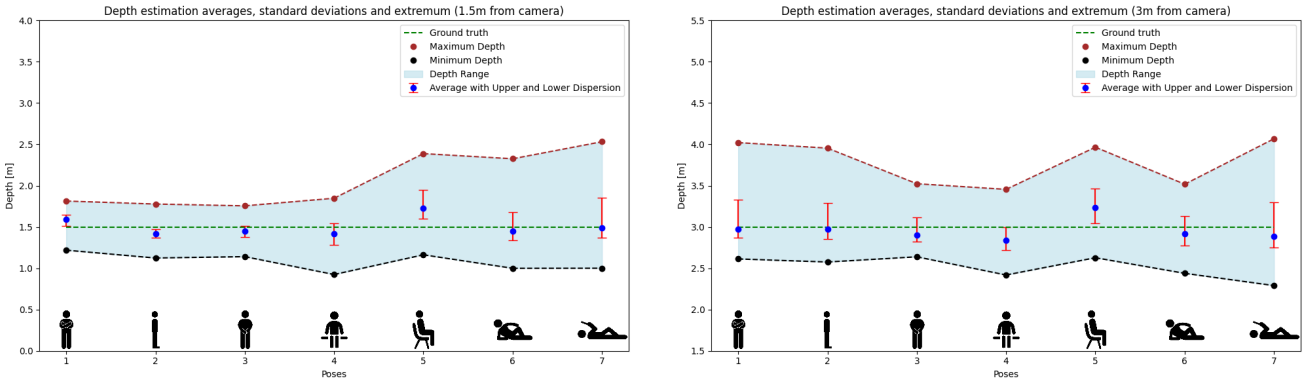


Fig. 15. Depth estimation averages, standard deviations, and extremum for 1.5m (left) and 3m (right) distance from the UGV.

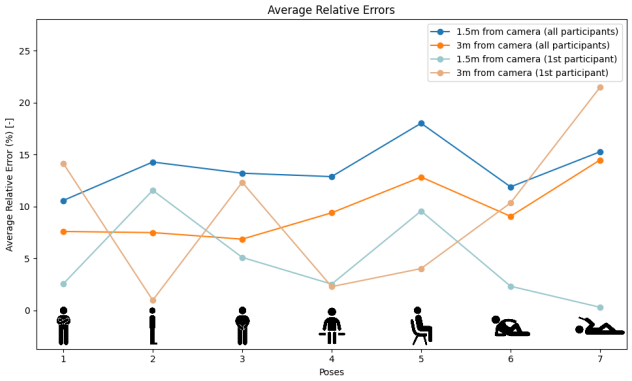


Fig. 16. Average relative errors for a 1.5m and 3m distance to UGV for an arbitrarily chosen participant and across all the participants.

the algorithm can prove to be sufficient or not. Taking into account that the photographs were taken in a well-lit and unobstructed environment, the results are expected to degrade in challenging SaR scenarios. For that reason, even though the results are promising to some extent, it is extremely likely that they require improvement before being applied in real life. Regarding the possibility of processing the photographs in real-time, in 10 runs of the algorithm, the 336 photographs took between 83.38 and 93.43 seconds to post-process, which averages 0.248 to 0.278 seconds per photograph. Since the photographs are captured at 15 fps, an acceleration of around 3.75x is required to achieve real-time processing. Ideally, this should be achieved through code optimization and not through more expensive hardware. Improvements on the proposed approach can also be achieved by adding a UAV to assist the UGV in detecting and localizing SaR victims, which leads to the following tracking experiment.

B. Object Tracking

Upon finishing the offline processing of all videos, the estimated volleyball trajectories were compared against the ground truth, employing different performance metrics. While each of the metrics gives an indication of how close a trajectory is to the ground truth, relying on a standalone metric proves to be ambiguous. For that reason, the results are analyzed by comparing the values for Root Mean Square Error (RMSE), extreme deviation, and area ratio. The RMSE shows how close the estimated volleyball paths are to the real ones on average. Extreme deviation spots where these estimates and the ground truth stray the farthest. The area ratio refers to the percentage of the area enclosed by the ground truth that is covered by the estimated trajectory. It thus checks how well the overall shapes match. Exploiting all three gives a better idea of how accurate the estimations are overall. Contrary to expectations, the obstacles proved to be too challenging for the UGV. The fact that the tracked object is a volleyball instead of a human increases the difficulty of the task, as the object is smaller and with a less distinctive pattern. Thus, in the presence of a cluttered environment with shadows, the UGV makes practically no detection, making it inapplicable to generate an estimated trajectory. For this reason, the analysis is performed for $r = 0.75, 1.0, 1.25$ m with unobstructed views and for $r = 1.25$ m with obstacles placed for the UAV only. The performance metrics for the UGV only, UAV only, and UGV-UAV collaboration in these scenarios are shown in Table III, where the best performance for each metric and scenario is highlighted in bold. To make the analysis more visual, the performance metrics should be accompanied by a visual inspection of Figures 17, 18, 19, and 20, which depict the estimated trajectories for $r = 0.75, 1, 1.25$ m and for $r = 1.25$ m with obstacles for the UAV, respectively. As expected, the metrics tend to degrade with increasing values of r and in cluttered environments.

As can be seen from Table III, the RMSE for the UGV-UAV collaboration sits between the RMSE for each robot alone in the four scenarios. Since the RMSE should ideally be as low as possible, the UGV-UAV seems to be a compromise between the two robots alone. On the one hand, it captures trends from one of the robots, which are closer to the real trajectory. On the other, it tends to deviate from the real trajectory if the measurements from at least one of the robots also deviate. This happens because no delay is assumed to exist in transmitting data from the robots to the external computer, making it synchronized. It is also stated that it is not always the same robot that performs better alone. In some of the scenarios, the UGV yields more promising results, while in others, the UAV presents better performance metrics. Given

TABLE III
PERFORMANCE METRICS FOR DIFFERENT SETUPS AND TRAJECTORIES

r (m)	RMSE/r (%) [−]			Extreme Deviation [m]			Area Ratio (%) [−]		
	UGV	UAV	UGV-UAV	UGV	UAV	UGV-UAV	UGV	UAV	UGV-UAV
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**

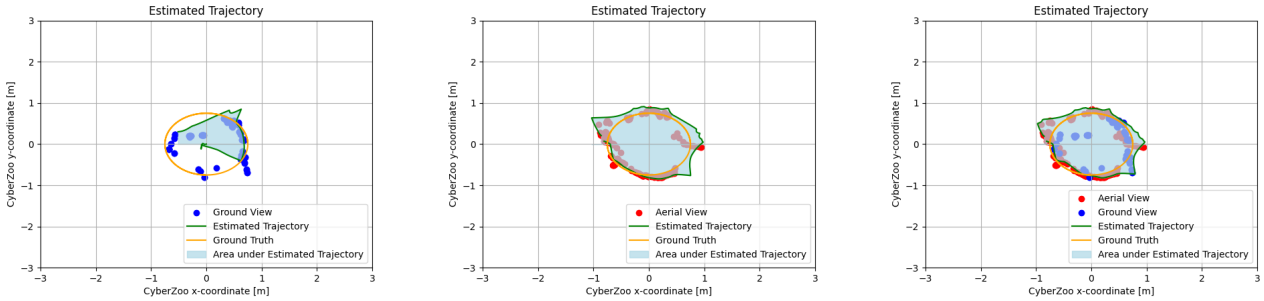


Fig. 17. Estimated object trajectories for r = 0.75 m. UGV alone (left), UAV alone (middle), and UGV-UAV collaboration (right).

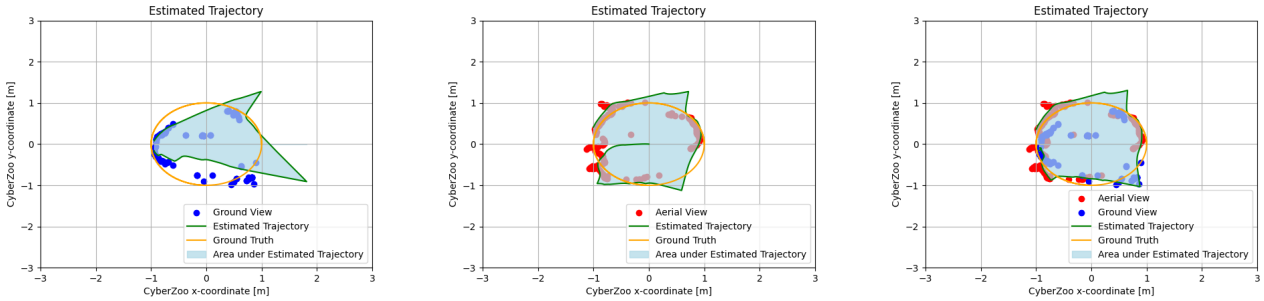


Fig. 18. Estimated object trajectories for r = 1.0 m. UGV alone (left), UAV alone (middle), and UGV-UAV collaboration (right).

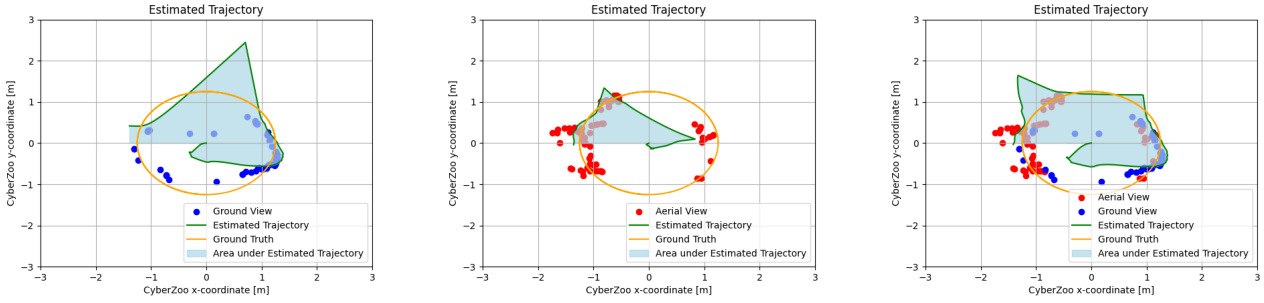


Fig. 19. Estimated object trajectories for r = 1.25 m. UGV alone (left), UAV alone (middle), and UGV-UAV collaboration (right).

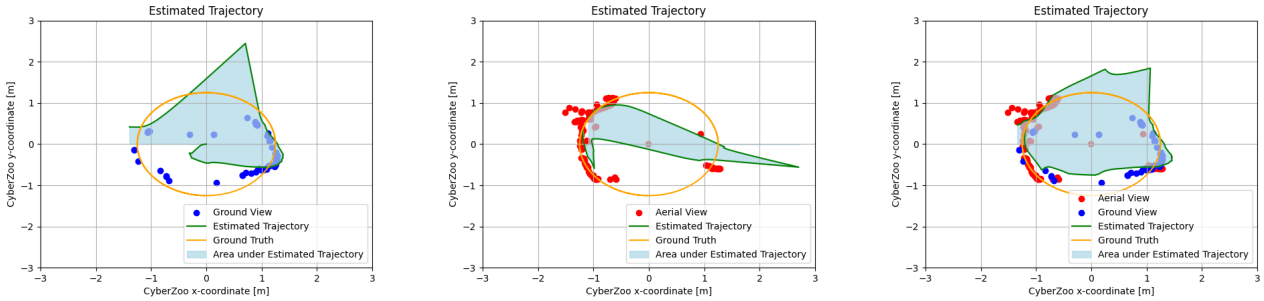


Fig. 20. Estimated object trajectories for r = 1.25 m with obstacles for the UAV. UGV alone (left), UAV alone (middle), and UGV-UAV collaboration (right).

that their standalone performances chiefly depended on their measurements, it is impractical to predict beforehand which of the two robots will perform better when the scenario is unknown, as this depends on the obstacles and the movements of the robots. For this reason, even though the UGV-UAV setup is not the best solution when it comes to RMSE in any of the scenarios specifically, it still represents the most advantageous compromise overall to handle an unknown scenario.

Regarding the extreme deviations from the true trajectory, the UAV produces the best results whenever its view is unobstructed. As a consequence, if small extreme deviations from the true trajectory are the most vital requirement in a particular mission and the scenario is known to be unobstructed, using a UAV standalone seems to be the best solution. A plausible reason for this phenomenon is that the camera from the UAV captures more fps. Due to the increased frame capture rate, it captures more points closer to each other, preventing the motion model in the KF from reaching higher velocities and deviating extremely from the true trajectory. Nevertheless, when the view of the UAV is obstructed, the amount of time with no measurements increases, allowing the estimated trajectory to deviate from the true trajectory significantly. That being the case, a UGV-UAV can significantly enhance performance, given that measurements from the UGV are extremely valuable when there are no measurements from the UAV for a long time.

The area ratio is the performance metric where a UGV-UAV collaboration stands out the most. As can be observed in the estimated trajectories, the UGV-UAV does an auspicious job of keeping the shape of the estimated trajectory close to a circle. When $r = 1.25$ m with an unobstructed view, the metric degrades since it is the only occasion where none of the robots detects the object in the first instants. While there is no information on the whereabouts of the object, it is assumed to be at the center of the Cyber Zoo $(x, y) = (0, 0)$. Thus, the shape of the estimated trajectory is significantly impacted. These findings are supported by the results presented in Table III, where the UGV-UAV collaboration scores the highest or very close to the highest, with a significant improvement to any of the two standalone robots.

Regarding the time to process the videos offline with the external computer, in 10 runs of a 10-second video for the ground view, the estimated trajectory takes between 19.41 and 22.30 seconds to obtain. Similarly, a 10-second video for the aerial view takes between 20.16 and 22.68 seconds to process. The collaborative setup takes between 36.68 and 42.19 seconds to generate the estimated trajectory. Thus, as expected, the computing power required to achieve a collaborative mission almost doubles. The code should be optimized to yield 4x acceleration if real-time processing is aimed.

In concluding remarks, metrics degrade with increasing r and in the presence of challenging environments. The collaboration between the robots appears to particularly enhance the area covered ratio. Nevertheless, it also plays a role in achieving a compromise in the RMSE obtained and improving the extreme deviation, especially in cluttered environments, provided that communication is smooth and with no substantial delays. The need to include both robots depends on the intricacies of the scenario, namely whether a larger or smaller area needs to be covered and on the presence of obstacles. It is likely that for unobstructed narrow-area trajectories, the deployment of one robot suffices. Particularly, the standalone UAV offers more encouraging results when compared to the standalone UGV. If a wide-area trajectory with obstacles needs to be tracked, the deployment of both robots is advisable. In cases where utmost precision is required, carefully planning the motion of the robots and even the possibility of deploying more robots need to be considered.

C. Elevation Map Generation

Subsequent to receiving the video feeds from the UGV, a frame where the signaling object was dropped and is in contact with the ground is selected. For this frame, the algorithm uses the procedure described in the methodology to determine the point where the object is touching the ground, the point where it would touch the ground if it were flat, and the elevation in pixels. Unless specified, the algorithm assumes by default that the UAV only carries and drops the object, and does not provide assistance in elevation determination. It is vital to note that throughout the analysis, the effect of partial occlusion in the signaling object was mitigated by taking the width of the object as explained in the methodology.

An example of the output of the algorithm is provided in Figure 21. In 10 runs of this experiment, the photograph takes between 0.54 and 0.62 seconds to be processed via the external computer. Given that real-time processing requires processing 15 fps for the UGV, the code requires around 9x acceleration to achieve it.

Fig. 21. When processing the video frame, the algorithm draws the point where the target object touches the ground (red), the point where it would touch the ground if the ground was flat (blue), and the line connecting them, which represents e (green). The photograph has reduced quality since the UGV captured it while moving.

In this frame, the ground distance from the UGV to the signaling object is 2 m, while the height of the box is 20 cm. The algorithm outputs a depth estimate of 2.091 m and an elevation estimate of 18.13 cm, representing relative errors of 4.55% and 9.35%, respectively. Even though this result is encouraging, it is crucial to test the performance of the algorithm when the UGV is positioned at different ground distances from the signaling object and in varying elevations. For this purpose, Figure 22 displays the distribution of measurements and relative errors for different depths and elevations when the UAV does not assist the UGV in determining the elevation.

From the top left plot in Figure 22, it is noticeable that the depth distribution from the UGV follows the perfect estimate closely. Apparently, for shorter distances, the algorithm slightly overestimates the depth, while for larger distances, the algorithm tends to underestimate it. Although the explanation for this occurrence is not trivial, it is suspected to be related to lens distortion, reduced resolution for larger distances, and the need for more precise calibration at larger distances. Even though the elevation is, to some degree, underestimated for the depth of 4 m, the data seems to be marginally above the perfect estimate across all distances. Consequently, elevation is expected to be generally underestimated, which is confirmed in the estimated elevation distribution. Looking at the relative errors, the curves appear to have inverse trends. While the depth relative errors tend to increase with decreasing depth, the elevation relative errors tend to increase with increasing the depth and, more specifically, with decreasing the elevation. Both of these trends are according to the expectations. For smaller depths, any deviation from reality has a higher impact on the depth relative error. Since the estimate for the elevation depends on the estimate for the depth, the errors for the elevation are a consequence of both error sources. Even though for larger depths, the depth relative errors typically decrease, the absolute errors slightly increase as observable in the depth distribution. This error propagates and has a higher impact on the elevation relative error when the elevation is small. As observed in the 3D graphs, the depth relative errors range from around 2% to 25%, whereas the elevation relative errors range from around 0% to 35%.

In order to assess the improvement in cases where the UAV provides a precise measurement of the ground distance to the signaling object to the UGV, Figure 23 portrays the elevation distribution and relative errors in the same scenarios, with UAV assistance. The depth distribution and relative errors are not shown since they are assumed to be perfect, with inexistent errors.

Judging by the elevation distribution, it is clear that, even though the median values practically did not change, the dispersion of elevations around the mean is significantly reduced. In this layout, the average relative errors are also notoriously mitigated. Except for one data point, the relative errors are lower than 13% in all instances. The data point corresponding to an elevation of 25 cm and to a depth of 4 m appears to be an outlier, where nearly no error was mitigated. Thus, for this data point, in particular, the error originates entirely from the elevation perception and is not related to a misleading depth estimate. An explanation for this happening relies on the fact that the homography matrix responsible for mapping the depth into real-world coordinates was calibrated using reference points up to distances of 3 m only. This finding highlights the importance of calibrating the camera and homography matrices using reference points that cover most of the areas of interest.

To recapitulate, there are two main sources of elevation error in the proposed setup, namely depth estimation and subsequent elevation estimation. Depth estimation errors originate mainly from assuming a simplified pinhole camera model and low image resolution, which leads to inaccuracies in object detection. In practical scenarios, occasional partial occlusion of the signaling object also plays a role. Ideally, the UAV supports the UGV in this task. As for elevation estimation errors, in addition to the same error sources, they also originate from inadequate homography calibration in particular regions. Furthermore, the simplified pinhole camera model plays a bigger role, since camera equations are applied twice, instead of just once. Without the assistance of the UAV, relative elevation errors distribute virtually homogeneously between 0% and 35%. With the assistance of the UAV, they sit below 13%, except for a particular data point where a relative error of 35% is reached. Since these results are generated in a well-lit environment, and the UGV is assumed to be aligned with the signaling object, these metrics are susceptible to degradation in more challenging scenarios. Moreover, depth values and elevation values encompassing a wider state-space region are also expected to further challenge these metrics. As with the discussion of other experiments, the applicability of the studied approach largely depends on the requirements of the mission. It is anticipated that the setup is not suitable for high-precision applications, such as for terrain mapping for construction. The pertinence of its employment in SaR missions depends on the characteristics of the terrain, which can be extremely hilly or relatively flat, and of the area to be covered.

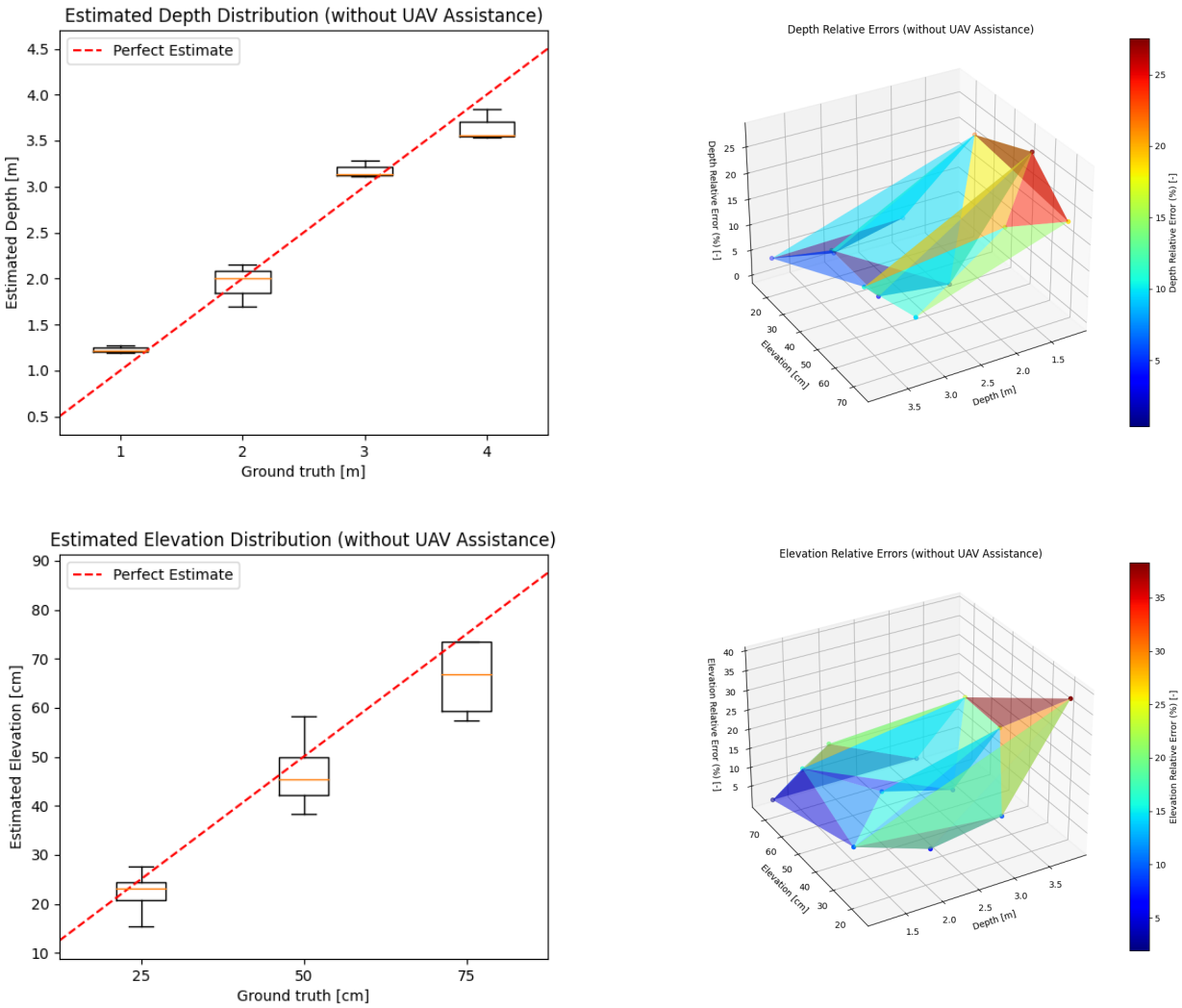


Fig. 22. Top: Depth distribution (left) and relative errors (right) without assistance from the UAV. Bottom: Elevation distribution (left) and relative errors (right) without assistance from the UAV.

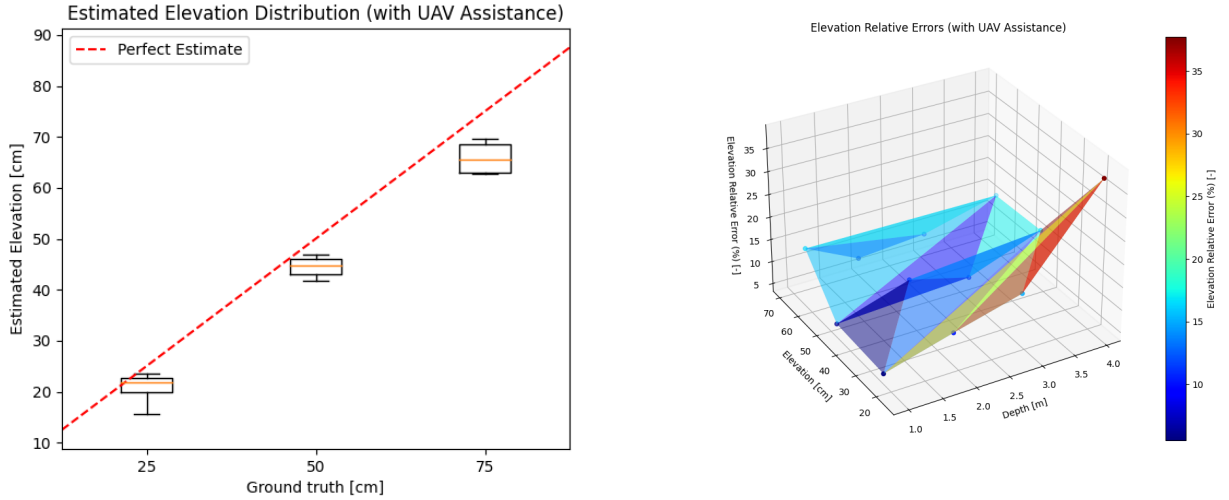


Fig. 23. Elevation distribution (left) and relative errors (right) assuming that the UAV provides a perfect estimate for depth.

VII. Limitations & Future Work

The current section goes through the limitations of the work, as well as suggestions on how it can be improved in the future. Even though some of the limitations have been mentioned throughout the paper, they are revisited to be discussed further.

A general limitation of the work is that it is not yet developed for real-time applications. To achieve this, it is essential to go through the code and make sure that no unnecessary computations are being performed. Furthermore, YOLO is known to have several times (often 10x or more) faster inference speeds with GPU acceleration. It is also always possible to use an external computer with more processing power, making real-time applications seem feasible. Onboard processing seems less likely, especially for elevation mapping. For the first set of experiments, an improvement of 3.75x is required using one-third of the processing power, resulting in an overall 11.25x boost. The object tracking task required a 4x time optimization using two-thirds of the processing power, resulting in an overall 6x acceleration. However, elevation mapping is expected to require 9x acceleration using one-third of the processing capability, representing an overall 27x boost and making it look far unrealistic. Nevertheless, it is recommended to double-check these assumptions, optimize the code and hardware, and attempt to achieve real-time onboard processing. The need to communicate to a central station for the computations to be performed is another central limitation. In an ideal setup, no external computer should be required. Furthermore, the study is limited in not considering the addition of further UGVs and UAVs to the robot team. This can enable increased accuracy in the depth estimation tasks through triangulation of the estimates of several robots, as well as the possibility of tracking more SaR victims simultaneously and having UAVs sequentially drop and pick up signaling objects to map the elevation in wider areas in a shorter amount of time.

Regarding the first set of experiments related to human depth estimation, a major limitation is a dependency on well-lit environments to achieve proper results. It is essential to make the system more robust to challenging environments, such as the presence of smoke or dark environments. For this purpose, similar approaches can be tested with thermal cameras instead of RGB cameras. Furthermore, in the set of poses studied, only one of the three measurements is taken to compare against the reference, and the one that yields the best results is used for each pose. Nevertheless, the algorithm should be able to autonomously identify which measurement to take as a reference and to merge information from different body measurements to make an even more informed estimate. It is also interesting to run the algorithm in non-conventional human poses, which are not so common in the COCO dataset, and to check whether the results are significantly degraded. The algorithm should also be tested for individuals who deviate the most from pattern measurements, such as for children and for individuals with a limb loss. Lastly, even though the program already includes a module to estimate the location of the missing key points based on the body symmetry and proportions, this has only been put into practice during the experiments when the face and the shoulders of some participants were not caught on the camera at short distances. It is, therefore, crucial to put it into a challenge by collecting and processing photographs of participants surrounded by different obstacles. Furthermore, real-time deployment was not yet achieved and is encouraged.

One of the major limitations of the second set of experiments is the lack of coordination and motion planning of the robots. They are assumed to be still since their motion planning is out of the scope of the project due to time constraints. Nevertheless, an effective coordination of the movements of the robots is expected to enhance the advantages of the heterogeneous robot team. As they navigate, robots can adeptly deviate from potential obstructions, ensuring a clearer line of sight and maneuverability within the environment. Furthermore, in instances where a specific area demands heightened attention or presents challenges, one robot can coordinate with another robot to cover that particular zone at any given time. Further investigation is also required into the efforts of getting the tracking algorithm to run in real-time and ideally onboard, since this is absolutely vital in tracking tasks. The impact of delays in transmitting the information, that can make the measurements and their receipt by the robots asynchronous, should also be considered, and the impacts should be assessed. Moreover, the trajectory estimation approach proposed in this research, should be tested for ground truth trajectories of other shapes, such as ellipses, straight lines, or random trajectories. Moreover, it is of interest to include wider outdoor trajectories and trajectories with varying speed magnitudes. By including a diverse set of typical SaR victim trajectories, it is also possible to predict whether another state estimator, such as a Particle Filter, an EKF, or a UKF, is more precise than a simple KF for this task. In addition, the volleyball posed several challenges due to its small size and non-distinctive pattern, causing the UGV to barely detect the ball in the presence of obstacles. Since the ultimate goal is to track the movements of humans, which are easier for YOLO to detect, real humans should be included in the experiments. When doing so, experiments should also be performed containing several human trajectories simultaneously. To handle multi-trajectory situations effectively, the algorithm needs to be extended to distinguish the data points that belong to each SaR victim, making it more complex but also more realistic.

As for the elevation mapping, the need to optimize the algorithm to run onboard and in real time persists. Since calibrating the homography matrix using more reference points is expected to make the algorithm more precise but also slower to run, it is crucial to investigate how many reference points can be added while keeping the running times low enough for real-time applications. Furthermore, the results hint that implementing a simplified pinhole camera model originates significant errors that are not fully mitigated even with the assistance of the UAV. Therefore, the prospect of including a more complex camera model that captures optical phenomena such as lens distortion and parallax is worth looking into. Moreover, the UAV should be able to autonomously identify regions of interest to determine the elevation, such as areas where there is a concentration of SaR victims or areas where the terrain looks particularly challenging. Subsequently, it

should autonomously drop the object. Lastly, in the current approach, the UAV carries only one object at a time and needs to pick it up and transport it somewhere else for the elevation at a different point to be estimated. Future work should explore possibilities to make this process scalable. An option includes carrying a larger number of small signaling objects.

VIII. Conclusions

The research objective included leveraging the complementary capabilities of flying and ground robots to deliver an affordable and effective SaR solution. For this purpose, a few typical SaR tasks were performed experimentally, namely estimating human depth from photographs using pose estimation, tracking the trajectory of an object by applying data fusion, and estimating terrain elevation. To ensure a simplistic setup and to keep the hardware costs low, these tasks were performed requiring visual depiction only, through photographs and videos.

The depth estimation experiment revealed the potential of the algorithm for accurately estimating human depth from photographs, yet unveiled its susceptibility to variations in poses. Even though most of the average relative errors originating from the algorithm are below 10%, they can also soar above 20% in specific circumstances. Such nuanced findings underscore the need for more robust algorithms capable of performing reliably beyond controlled environments and hint at the benefit of having access to an aerial view of the scene. The need to integrate the proposed system into a broader depth estimation module, which includes other forms of sensing, such as thermal imaging and acoustic sensing, remains a priority.

When tracking the trajectories of objects, the collaborative advantage of deploying both aerial and ground robots is evident in keeping the overall shape of the trajectory close to the true trajectory. Furthermore, this setup is more fault-safe in keeping a low RMSE and in preventing large extreme deviations, especially in cluttered environments. Overall, for unobstructed narrow-area trajectories, the deployment of one robot suffices. Particularly, the standalone UAV offers more encouraging results when compared to the standalone UGV. For wide-area trajectories with obstacles, the advantages of deploying both robots are more evident. In the widest trajectory simulated and in the presence of obstacles to the aerial view, the deployment of both robots showcased a reduction of 28% in the extreme deviation of the true trajectory, and an increase of 42% in the area covered ratio compared to the best-performing robot. The ability to navigate obstacles and maintain precision in diverse environments remains a challenge that demands further attention.

The proposed setup for elevation mapping faces two primary elevation error sources: depth estimation and subsequent elevation estimation. Depth estimation inaccuracies stem from simplified camera models and low image resolution, impacting object detection, compounded by occasional object occlusion. Elevation errors also result from poor homography calibration in specific regions, exacerbated by the double application of the simplified camera model. Without UAV assistance, elevation relative errors range from 0% to 35%, dropping to under 13% with the UAV help, except for one instance reaching 35%. These findings, observed in well-lit scenarios with aligned UGV-object settings, are expected to degrade in more challenging environments. The suitability of the approach varies: it does not meet high-precision needs but could find utility in SaR missions contingent on terrain characteristics and coverage area. Future work should delve into making this process adequate for real-time processing and scalable.

Collectively, these experiments represent notable progress in leveraging visual data and robotic capabilities for SaR. However, they also serve as a poignant reminder of the multifaceted nature of real-world scenarios, where dynamic and unpredictable conditions demand adaptable, versatile, and resilient systems. While running these experiments in real-time seems realistic, running them onboard is more daunting, especially for the elevation mapping task. Moving forward, bridging the gap between experimental findings and real-world deployment remains imperative. Addressing the identified limitations by refining algorithms, enhancing collaborative robot functionalities, and devising adaptable strategies tailored to diverse SaR scenarios will be pivotal. In essence, while these experiments mark significant strides in harnessing visual-based techniques within robotic systems for SaR missions, they contribute to an ongoing journey towards more effective, adaptable, and accessible solutions vital for saving lives.

References

[1] R. T. Newkirk, "The increasing cost of disasters in developed countries: A challenge to local planning and government," Journal of Contingencies and Crisis Management, vol. 9, no. 3, pp. 159–170, Dec. 2001.

[2] M. Zorn, Natural Disasters and Less Developed Countries, S. Pelc and M. Koderman, Eds. Springer International Publishing, Cham, Switzerland, Aug. 2018, pp. 59–78.

[3] R. R. Murphy, S. Tadokoro, and A. Kleiner, Disaster Robotics, B. Siciliano and O. Khatib, Eds. Springer International Publishing, Heidelberg, Germany, Jan. 2016, pp. 1577–1604.

[4] J. Qi, D. Song, H. Shang, N. Wang, C. Hua, C. Wu, X. Qi, and J. Han, "Search and rescue rotary-wing UAV and its application to the Lushan Ms 7.0 earthquake," Journal of Field Robotics, vol. 33, no. 3, pp. 290–321, Jul. 2016

[5] J. P. Zelten, "Digital photography and the dynamics of technology innovation," Ph.D. dissertation, Massachusetts Institute of Technology, Massachusetts, USA, Feb. 2002

[6] I. C. Condotta, T. M. Brown-Brandl, S. K. Pitla, J. P. Stinn, and K. O. Silva-Miranda, "Evaluation of low-cost depth cameras for agricultural applications," Computers and Electronics in Agriculture, vol. 173, pp. 105-394, Jun. 2020

[7] A. P. Hill, P. Prince, J. L. Snaddon, C. P. Doncaster, and A. Rogers, "Audiomoth: A low-cost acoustic device for monitoring biodiversity and the environment," HardwareX, vol. 6, no. e00073, Oct. 2019

[8] H. H. Titi, Feasibility Study for a Freeway Corridor Infrastructure Health Monitoring (HM) Instrumentation Testbed. Wisconsin DOT Research & Library Unit, Wisconsin, USA, Jul. 2012

[9] L. Zhaohua and G. Bochao, "Radar sensors in automatic driving cars," in 2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT), Nanchang, China: IEEE, Oct. 2020, pp. 239–242

[10] S. Hummel, A. Hudak, E. Uebler, M. Falkowski, and K. Megown, "A comparison of accuracy and cost of lidar versus stand exam data for landscape management on the Malheur National Forest," Journal of Forestry, vol. 109, pp. 267–273, Jul. 2011

[11] D. Van Nam and K. Gon-Woo, "Solid-state lidar based-slam: A concise review and application," in 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju Island, Korea (South): IEEE, Jan. 2021, pp. 302–305

[12] P. Fankhauser, M. Bloesch, P. Krüsi, R. Diethelm, M. Wermelinger, T. Schneider, M. Dymczyk, M. Hutter, and R. Siegwart, "Collaborative navigation for flying and walking robots," in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea: IEEE, Dec. 2016, pp. 2859–2866

[13] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning," IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 664–671, Jan. 2017

[14] C. D. Rodin, L. N. de Lima, F. A. de Alcantara Andrade, D. B. Haddad, T. A. Johansen, and R. Storvold, "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), Rio de Janeiro, Brazil: IEEE, Oct. 2018, pp. 1–8

[15] J. McGee, S. J. Mathew, and F. Gonzalez, "Unmanned aerial vehicle and artificial intelligence for thermal target detection in search and rescue applications," in 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece: IEEE, Sep. 2020, pp. 883–891

[16] D. Falconer, R. Ficklin, and K. Konolige, "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, San Francisco, CA, USA: IEEE, Apr. 2000, pp. 1868–1875

[17] J. Geisheimer, W. Marshall, and E. Greneker, "A continuous-wave (cw) radar for gait analysis," in Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, vol. 1, Pacific Grove, CA, USA: IEEE, Nov. 2001, pp. 834–838

[18] Q. Zhou, J. Liu, A. Host-Madsen, O. Boric-Lubecke, and V. Lubecke, "Detection of multiple heartbeats using doppler radar," in 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 2, Toulouse, France: IEEE, May 2006, pp. II–II

[19] M. B. Bejiga, A. Zeggada, and F. Melgani, "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), Beijing, China: IEEE, Jul. 2016, pp. 693–696

[20] V. A. Feraru, R. E. Andersen, and E. Boukas, "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), Abu Dhabi, United Arab Emirates: IEEE, Nov. 2020, pp. 57–64

[21] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Search and rescue operation using UAVs: A case study," Expert Systems with Applications, vol. 178, pp. 114-937, Sep. 2021

[22] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of YOLO algorithm developments," Procedia Computer Science, volume 199, pages 1066–1073, Oct. 2022

[23] N. D. Nath, C.-S. Cheng, and A. H. Behzadan, "Drone mapping of damage information in GPS-denied disaster sites," Advanced Engineering Informatics, vol. 51, pp. 101-450, Jan. 2022

[24] L. Xing, X. Fan, Y. Dong, Z. Xiong, L. Xing, Y. Yang, H. Bai, and C. Zhou, "Multi-UAV cooperative system for search and rescue based on YOLOv5," International Journal of Disaster Risk Reduction, vol. 76, pp. 102-972, Jun. 2022

[25] Y. I. Putra, A. H. Alasiry, A. Darmawan, H. Oktavianto, and Z. M. E. Darmawan, "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), Yogyakarta, Indonesia: IEEE, Dec. 2022, pp. 508–513

[26] M. Adel Musallam, R. Baptista, K. Al Ismaeil, and D. Aouada, "Temporal 3d human pose estimation for action recognition from arbitrary viewpoints," in 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA: IEEE, Dec. 2019, pp. 253–258

[27] S. Vasuhi, M. Vijayakumar, and V. Vaidehi, "Real time multiple human tracking using Kalman filter," in 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, India: IEEE, Mar. 2015, pp. 1–6

[28] P. Li, T. Zhang, and B. Ma, "Unscented Kalman filter for visual curve tracking," Image and Vision Computing, vol. 22, no. 2, pp. 157–164, Feb. 2004

[29] A. Howard and H. Seraji, "Vision-based terrain characterization and traversability assessment,"Journal of Robotic Systems, vol. 18, no. 10, pp. 577–587, Apr. 2001

[30] G. Jocher, A. Chaurasia, and J. Qiu, YOLO by Ultralytics, version 8.0.0, Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[31] C. D. Fryar, Q. Gu, C. L. Ogden, and K. M. Flegal, "Anthropometric reference data for children and adults; USA, 2011-2014," Vital and health statistics. Series 3, Data from the National Health and Nutrition Examination Survey, vol. 39, Aug. 2016

Part II

Preliminary Analysis

*This part has been assessed for the course AE4020 Literature Study.

3

Literature Review

3.1. Introduction

Natural disasters are a significant and growing threat to human health and safety, with an increasing frequency and cost of damage. In recent decades, the number and severity of natural disasters have risen dramatically, highlighting the need for effective disaster management [7]. Thus, optimizing search and rescue (SaR) operations has become a hot research topic in academia and industry due to their importance in mitigating further life losses and injuries. In the case of the least developed countries, the risk is particularly concerning, as nearly 90% of disaster-related deaths and 98% of people affected by disasters between 1991 and 2005 occurred in these countries [8]. Therefore, it is vital to make SaR operations not only more reliable but also more accessible.

In the past twenty years, SaR has become increasingly augmented with robotics [16]. Besides being able to traverse hazardous environments, robots have the capability of mapping environments in a fast and automated way. This competence makes them vital in disaster response effectiveness since survival rates drop steeply in the aftermath of a catastrophe. For example, the survival rate to a very strong earthquake (surface wave magnitude of 7.0) dropped from 91% in the first thirty minutes to 36.7% at the end of the second day [17]. Therefore, even if the robots do not perform the actual rescue themselves, providing a reliable map of the location of the victims speeds up the mission planning and increases its chances of success. For this thesis, the collaboration between an unmanned ground vehicle (UGV) and an unmanned aerial vehicle (UAV) is proposed to localize as many victims as possible in a SaR scenario.

3.1.1. Research scope

The scope of the research encompasses the design and hardware implementation of a UGV-UAV collaboration software, including processing the captured images from their cameras. The research focus for the thesis is developing a fast and low-cost solution to create a global map containing the coordinates of the detected victims.

The proposed hardware implementations and experiments will be conducted in the Cyber Zoo at the Delft University of Technology. The UGV used in this project is a differential drive robot equipped with a camera whose purpose is to detect victims from the ground. The UAV is a quadcopter equipped with a single camera, whose purpose is to provide a rough map of the scenario through aerial photographs.

3.1.2. Literature review objective

Both UGVs and UAVs have been applied to SaR missions in the past. UAVs provide affordable access to aerial data and are known for their ability to scan large areas in a small amount of time. However, their camera is subject to tilting motions that often result in reduced image quality. UGVs are typically slower at covering large areas due to rough terrain and a smaller angle of view. Nevertheless, they are better suited to carrying weights, accessing confined spaces, and taking high-quality photographs. Thus, combining the advantages of both types of robots is promising to improve the performance of the team. The research objective for the literature review given below is motivated by this logic:

The literature review delves into the development of an affordable autonomous UGV-UAV collaborative team to create a global map of the locations of victims in SaR scenarios.

In order to address the research objective, it is first necessary to define the different fields that potentially contain research gaps to be addressed.

The role of the drone is to fly to an area dictated by the ground robot, explore it and transmit raw aerial photographs of the scenario. Thus, the first sub-objective of the research is to learn how to process the photographs to extract the locations of obstacles and victims in an efficient way. Paparazzi open-source software is to be used for the motion planning of the drone.

The role of the ground robot is to transmit photographs from the ground and visit all the victims while avoiding obstacles. Hence, the images taken by the ground robot also need to be processed and merged with the photographs received from the drone to create a global map, which constitutes another sub-objective of this research. After generating the global map, the UGV adopts a target-oriented strategy to visit all victims as fast as possible while avoiding obstacles. For this purpose, a suitable mission and motion planning algorithm has to be chosen. This report reviews three promising solutions: Fuzzy Logic Control (FLC), Model Predictive Control (MPC), and search algorithms. Hence, the last sub-objective emerging from the UGV is familiarising these control strategies and path planning algorithms.

3.1.3. Literature review research questions

The objective and sub-objectives raise several research questions, forming the direction for the literature review.

1. How to process the consecutive aerial photographs and merge the photographs from the UGV and the UAV?
   (a) Which object classification algorithms are the most suitable for accuracy and ease of implementation?
   (b) How to make sure that an object captured from two different points of view is, in fact, the same object?
   (c) How to scale the dimensions of the objects in an autonomous way?
   (d) How to assess if the information transmitted by the UAV is relevant?
2. How to make the two robots communicate effectively?
3. What state of the art algorithm should be used to plan the path and motion of the UGV?

3.1.4. Literature review structure

The literature review finds research supporting a thesis that can answer the research questions and fulfill the research objectives. In Section 3.1, the relevance of the project is discussed. Furthermore, the project is clearly stated by defining its scope and objectives. Then, in Section 3.2, an overview of different image processing algorithms is given. This includes object classification algorithms and techniques to fuse the information in the images taken from different points of view. One of the most crucial aspects to ensure the effectiveness of the system is the functional communication between the UGV and the UAV. Hence, Section 3.3 is dedicated to describing communication protocols for the sequential and concurrent deployment of the two robots. Subsequently, Section 3.4 inspects the state of the art FLC-based controllers, MPC-based controllers, and search algorithms for path planning. The literature review is concluded in Section 3.5 by providing a summary of the literature review and by identifying possible improvements in future work based on research gaps found in the literature. An overview of the structure of the literature review is given in Table 3.1.

Table 3.1: Literature review structure. Own elaboration.

Chapter	Content	Research Questions
3.1 Introduction	Problem Definition, Research Questions	-
3.2 Image Processing	Object Classification and Image Fusion	1a & 1b & 1c & 1d
3.3 Communication Protocols	Sequential Deployment, Concurrent Deployment, Design Considerations	2
3.4 Control Strategies and Path Planning	Fuzzy Logic Control, Model Predictive Control and Search Algorithms	3
3.5 Conclusion	Literature Review Summary	-

3.2. Image Processing

One of the main goals of the research is the post-processing of photographs collected by the UGV and the UAV. Thus, the current chapter provides an extensive literature review on object classification and image fusion. Firstly, the photographs must be analyzed to extract relevant information, such as detecting and categorizing objects as obstacles or victims, to answer **Question 1(a)**. To achieve this, Section 3.2.1 focuses on assessing the state of the art object detection algorithms. Once the objects have been detected, the images taken from two different angles of view must be fused. By performing image fusion, it is expected that errors and uncertainty are decreased. Hence, Section 3.2.2 evaluates topics such as object matching and scaling to address **Questions 1(b) and 1(c)**, respectively. It is then essential to determine when the images taken by the UAV are not enhancing the operation of the UGV and discard them when that is the case to fulfill the purpose of **Question 1(d)**. The chapter concludes with a discussion of the essential findings and an assessment of the impact of the review on the project in Section 3.2.3.

3.2.1. Object classification

Object classification is a crucial task in computer vision, which involves identifying and categorizing objects in images or videos. Accurately detecting and classifying objects is crucial in various applications, ranging from surveillance to autonomous driving and robotics. Significant advancements in computer vision have been made in recent years, particularly deep learning-based methods. These methods have demonstrated remarkable performance in object classification tasks, often outperforming traditional computer vision-based methods. Nevertheless, the latter still have their own strengths and can be effective in specific scenarios. A comparison between the two workflows is depicted in Figure 3.1.

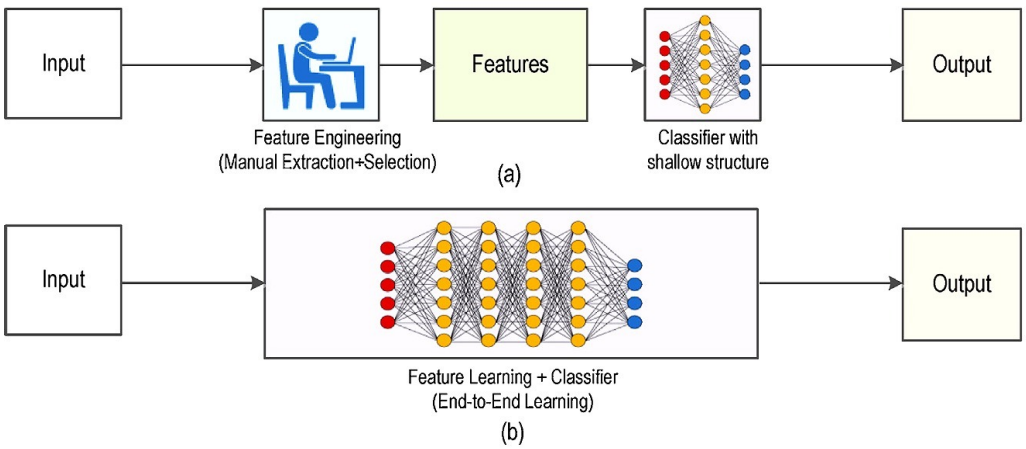


Figure 3.1: a) Traditional computer vision workflow, and b) Deep learning workflow. Retrieved from [18].

In this section, both traditional computer vision-based and deep learning-based methods for object classification are explored in Section 3.2.1 and Section 3.2.1. Then, Section 3.2.1 contains considerations to help the reader choose the most suitable workflow for a specific scenario.

Traditional computer vision
Traditional computer vision-based methods involve extracting relevant features from input images, which are then used to train classifiers or detectors, as represented in Figure 3.1. In this subsection, several feature extraction algorithms and classifiers are examined: color-based detection, feature-based detection, template matching, Bag of Visual Words (BoVW), and Support Vector Machines (SVMs). No distinction is made between feature extractors and classifiers because most algorithms can have both purposes, and numerous examples of combinations exist in the literature.

Firstly, color-based detection segments regions of an image based on their color, making it useful for detecting objects with distinct color patterns. Color-based detection is helpful in robotics for tasks such as identifying and sorting different colored objects in a warehouse. For instance, a robot with a color camera and color-based detection algorithm can sort items based on their colors, such as separating apples and oranges. As an example of this approach, Thendral et al. [19] perform a case study to quickly and accurately identify oranges based on their color without needing physical contact or additional sensors.

Another common approach is to use feature-based detection to identify key features in an image, such as edges or corners, and use these features to classify objects. This method is helpful in detecting objects with distinct visual features, such as the octagonal shape of a stop sign. One of the primary applications of the method is autonomous vehicles, where lane markings, traffic signs, and other vehicles on the road have distinct shapes, as investigated by Chen et al. [20]. In search and rescue missions, feature-based detection can detect humans in a disaster zone. To illustrate this, Gautham et al. [21] propose a framework using Scale Invariant Feature Transformation (SIFT) for high-density detailed feature extractions since disaster scenarios are characterized by their complexity. In essence, the SIFT algorithm transforms the image into a collection of local feature vectors that are distinctive and invariable to scaling, rotations, and translations of the image [22]. An alternative to this algorithm is to use Speeded Up Robust Features (SURF), a concept introduced by Bay et al. [23] in 2006. SURF is generally faster and more robust, using a more specific descriptor and a more robust method of calculating the scale and orientation. In 2006, Rosten and Drummond [24] introduced Features from Accelerated Segment Test (FAST), a method to extract more detailed information about key points. Unlike SIFT and SURF, FAST does not produce a descriptor by itself, but it can be combined with the former to achieve better results.

When a template of the object is available, and the object is not expected to be occluded, template matching is often a viable solution. It compares an input image to a predefined template image, identifying regions of the input image that match the template. Hence, it is also particularly useful for detecting objects with distinct shapes. An example is robotic surgery, where it can be used to detect and track surgical instruments in real time during a procedure. This can be done by creating a template of the surgical instrument and then using template matching to track its location and orientation in the surgical field, such as in the work of Reiter et al. [25].

The BoVW model is also a popular technique for object recognition in computer vision. It is based on the idea that an image can be represented as a histogram of visual words, similar to words in a text document. In the BoVW approach, an image is first divided into small regions, and features are extracted from each region. These features might be extracted using SIFT, SURF, or other descriptors. Then, the features are clustered into a set of visual words, which is fundamentally a set of visual features, using a clustering algorithm. A clustering algorithm is, in essence, a machine learning algorithm that is used to group similar objects based on their features. An outcome is a dictionary of visual words representing the range of features in the image. Once the visual words have been defined, an image can be represented as a histogram of the frequency of each visual word in the image. Afterward, the histogram is used as a classifier. This technique is used by Xu et al. [26] to classify a terrain using aerial images.

SVMs are the most common classifiers found in the literature. The formulation was first introduced by Cortes and Vapnik [27] in 1995. The idea behind SVM is to find the best decision boundary that separates the data points into different classes. One of the advantages of SVM is that it has a solid theoretical foundation based on statistical learning theory, which makes it possible to control the trade-off between model complexity and generalization performance. Furthermore, SVM is suitable for small to medium-sized

datasets and relatively insensitive to irrelevant features. Nevertheless, it showcases limitations, including its sensitivity to the choice of hyperparameters, its tendency to overfit on noisy datasets, and its difficulty in handling large-scale datasets. Examples of successful SVM implementations include the work of Kavitha and Suruliandi on melanoma detection [28] in the field of medicine and the case study performed by Wang et al. [29] on apple harvesting robots. In autonomous driving, applications include autonomous lane change, as investigated by Liu et al. [30].

Deep learning

Deep learning is a sub-field of machine learning that has revolutionized computer vision in recent years. Deep learning approaches use neural networks with many layers to learn features directly from raw data, such as images or videos, without needing handcrafted features (see Figure 3.1).

Neural networks are a class of machine learning algorithms inspired by the structure and function of the human brain. Its most basic formulation is the perceptron, a single-layer neural network capable of classifying linearly separable patterns. Over the years, researchers developed and refined neural network architectures, introducing multi-layer neural networks, which allowed for non-linear pattern recognition tasks. However, the lack of computing power and training data limited the practical application of neural networks. With the substantial developments in computer science and backpropagation, an algorithm for training multi-layer neural networks, neural networks gained broader acceptance in machine learning. Backpropagation allows neural networks to learn from large amounts of labeled data and has been used to achieve a state of the art performance on various tasks, including image recognition, speech recognition, and natural language processing.

Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of biological neurons in the human brain. ANNs are composed of multiple layers of artificial neurons, each receiving input from the neurons in the previous layer and producing an output passed on to the next layer of neurons. The produced output is usually a non-linear function of a linear combination of the inputs. The function applied by each neuron is called the activation function and is one of the design hyperparameters. In an ANN, the input layer receives input data, which is passed through one or more hidden layers, each consisting of multiple neurons. The final layer is the output layer, which produces the final output of the network. The connections between neurons in an ANN are characterized by a set of weights, which are adjusted during training to minimize the error between the predicted and actual output. Figure 3.2 displays a typical architecture of an ANN.

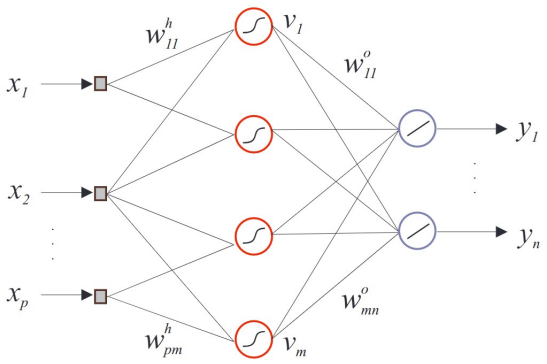


Figure 3.2: A simple ANN architecture composed of a sigmoid hidden layer and a linear output layer. Weights are represented as w, whereas biases are represented as v. Retrieved from [31].

A Deep Neural Network (DNN) is a type of ANN containing multiple hidden layers. By containing more hidden layers, they can learn more complex and abstract representations of the input data. While increasing the number of hidden layers can mean improved accuracy, it also leads to higher computational intensity and a tendency to overfit.

Convolutional Neural Networks (CNNs) are a class of DNNs specifically designed for image-processing tasks. They use a series of convolutional layers to extract features from images, with each layer learning a set of kernels that are applied to the input image to produce a set of feature maps. These feature maps are

then passed through non-linear activation functions to introduce non-linearity into the model. The output of the convolutional layers is then passed through one or more fully connected layers, which produce the final classification output. A typical CNN architecture and the graphical representation of the procedure are shown in Figure 3.3 and Figure 3.4, respectively.

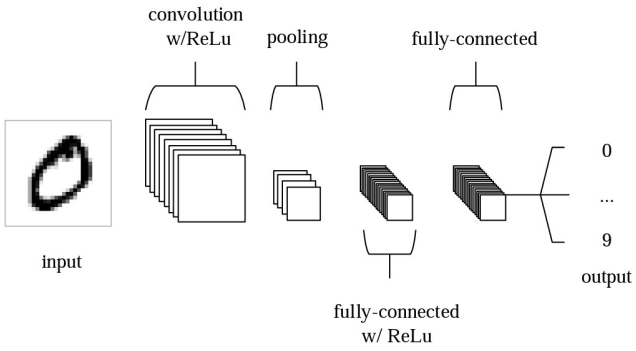


Figure 3.3: A simple CNN architecture composed of five layers. Retrieved from [32].

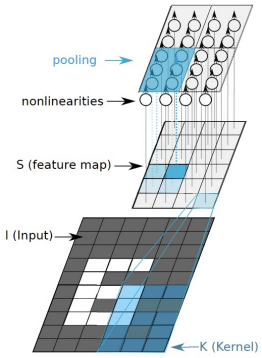


Figure 3.4: Graphical representation of a CNN. Retrieved from [31].

The main advantage of CNNs over ANNs for image processing tasks is their ability to exploit the spatial structure of images. ANNs treat each pixel in an image as a separate input, which does not consider the spatial relationships between pixels. In contrast, CNNs use convolutional layers to learn spatial relationships between pixels in an image through kernels, which are matrices convolved with the input image. Therefore, convolutional layers in CNNs can learn filters specific to different patterns. By learning them, CNNs can recognize more complex features in images. Additionally, CNNs use pooling layers to reduce the size of the feature maps and make them more robust to translations of the input image. Hence, CNNs can learn translation-invariant representations, which means they can recognize an object regardless of its position in the image. In contrast to ANNs, which can have millions of parameters, CNNs typically have fewer parameters due to parameter sharing. The same set of filters is applied to different input image regions, and they make use of pooling layers, which reduce the size of the feature maps.

One of the main advantages of deep learning in computer vision can also be one of its main disadvantages. On the one hand, it has the ability to automatically extract relevant features from raw data, making it possible to train models on large-scale datasets with millions of images; on the other hand, it is generally computationally expensive, making it impractical for some applications. Essentially, deep learning models may require large amounts of labeled data, which can be time-consuming. Furthermore, deep learning models may be susceptible to overfitting, which occurs when the model becomes too complex and starts to fit the training data too closely, resulting in poor performance on new data.

Dhillon and Verma [33] performed a global review of models, methodologies, and applications of CNNs to object detection. Some of the mentioned applications include the framework suggested by Villa et al. to monitor wild animals [34] and a technique proposed by Shan et al. for automatic recognition of facial expressions [35]. Fundamentally, CNNs have been applied to numerous image processing applications and form the current state of the art in object detection.

Selection considerations

Several considerations should be taken into account when selecting between traditional computer vision algorithms and deep learning-based approaches for object classification. It is vital to highlight that there are no complex rules regarding the best selection, as it depends on the problem. O'Mahony et al. [36] have compared traditional computer vision and deep learning by naming the most important advantages of each technique. In their publication, they emphasize that, even though deep learning consistently performs better than traditional algorithms, there are still situations where using profound learning results in an excessive workload.

There is no question that deep learning revolutionized the field of computer vision, as it introduced the concept of end-to-end learning. The underlying patterns are discovered automatically by feeding it a

dataset of labeled images, which have been annotated with what classes of objects are present in each image. Consequently, the traditional problem of choosing which features are essential in each given image becomes irrelevant. Hence, the workflow of a computer vision engineer changed dramatically, where the knowledge and expertise in extracting hand-crafted features have been replaced by knowledge and expertise in iterating through deep learning architectures [36].

Despite the substantial change in paradigm, using deep learning can be too extreme in some situations, as traditional computer vision techniques can solve the problem much more efficiently and in fewer lines of code. Consider classifying two product classes on an assembly line conveyor belt, one with red paint and one with blue paint. A deep neural network works provided that enough data can be collected to train it. However, the same can be achieved by using simple color thresholding. One must practice common sense when choosing which route to take for a given computer vision application [36].

An overview of the strengths and weaknesses of various image processing algorithms can be found in Table 3.2. Siamese networks, Faster Region-based Convolutional Neural Network (R-CNN), and You Only Look Once (YOLO) algorithms are treated in Section 3.2.2 due to their extensive applicability in object-matching tasks.

Table 3.2: Image processing algorithms comparison. Own elaboration.

Algorithm	Advantages	Disadvantages
Color-based detection	Simple to implement, computationally efficient	Sensitive to lighting conditions and color variations, may require manual tuning of color thresholds
Feature-based detection (SIFT, SURF)	Robust to variations in lighting, scale, and rotation, can handle complex scenes	Can be computationally intensive, may require manual selection and design of features, may not work well with textureless or repetitive scenes
Template matching	Simple to implement, computationally efficient	Sensitive to variations in scale, rotation, and lighting, requires an accurate template
Bag of Visual Words (BoVW)	Robust to variations in lighting, scale, and rotation, can handle complex scenes, can capture spatial relationships between features	Requires a large amount of training data, can be computationally intensive, may require manual selection and design of features
Siamese networks	Can learn to compare images directly without relying on feature extraction, can handle variations in lighting, scale, and rotation, can work well with small datasets	Requires a large amount of training data and computational resources, may not generalize well to unseen data, can be sensitive to image quality
Convolutional Neural Networks (CNN)	Can learn features automatically from raw pixels, robust to variations in lighting, scale, and rotation, can handle complex scenes	Requires a large amount of training data and computational resources, can be prone to overfitting, may not be interpretable
Faster R-CNN	Can detect objects accurately and efficiently, can handle complex scenes, can capture spatial relationships between objects	Requires a large amount of training data and computational resources, may not be as accurate as YOLO for large objects
YOLO (You Only Look Once)	Can detect objects accurately and efficiently, can handle complex scenes, can handle larger objects better than Faster R-CNN	May not be as accurate as Faster R-CNN for small objects or scenes with occlusions

3.2.2. Image fusion

One of the primary motivations for using a multi-robot setup in this research is that obtaining information from the air and the ground can help locate and assist more victims in the aftermath of a disaster. Thus, image fusion is critical in combining information from different viewpoints. To exemplify this challenge, Figure 3.5 depicts the view for a UGV and a UAV when collaborating on creating an environment map.

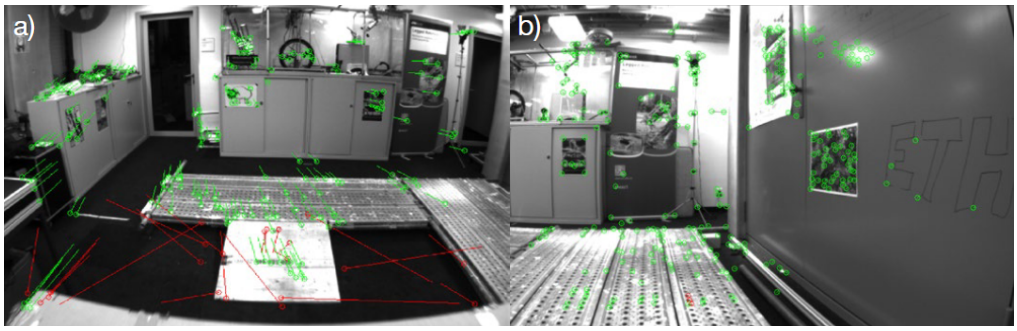


Figure 3.5: Example of change in viewpoints for a) flying robot and b) ground robot. Retrieved from [4].

Before using the pictures to plan a joint mission, it is essential to ensure that the objects captured from different viewpoints are indeed the same and that their dimensions are accurately estimated. Moreover, it is equally important to assess the relevance of the information transmitted by the UAV. If the UAV is temporarily unavailable or transmitting noisy images, it is vital that the UGV learns to recognize it and not rely on this information. Various techniques for object matching, object scaling, and relevance assessment are thus explored in Section 3.2.2, Section 3.2.2, and Section 3.2.2, respectively.

Object matching

One of the critical challenges in multi-robot systems is to ensure that the robots can recognize the same objects despite differences in viewpoints. The current subsection explores various approaches for matching objects captured from different viewpoints.

One approach to matching objects captured from different viewpoints is to use the traditional computer vision techniques assessed in Section 3.2.1. Features such as corners and edges are matched across the images to establish correspondence. These correspondences can then compute a transformation between the two images, which can be used to align the images and match the objects. Traditional methods are suitable for applications where the objects are well-defined and have distinct features that can be reliably detected. In Figure 3.5, the green dots represent features being tracked across the two photographs.

Novel deep learning-based object detection algorithms, including Faster R-CNN and YOLO, are powerful tools for detecting objects in images. As reviewed by Du [37], these novel algorithms can learn rich representations of objects and often outperform CNNs in real-time object detection tasks. Therefore, a promising object-matching solution is to combine Siamese networks with YOLO, as suggested by Melekhov et al. [38]. A Siamese network is a type of neural network architecture that learns to compare two inputs and determine how similar they are. Thus, detected objects can be fed into a Siamese network to produce meaningful matches, even when the objects have complex shapes and appearance variations.

Once the objects have been matched, verifying that the results are satisfactory is crucial. By using visual cues, such as object appearance, it is possible to establish object identity and check the associations. For instance, if two images contain a person and a car, the person in the first picture can never be considered a car in the second picture. In this straightforward example, visual cues should be used to reject the match and search for a better match. Further verification can be done by considering geometric constraints. Marques et al. [39] propose using the spatial relationship between objects in the scene to match them. Reconsidering the previous example, if two images contain a person and a car, and the person is located next to the car in both images, it is likely that the person and the car are the same physical objects.

Matching objects captured by multiple robots is a critical task in multi-robot systems. In this subsection, diverse object matching and verification approaches have been discussed, including traditional computer vision techniques and deep learning-based object detection algorithms. Traditional computer vision is

typically advantageous in situations where a limited number of well-defined objects is to be matched. As an example, if the goal is to match a hexagon-shaped stop sign seen from different perspectives, traditional computer vision is an obvious choice due to the well-defined shape of the object and the need for less training data. However, if the environment is unpredictable and it is not possible to select characteristic features in objects (such as edges or corners), deep learning-based object detection has shown to be able to learn rich representations of objects from raw data and is, therefore, the most suitable option. Furthermore, verification algorithms have also been dissected, namely visual cues and geometric constraints. By exploring these approaches and verifying them, robots can work collaboratively to identify and locate objects of interest, even in complex and dynamic environments.

Object scaling

When processing the images, it is essential to ensure that the objects match and that the images are correctly scaled. The size of an object can be used to estimate its distance from the camera, which is a critical parameter for terrain mapping. Nonetheless, determining the scale of an object is challenging, especially when the cameras have different focal lengths. In this chapter, methods to scale objects in images without the need for human intervention are reviewed.

The most straightforward way of scaling objects in images is to use known object dimensions as a reference. If the dimensions of a specific object are known, this information can be used directly to estimate the dimensions of other objects in the image. This method called object-based scaling, is often used in crowd surveillance, where the dimensions of people or vehicles are known or can be estimated from other sources.

It is also possible to estimate the sizes of objects using monocular depth estimation and stereo vision. Monocular depth estimation estimates the depth of a scene using a single camera. In contrast, stereo vision involves using two cameras placed a certain distance apart to capture two slightly different images of the same scene. Monocular depth estimation relies on perspective, shading, texture, and motion cues. On the other hand, stereo vision analyses the disparities between the images to estimate the depth of objects in the scene and their size. Stereo vision is generally more accurate, but it requires an extra camera. Lagisetty et al. [40] have successfully used stereo vision for robotic object detection and obstacle avoidance.

A more recent approach is to use deep learning techniques to learn the scale of objects in images. This can be achieved by training a neural network to estimate the size of objects based on their visual appearance. For this purpose, a large dataset of labeled images where the ground truth size of objects is known needs to be collected. Once the network is trained, it can automatically scale objects in new images without needing prior knowledge of their size. A clear advantage of this approach is that object classification, matching, and scaling can be achieved simultaneously as three different outputs of the same network. Kuhad et al. [41] implement this technique to the application of estimating the number of calories contained in a food sample by estimating the size of the portion and using visual features such as colors.

Regardless of the approach used, it is crucial to consider potential differences between different cameras, such as varying focal lengths, the field of view, and perspective distortion. These factors can have a significant impact on the accuracy of the scaling, and careful calibration of the cameras is necessary to achieve accurate results. Moreover, the accuracy of the scaling is often limited by the resolution and quality of the images, as well as by the presence of occlusions.

Relevance assessment

When working with multiple robots, it is essential to ensure that the information transmitted by each robot is relevant to the mission. Nonetheless, the relevance of information is not solely determined by the type of information itself but also by the quality of the data that the robots have captured. For instance, if the image quality of the photographs taken by a robot is poor, it may not be possible to extract useful information from them, even if the object detection algorithm is state of the art. For this reason, it is vital to perform a quality assessment of the data. This includes evaluating the image quality and focusing on its resolution, sharpness, and contrast. The most common technique found in the literature for evaluating image quality is to use metrics such as peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM), using the formulations included in the work of Setiadi [42]. Ding et al. [43] propose using neural networks, SVMs, PSNR, and SSIM to assess image and video quality. Using this combined framework makes it possible to obtain mapping functions between objective quality assessment and

subjective quality assessment indexes. This functionality is particularly valuable if assessment criteria depend on the circumstances. For example, suppose a robot is in a challenging environment, such as a poorly illuminated room. In that case, the images may not be high quality, but they may still contain valuable information. In contrast, if the robot captures images in a clutter-free environment, the images may be of high quality, but the information they provide may not add to the mission.

Suppose one robot is the center of operations, and an auxiliary robot is used to get a different perspective for the mission. In that case, it is vital that the auxiliary robot transmits relevant information. Assessing the relevance of information depends on the specifications of the mission. Taking as an example obstacle avoidance tasks, if the goal is that a UAV double-check the location and dimensions of obstacles detected by the UGV, it only provides relevant photographs if it is referring to the same obstacles. Thus, in this case, it makes sense to discard all images provided by the UAV where object-matching algorithms do not find correspondence with the photos taken by the UGV.

To sum up, assessing the relevance of information transmitted by the robots involves evaluating the quality of the images captured, the context in which they were captured, and if they are inside the scope. Only by considering these factors can it be ensured that the information is helpful for the mission.

3.2.3. Discussion

Several object classification and image fusion algorithms have been dissected throughout the current chapter. It has been found that even though some algorithms are more sophisticated than others, there are no hard rules regarding the perfect algorithm selection. The designer can use a specific method for different purposes and even combine methods to increase performance.

There are two broad categories of object classification algorithms, namely traditional computer vision and deep learning. Traditional methods, such as SIFT and SURF, have been successfully applied to object detection tasks for many years. However, for SaR applications, it can be too demanding to define a priori the characteristic features of each object. SaR environments are characterized by their complexity, unpredictability, and object occlusions. Hence, deep learning-based approaches, particularly those based on convolutional neural networks (CNNs), are a promising alternative, especially if complex and diverse data sets can be collected, which is expected to be the case at the CyberZoo at the Delft University of Technology. Nevertheless, traditional computer vision algorithms would again be suitable candidates if a simplified representation of obstacles and victims is to be considered, such as red circles and blue circles on the ground.

As for image fusion, the importance of accurate object matching, scaling, and performing relevance assessments has been addressed. It has been found that Siamese networks are novelty and have shown encouraging results in matching objects across different viewpoints. When scaling objects, state of the art consists of monocular depth estimation, stereo vision, and deep learning. In a laboratory setting, it is possible to include an object with a known size as a reference in case object scaling is not the focus of the work. Finally, relevance assessment relies on image quality, context, and scope. The state of the art is to combine neural networks, SVMs, and metrics such as PSNR or SSIM to evaluate not only the quality of the image but also the value it brings to the mission.

Considering the discussed topics, image processing is highly relevant to various engineering applications. Guaranteeing robust image processing is even more crucial in applications without human intervention, which is standard practice in robotics. Therefore, a clear research gap is a need to optimize sophisticated deep-learning object detection and matching algorithms to require less data and make computer vision more accessible, even for non-experts.

3.3. Communication Protocols

In a multi-robot system, the effective communication of robots is critical to their operation. This chapter investigates the communication requirements of a UGV-UAV collaboration and how communication can be designed to be functional and efficient to answer **Question 2**.

In Section 3.3.1, a comprehensive review of communication protocols in a sequential deployment setting is provided. Due to its relative simplicity, the sequential deployment setting is the starting point. Subsequently, Section 3.3.2 explores the communication protocols for concurrent deployment, which is a more realistic scenario and closer to the research objectives. In this setting, the robots must communicate

in real time and bidirectionally to ensure effective coordination. Thus, the significant challenges of moving from sequential to concurrent deployment are mentioned, and examples of how concurrent deployment can be used in real-life applications are given. Then, Section 3.3.3 compromises a study on the various design options available, including parameter tuning, such as the frequency at which information is exchanged. Finally, Section 3.3.4 discusses how the literature review on communication protocols can shape the research. This includes an analysis of the essential findings and how they can contribute to designing and implementing the communication protocols between the two robots.

3.3.1. Sequential deployment

A sequential deployment of robots means that they work in a predetermined order, and each robot performs its task independently. Consequently, the second robot being deployed has to wait for the first to complete its task, leading to reduced speed performance. Even though sequential deployment presents fewer real-life applications than concurrent deployment, there are still many examples where sequential deployment suffices and provides satisfactory results.

Fankhauser et al. [4] use a UAV to provide the first mapping of an indoor environment with obstacles and a target location in their first experimental setup. The UGV is only then deployed and updates the global map created by the UAV with its own observations. In the case of this experiment, the data is transmitted via Wi-Fi to a processor incorporated into the UGV with high computational power and performs all merging computations. In a later stage, Fankhauser et al. declare that their work can be adapted to support concurrent deployment. This statement is further discussed in Section 3.3.2. Delmerico et al. [44] also propose an air-ground collaboration architecture to solve a ground path planning problem. Unlike the work of Fankhauser et al. [4], this operation is conducted outdoors. In the first stage, a human operator operates a UAV to follow determined waypoints. Then, the UAV learns to explore the area of interest and creates a map of the terrain in its local processor. Similarly, the UGV is then deployed and updates the map based on its own perception. However, instead of receiving the rough mapping provided by the UAV locally and operating on its own processor, all information is sent to an external computer responsible for merging the mappings and returning key information. This setup, composed of a UGV, a UAV, and an external computer, is often used in real life because it alleviates the workload on the robots.

De Petris et al. [45] examine a sequential UGV-UAV deployment architecture to map unknown environments. The UGV is the primary exploring agent in their research and carries the UAV while moving. As long as the environment can be explored from the ground, the UAV remains in its platform, and all exploration is conducted by the UGV. However, if particular areas are inaccessible from the ground, the UAV is deployed and explores them. Meanwhile, the UGV waits for the UAV to land back on the platform before continuing its operation. An illustration of the experiment can be found in Figure 3.6, which depicts an example of sequential deployment.

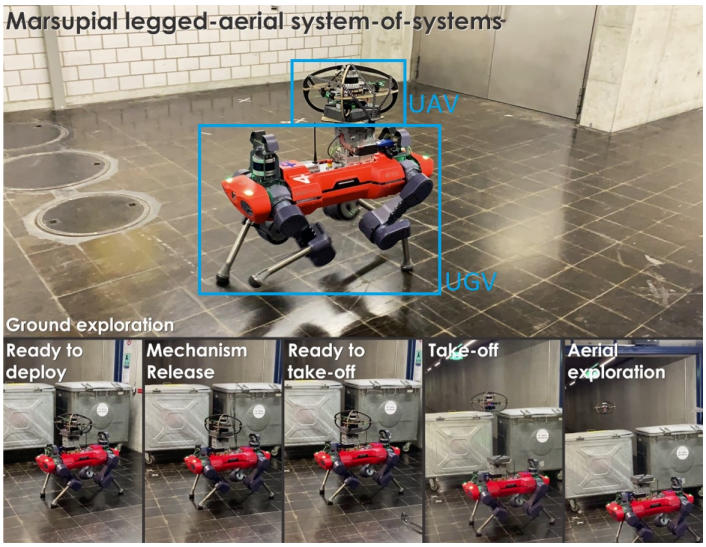


Figure 3.6: Sequential UGV-UAV deployment for unknown environment exploration. Retrieved from [45].

In the case of this experiment, both the UGV and the UAV keep a mapping of the explored terrain in their local processors. Selected information is shared only when requested. Even though the information is transmitted wirelessly in this setup, docking stations offer the opportunity to communicate via USB sticks. Wang et al. [46] designed a Physical Couple Interface (PCI) to allow wired communication in UGV-UAV applications. Wired communication has the advantage of speeding up information exchange. Nevertheless, it has the physical limitation of the proximity of the robots. Furthermore, a more precise UAV landing control is required to ensure it is correctly plugged in after landing.

Some of the most common communication protocols in the literature are Wi-Fi and Bluetooth. Wi-Fi is a natural choice in most cases due to its easy availability, especially in indoor settings. However, when power saving is a significant concern, the low-power wireless technology of Bluetooth can transmit data over short distances. Another option would be to use ZigBee, which is not only a low-power but also a low-data-rate communication protocol. However, it is more often used in environments that are not subject to considerable changes, such as industrial automation. Cellular communication via a cellular modem and a SIM card can also be considered if data needs to be transmitted over large distances. Lastly, radio frequency presents itself as a flexible approach. Equipping the robots with radio frequency transceivers and adjusting the wavelength of the signal makes it possible to make a compromise between transmission distance and frequency.

In summary, the communication infrastructure for sequential deployment can be relatively simple, as the communication is unidirectional and time-dependent. It may consist of a wireless or a wired network, depending on the environment where the robots operate and the needs of the mission. Generally, the bandwidth requirements for communication in sequential deployment are low, and the design can be straightforward.

3.3.2. Concurrent deployment

Concurrent deployment of robots means that they perform tasks simultaneously and do not necessarily need to wait for inputs from other robots. Even though they are still working together to achieve a common goal, they are more independent. Compared to sequential deployment, concurrent deployment requires more complex and dynamic communication protocols. In this setting, the robots need to communicate with each other in real-time and bidirectionally.

Fankhauser et al. [4] have extended their previous work on sequential deployment to allow for concurrent operations. To do so, information is transmitted per batch at a defined frequency. By transmitting information per batch, the UGV can update a specific batch and communicate its findings while the UAV is exploring another batch and communicating its own updates. This allows them to share incomplete maps and incentivizes the UAV to explore new areas. The UGV then utilizes map updates from the UAV to increase the range and precision of path planning. In this experiment, all algorithms are running real-time: localization and mapping, legged state estimation, and navigation planning.

Miki et al. [47] investigate a framework where a UAV physically assists a UGV in climbing a cliff by attaching a tether to a structure on top of the cliff. The UGV then climbs by winding the fixed tether. The setup used in their experiments is depicted in Figure 3.7 and Figure 3.8.

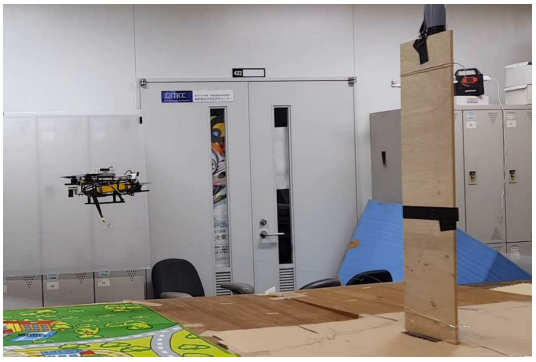


Figure 3.7: The UAV attaches a tether to a structure by flying around it. Retrieved from [47].

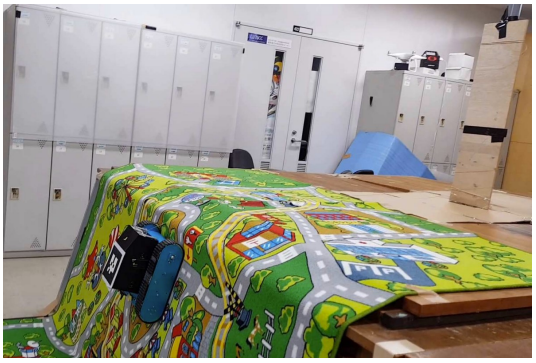


Figure 3.8: The UGV climbs a cliff by winding the tether attached by the UAV. Retrieved from [47].

This novel collaborative system makes the two robots navigate as a team through an unknown environment and enhances UGV traversability by reaching previously inaccessible areas. During the course of the mission, a global map is maintained by the two robots that communicate via Wi-Fi. To achieve the required computing power, both robots are custom-made. The computations on the drone run on a Jetson TX2 module, which is developed specifically for applications where real-time processing is required and bandwidth can be an issue. As for the ground robot, a UP core processing unit is used.

Another application where the physical coordination between a ground robot and a flying robot is beneficial is object manipulation. In their work, Lissandrini et al. [48] propose an architecture of two heterogeneous robots to enable heavy payload transportation and agile manipulation. In the experimental setup, the two robots grasp a plastic bar allowing elastic deformation. A plastic box is placed as an obstacle so that the robots need to perform an avoidance maneuver to take the plastic bar to the destination. In their work, Lissandrini et al. run the experiment simultaneously in the Gazebo simulator and experimental environments. To ensure a common interface to both the simulation and laboratory, a Robotic Operating System (ROS) network is used. All computations are done off-board on a local computer. Thus, the UGV and the UAV communicate with the local computer via Wi-Fi.

Zhou et al. [49] focus on a swarm of UAVs to explore an indoor environment as fast as possible. They propose a fully autonomous system with little communication by ensuring UAVs explore different regions. The UAVs perform computations on board and store their own findings. However, even in such a scenario, communication protocols are set for the case when two robots accidentally explore the same region. For this purpose, communications are maintained through a local ad hoc network.

The battery of the drone is a limiting operation factor in many real-life applications. Flying robots typically cannot hover for more than 10-15 minutes. Ren et al. [50] suggest an air-ground collaboration to overcome this issue. The ground robot acts as a mobile charging station and follows the flying robot from the ground. As soon as the battery drops below a defined threshold, the flying robot lands on the ground robot and recharges before being deployed again. No communication protocols are referred to in their work, as it is entirely done in simulation. Thus, it is still a research gap to be explored in real life in the future.

Depending on the requirements of each application and the amount of data that needs to be transmitted, several communication options are available for the concurrent deployment of robots. ROS remains one of the most popular platforms, as it allows robots to communicate using a variety of protocols, making it a flexible option. However, robotic operating systems can be targets of cyber attacks, which raise security concerns. In their study, Mukhandi et al. [51] show that it is possible to integrate Message Queuing Telemetry Transport (MQTT) and ROS for securing a ROS-enabled robotic system. MQTT is a lightweight, publish-subscribe messaging protocol designed for Internet of Things (IoT) devices. Following the logic of IoT, Extensible Messaging and Presence Protocol (XMPP) is also an open-source protocol that allows real time communication between devices over the internet. It is highly scalable and robust, making it suitable for robotic applications. Bendel et al. [52] propose a communication infrastructure using XMPP and demonstrate the capabilities of their service platform in one case study of remote robot control. The third IoT communication protocol in literature is Wireless Sensor and Actuator Network (WSAN). WSAN is a wireless communication protocol that allows robots to communicate with each other using sensors and actuators. It is suitable for dynamic environments where robots adapt to changing conditions. Curiac [53] investigated the conceptual framework, challenges, and perspectives of WSANs, concluding that it is a good backbone for complex distribution and mobile control applications.

Essentially, concurrent deployment of robots requires more complex and dynamic communication protocols than sequential deployment. Real-time, bidirectional, dynamic, robust, low latency, and scalable communication protocols are crucial for efficiently coordinating tasks. Different experimental setups are found in the literature; computations can be done on-board or off-board, and robots can be customized, as in the case of the work of Miki et al. [47], where the processor for the drone is chosen specifically to enable low bandwidth communication. Having a powerful external computer performing the computations allows for the integration of real-life hardware and simulations and more sophisticated service protocols. IoT led to several communication infrastructures, such as ROS, MQTT, XMPP, and WSAN. It has been found that combining different options can lead to an increasingly safe system.

3.3.3. Design considerations

Designing communication protocols in robotics requires careful consideration of several important factors, namely the environment in which the robots operate, the available bandwidth, latency, robustness, scalability, security, and flexibility.

Firstly, the specific environment in which the robots operate is critical when designing a communication protocol. For example, if the robots operate in a noisy environment or with low visibility, the communication infrastructure needs to be designed to handle noise and interference or degraded signal quality. Furthermore, the available bandwidth is a crucial design variable. The communication protocol should be designed to efficiently use the available bandwidth, especially in scenarios where multiple robots communicate simultaneously. Bandwidth limitations can cause communication delays or failures, which can impact the overall performance of the robotic system. In the same line of thought, the latency must be minimized to ensure timely decision-making and coordination of tasks. In case there is network congestion or lost or corrupted messages, it is vital to ensure that communication is robust. Hence, a system that maintains reliable communication, even in the face of disruptions, is less likely to fail. In cases where swarms of robots are studied, scalability is a crucial issue. As the number of robots in the system increases, the protocol must handle the increased communication load without negatively impacting performance. Security is one of the most critical considerations in the design of communication protocols for robotics. Especially in outdoor environments, preventing unauthorized access or interference with the robotic system is vital. Lastly, the protocol should be flexible and capable of handling task and scenario changes.

The frequency of information exchange between robots highly depends on the task. For example, if the robots perform a task requiring high precision, such as mapping an area while avoiding obstacles, the frequency of information exchange must be high to ensure timely and accurate decision-making. In contrast, if the robots are performing a task that requires less precision, such as monitoring a particular area, the frequency of information exchange can be lower. If the robots exchange information too frequently, it may result in a communication overload and harm the performance of the system. Nonetheless, if the robots exchange information too infrequently, it may lead to delayed decision-making and coordination of tasks.

To sum up, designing communication protocols for robots requires careful consideration of several vital factors. The protocol must be designed to operate effectively in the specific mission. Even though the design may not always be straightforward, it is essential in multi-robot settings to come up with solid solutions from the beginning so that the mission can run smoothly and the objectives can be fulfilled.

3.3.4. Discussion

This chapter explores communication protocols for both sequential and concurrent deployment scenarios in robotic systems. Sequential deployment offers a more straightforward approach; it might be worth considering first implementing sequential deployment before moving to concurrent deployments, such as in the study conducted by Fankhauser et al. [4]. Even if the final goal is that both robots are kept active simultaneously, ensuring that they can perform their tasks sequentially helps in troubleshooting. For the purpose of this research and most real-life applications, achieving a well-performing concurrent system is naturally the final goal.

Given that the CyberZoo at the Delft University of Technology is equipped with a Wireless Local Area Network (WLAN), transmitting information via Wi-Fi is straightforward. The literature study concluded that Wi-Fi is one of the most common means of communicating in robotic applications in indoor environments due to its availability and reliability. ROS is worth considering, as it allows a standard interface to interact with both robots. Since the experiments are to be performed in a controlled academic setting, integrating MQTT, XMPP, or WSAN is not a priority, as safety is not a concern. Knowing that the processing power of the robots is not substantial, it makes sense to have an external computer to perform complex calculations while interacting with both robots. Hence, it is expected that communication will happen bidirectionally between the robots and the computer.

After the experiments are moved to a concurrent deployment scenario, determining the frequency of information exchange is a critical design consideration. Depending on the bandwidth of the WLAN and the processing power of the hardware, it is expected that the frequency of information exchange is tuned by trial and error. Regarding design considerations, latency and flexibility are prioritized over security or scalability.

Overall, communication protocols are fundamental to the effective operation of robotic systems. The literature review presented in this chapter provides valuable insights into designing a suitable communication infrastructure for this research. An identified research gap is the development of highly scalable communication frameworks, as recent applications typically involve multiple robots. An example is the deployment of swarms of drones in concerts and shows, as described by Ohta [54]. In the context of deployment in the vicinity of people, ensuring increased cyber security of communication protocols is another research gap to be addressed.

3.4. Control Strategies and Path Planning

Before deciding which control and path planning strategy to adopt for the UGV, it is crucial to review and compare the most favorable algorithms in similar applications, to address **Question 3**. Even though several options have been used in robotics in the past, the most promising solutions, given the characteristics of this project, include FLC, MPC, and search algorithms. Thus, it is essential to understand their mathematical formulations, assets, drawbacks, and examples of past implementations. For this purpose, inspections on FLC, MPC, and search algorithms are provided in Section 3.4.1, Section 3.4.2, and Section 3.4.3, respectively. Subsequently, Section 3.4.4 discusses the reviewed content and concludes the chapter.

3.4.1. Fuzzy Logic Control

FLC is a mathematical framework that deals with reasoning that is approximate rather than fixed and exact. The concept was first introduced by Zadeh in 1965 [55]. It is used in control systems to produce decisions based on incomplete or imprecise information. Fuzzy logic is an alternative to Boolean logic, where binary values of true and false are replaced by degrees of truth represented by fuzzy sets. This allows for a more efficient and natural representation of uncertain and complex systems, leading to better control decisions. This concept is thoroughly described in Section 3.4.1, where the fundamentals of FLC are given. The general mathematical background is then followed by a review of two FLC types, namely Mamdani and Takagi-Sugeno Control, in Section 3.4.1 and Section 3.4.1, respectively. A review of the advantages and disadvantages of FLC found in the literature is presented in Section 3.4.1. Lastly, examples of FLC in robotics are identified in Section 3.4.1.

Fundamentals of FLC

The basic idea behind FLC is to map input variables to output variables using rules derived from expert knowledge or data. For this purpose, three steps must be taken: fuzzification, inference, and defuzzification. The architecture of a typical FLC-based controller can be found in Figure 3.9.

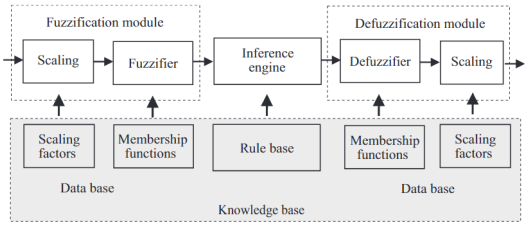


Figure 3.9: Architecture of an FLC-based controller. Retrieved from [31].

During the fuzzification step, input variables are converted into fuzzy sets, which represent linguistic terms such as ”low”, ”medium”, or ”high”. A fuzzy set A on a domain X is defined by the membership function $\mu_A(x)$, which is a mapping from the universe X into the unit interval $\mu_A(x) : X \rightarrow [0, 1]$. The description of the value of $\mu_A(x)$ is expressed in the equation below.

$$\mu_A(x) = \begin{cases} 0 & \text{x is with full certainty not a member of A} \\ \in (0,1) & \text{x is partially a member of A} \\ 1 & \text{x is with full certainty member of A} \end{cases} \tag{3.1}$$

According to this representation, a variable x belonging to Universe X can become a partial member of A. This is the characteristic that distinguishes fuzzy sets from traditional crisp sets and eases the integration of linguistic information. Hence, vague linguist terms such as ”far” or ”near” are precisely quantified using analytical expressions. The shapes of the membership functions are chosen intuitively and define the conventions for classifying the variables. Common shapes for membership functions can be found in Figure 3.10.

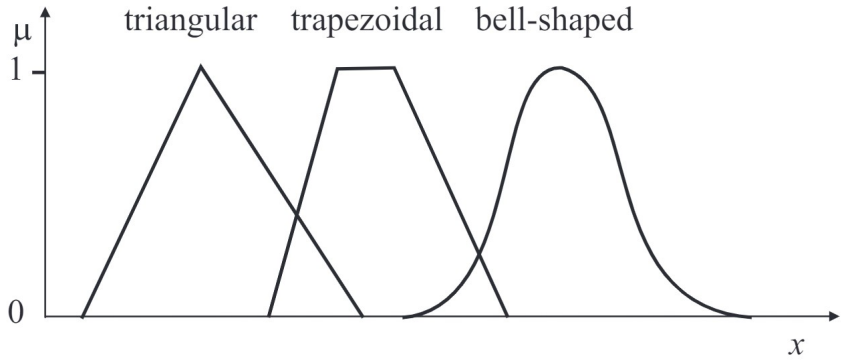


Figure 3.10: Examples of membership function shapes. Retrieved from [31].

A set of rules is used throughout the inference process to make decisions based on the input variables. The rules are usually expressed in the form of ”if-then” statements, where the antecedent (if-part) defines the conditions that must be met, and the consequent (then-part) defines the action to be taken. The rules are evaluated in order, and the outputs from each rule are combined using fuzzy operators to produce the overall output. Since rules are typically represented as R, the basic inference process is illustrated in the following equation:

$$R_i : \textbf{If } x \; is \; A_i \textbf{ then } y \; is \; B_i \qquad i = 1, 2, ..., K \tag{3.2}$$

In (3.2), x represents the input or antecedent linguistic variable, A_i is the antecedent linguistic variable represented by a fuzzy set, y is the output or consequent linguistic variable, whereas B_i represents the consequent linguistic variable in terms of a fuzzy set. The usage of fuzzy connectives such as ”and” and ”or” is further illustrated in the following expressions:

$$R_i : \textbf{If } x_1 \; is \; A_{i1} \textbf{ and } x_2 \; is \; A_{i2} \; ... \textbf{ and } x_p \; is \; A_{ip} \textbf{ then } y \; is \; B_i \qquad i = 1, 2, ..., K \tag{3.3}$$

$$R_i : \textbf{If } x_1 \; is \; A_{i1} \textbf{ or } x_2 \; is \; A_{i2} \; ... \textbf{ or } x_p \; is \; A_{ip} \textbf{ then } y \; is \; B_i \qquad i = 1, 2, ..., K \tag{3.4}$$

The connectives ”and” and ”or” are usually implemented using t-norms and s-norms, respectively. Due to their simplicity, the most common approach is to use the minimum function for the connective ”and” and the maximum function for the connective ”or”. Of course, there is a margin to combine different connectives and select other t-norms and s-norms.

The output computational process for an FLC involves using the mathematical concept of implication [56]. Various implication algorithms are available, but Mamdani (or minimum) implication is most commonly used in literature. The membership degree of each rule R, represented as μ_R, is calculated for Mamdani implication using the following equation:

$$\mu_R(x, y) = I(\mu_A(x), \mu_B(y)) \tag{3.5}$$

The Mamdani implication is computed as follows:

$$I(\mu_A(x), \mu_B(y)) = \min(\mu_A(x), \mu_B(y)) \tag{3.6}$$

Thus, Mamdani implication can be referred to as minimum implication. The previous Equations (3.5) and (3.6) form the basis for max-min inference, also known as Mamdani inference. The following Section 3.4.1 covers Mamdani control in more detail. Another relevant fuzzy controller is the Takagi-Sugeno controller, covered in Section 3.4.1. The defuzzification step is covered in the respective subsections.

Mamdani Control

The architecture of a Proportional-Integral-Derivative (PID) fuzzy controller is depicted in Figure 3.11.

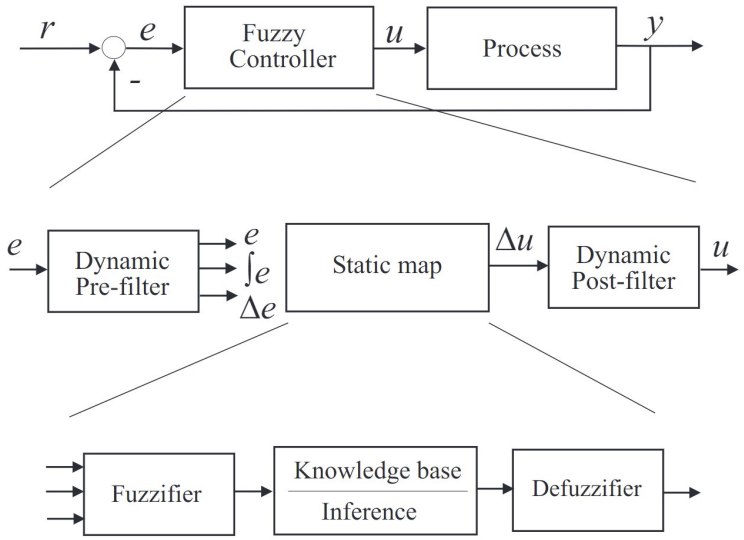


Figure 3.11: Architecture of a PID fuzzy controller. Retrieved from [31].

As it can be seen from Figure 3.11, the fuzzy controller is composed of a dynamic pre-filter, a static map (the general architecture described in Section 3.4.1), and a dynamic post-filter. The pre-filter determines the integral and the derivative of the error signal. Naturally, these two additional inputs must also be fuzzified to be used in the inference. A common approach is to represent them as a singleton fuzzy set whose value is one of the crisp values.

After converting the inputs to fuzzy sets A' and performing inference based on rules to obtain the output fuzzy set B', it is necessary to convert the output back to a crisp value that can be used as a control command. This process is called defuzzification and constitutes the last step of the procedure. There are several methods for defuzzification, being Center of Gravity (COG) and Mean of Maxima (MOM) are the two most used (see Figure 3.12). Once again, there is a margin to pick any method deemed appropriate.

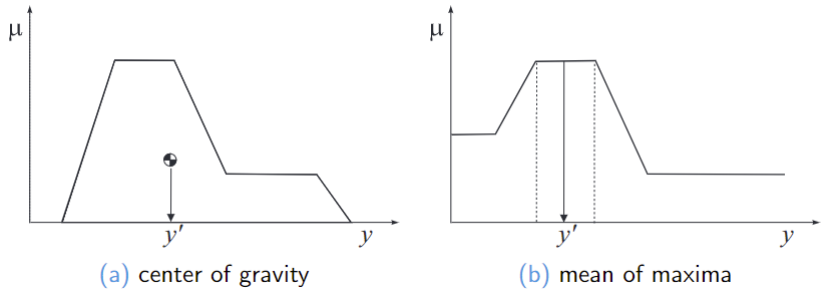


Figure 3.12: a) COG and b) MOM defuzzification methods. Retrieved from [31].

Mamdani systems are known for their simplicity and ease of interpretation, making them popular for many real-world applications.

Takagi-Sugeno Control
Takagi-Sugeno (TS) is a type of fuzzy logic control system that was proposed by T. Takagi and M. Sugeno in 1985 [57]. It is a hybrid system that combines the strengths of both fuzzy logic and mathematical modeling.

The main difference between Mamdani control and Takagi-Sugeno control is how the consequents are defined. While in Mamdani control, the consequents are linguistic terms to describe the output y (see (3.2)), Takagi-Sugeno specifies the outputs as an analytical expression based on the inputs, $f(x)$. Thus, rules are adapted and defined as in the equation below.

$$R_i : \textbf{If } x \; is \; A_i \textbf{ then } y_i = f_i(x) \qquad i = 1, 2, ..., K \tag{3.7}$$

In (3.7), $f_i(x)$ is typically either a constant or a linear combination of the inputs.

The output of a controller with K rules is computed by taking the average of all outputs y weighted by each rule's membership degrees, as expressed in the following equation:

$$y = \frac{\sum_{i=1}^{K} \mu_{R_i} y_i}{\sum_{i=1}^{K} \mu_{R_i}} \tag{3.8}$$

Takagi-Sugeno systems are also known for their simplicity, as they use linear mathematical models that allow for simple and efficient implementation. They are also particularly suitable for predictive modeling applications since they clearly understand the relationship between inputs and outputs. The fact that they combine fuzzy logic and mathematical modeling contributes to their flexibility.

Advantages and disadvantages of FLC
FLC allows experts to express their knowledge about the system using linguistic terms and rules, which can be more intuitive and easier to understand than mathematical models. FLC is also well suited for dealing with uncertainty and imprecision, as it allows for the integration of expert knowledge and data flexibly and adaptively. It is relatively simple to implement and can be applied to many systems, making it a popular choice for many control applications. Not only is it robust to changes in system dynamics, as the rules can be updated to reflect changes in the system, but it also can handle a wide range of input and output variables. It can easily accommodate changes in the system or the goals of the control. Moreover, it is less sensitive to measurement noise than other control methods, as the fuzzification process can smooth out the input signals. Lastly, it provides interactive control, as the expert can adjust the rules and control parameters to achieve the desired performance.

Even though FLC-based controllers present a wide range of advantages, these also come with drawbacks. Naturally, it requires expert knowledge of the system to create the rules and determine the control parameters, which can be time-consuming and challenging. Creating rules that accurately represent the behavior of the system can be complex and challenging, especially for large and complex systems. It is also important to highlight that FLC lacks the mathematical rigor of other control methods, making it challenging to analyze the stability and robustness of the control system. It can be computationally intensive, especially for large and complex systems, as the rules need to be evaluated in real-time. As with other intelligent control methods, it is prone to over-fitting, where the rules fit too closely to the training data and do not generalize well to new data. Another one of the major drawbacks is that it is typically not well suited for systems with strong non-linearities, as the linear combination of rules may not accurately capture the non-linear behavior of the system. Lastly, it is susceptible to local optima, where the control system converges to a sub-optimal solution rather than the global optimum.

An overview of the advantages and disadvantages of FLC, based on the work of Albertos et al. [58], is given in Table 3.3.

Table 3.3: Overview of the advantages and disadvantages of FLC. Own elaboration.

Advantages of FLC	Disadvantages of FLC
• Natural expression of knowledge	• Expert knowledge required
• Handling of uncertainty	• Complexity in rule creation
• Simplicity	• Lack of mathematical rigor
• Robustness	• Computational cost
• Flexibility	• Prone to over-fitting
• Insensitivity to measurement noise	• Not suited to strong nonlinearities
• Interactivity	• Susceptible to local optima

Examples of FLC in robotics
FLC-based controllers have numerous applications in robotics, including navigation and control tasks, object manipulation, adaptive control, and human-robot interaction, just to name a few.

FLC can be used to control the motion of robots, enabling them to navigate uncertain or changing environments. Seraji et al. [59] propose using a novel measure of terrain traversability to infer a fuzzy rule-based traversability index in real-time. This index is then used as one of the weighting factors in guiding the robot. Besides being used for navigation, FLC can be used to make decisions about obstacle avoidance, as demonstrated by Pandey et al. [60] in a simulation environment.

In cases where robotic manipulation is of interest, FLC can control the dexterity of robotic arms, allowing them to perform tasks such as grasping and manipulating objects in an unstructured environment. Ciobanu and Popescu [61] created a framework with an FLC-based controller to make a robot capable of complex in-hand manipulation tasks.

In adaptive control, fuzzy logic can adjust robot behavior based on environmental changes or the performance of the robot. This allows the robot to improve its performance and adapt to new conditions continually. In the case of Das and Kar [62], a fuzzy logic system is used to estimate parameter variations and unknown nonlinearities in the kinematics of a wheeled robot, and improvements have been noted both in simulation and experimental tests.

FLC can also be used to interpret human input and control the behavior of robots in human-robot interaction applications, such as in rehabilitation or assistive technology. Ascensão [56] uses FLC to control a drone that performs therapeutic sessions for children diagnosed with autism.

Considering the examples mentioned above, it is concluded that FLC provides a versatile and flexible approach for controlling robots in uncertain and complex environments, making it an essential tool for planning the mission and motion of robots.

3.4.2. Model Predictive Control

Model Predictive Control (MPC) is a family of control methods that use mathematical models to predict the future behavior of a system and optimize control inputs to achieve a desired outcome. It was first proposed in an early form by Clarke et al. in 1987 [63]. The current chapter compromises both the fundamentals of MPC (explained in Section 3.4.2) and the advantages and disadvantages of using MPC (given in Section 3.4.2). Subsequently, Section 3.4.2 entails a literature review on the usage of MPC in robotics.

Fundamentals of MPC
Due to the predictive character of the MPC, its performance highly depends on how accurate the available models of the system are. The mathematical models used in MPC can be either linear or nonlinear, depending on the complexity of the system being controlled. The MPC algorithm consists of three main steps: prediction, control optimization, and action [64]. The model of the system predicts its future behavior based on the current state and control inputs in the prediction step. In the control optimization step, an optimization problem is solved to determine the control inputs that minimize a cost function while satisfying constraints on inputs, outputs, and the system. The optimization process is typically performed in real-time and repeated at a fast rate, with the computed control, and the model is updated based on new observations.

Then, the first action is implemented, the horizon is moved forward for one control time step, and the optimization is performed again. These steps are illustrated in Figure 3.13.

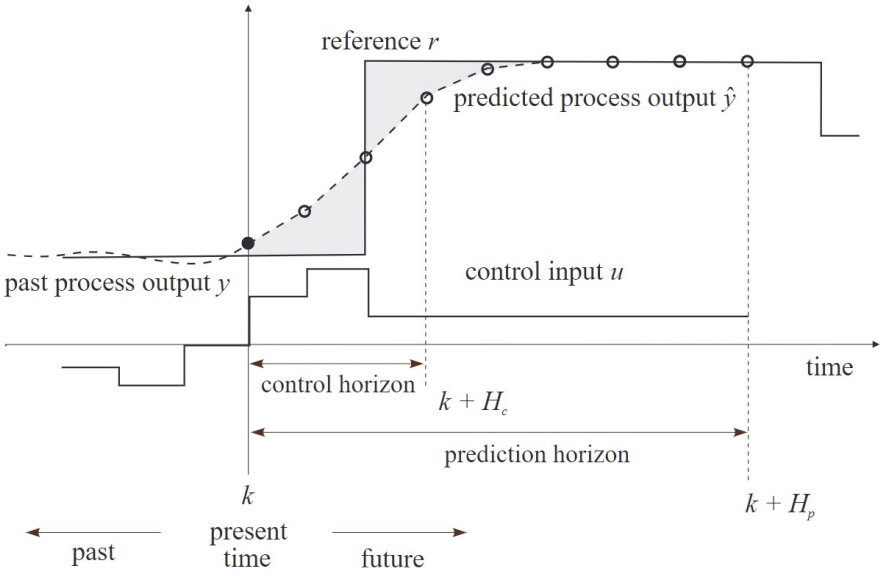


Figure 3.13: Fundamental MPC scheme. Retrieved from [31].

In a deterministic, time-invariant system, the dynamics are modeled using the relationship expressed in the system of equations below.

$$\begin{cases} \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{y}_k = g(\boldsymbol{x}_k, \boldsymbol{u}_k) \end{cases} \tag{3.9}$$

In (3.9), x_k represents the current state vector, u_k designates the current control input vector, and y_k represents the current system output. x_{k+1} is the value of the state vector in the following time step.

Figure 3.13 illustrates the fundamental principles of MPC, which involve predicting the process output $\hat{y}$ over a prediction horizon H_p at the current time step k. Next, the optimal control sequence is computed over a control horizon H_c, and the first action is executed. In order to determine the optimal control sequence, a cost function is minimized. The general form of the cost function is expressed in the following equation:

$$J_k(x_k, \boldsymbol{u}_k) = h(x_k, \boldsymbol{u}_k) \tag{3.10}$$

In (3.10), J is a scalar indicator describing how good the control inputs will be in the future. Even though the cost function can be theoretically arbitrary, it typically represents a trade-off between variables. A possible formulation is to have the first term represent the error in following the reference signal and a second term corresponding to the controller effort. Both terms are then penalized in the cost function to ensure that the reference is followed closely while applying a low control effort. At each time step k, the minimization expressed in (3.11) is computed, subject to the constraints in (3.12).

$$\boldsymbol{u}_k^* = \arg\min_{\boldsymbol{u}_k} J_k(x_k, \boldsymbol{u}_k) \tag{3.11}$$

$$\begin{cases} \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{y}_k = g(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \underline{\boldsymbol{u}} \leq \boldsymbol{u}_k \leq \overline{\boldsymbol{u}} \\ \underline{\boldsymbol{x}} \leq \boldsymbol{x}_{k+1} \leq \overline{\boldsymbol{x}} \end{cases} \tag{3.12}$$

In (3.11), $\boldsymbol{u}_k^*$ represents the optimal control sequence over the prediction horizon. The two first constraints in (3.12) impose the dynamics (see (3.9)), while the two other constraints set limits to the input variables and states (being $\underline{\boldsymbol{u}}$ and $\underline{\boldsymbol{x}}$ the lower limits for the current control input and following state and $\overline{\boldsymbol{u}}$ and $\overline{\boldsymbol{x}}$ the upper limits for the current control input and following state).

Advantages and disadvantages of MPC
MPC is particularly useful in handling constraints on inputs, outputs, and the system itself, making it ideal for controlling systems with hard constraints. In addition, the algorithms can explicitly consider trade-offs between conflicting objectives, such as maximizing production and minimizing energy consumption. It can track a desired reference signal, even with disturbances. Unlike FLC, it can handle linear and nonlinear systems, making it a flexible control strategy for many applications. By using predictions of the future behavior of the system, MPC algorithms can achieve improved performance compared to traditional control methods, such as PID control. It is essential to highlight that this requires knowledge of an accurate model of the system. Lastly, MPC-based controllers can provide real-time control for complex systems, making them ideal for robotics.

MPC-based controllers also pose a list of disadvantages. The biggest obstacle to real hardware implementation is computational complexity. MPC algorithms require solving an optimization problem at each time step, which can be computationally intensive, especially for large and complex systems. This can be a problem for real-time control applications requiring fast updates. As previously mentioned, it requires a mathematical model of the system being controlled, which can be complex to develop and maintain. Inaccuracies in the model can lead to poor performance or instability of the control system. Furthermore, these algorithms are sensitive to the initial conditions and may require a warm-up period to converge to a satisfactory solution. Setting up the hyperparameters is definitely a challenge. If the prediction horizon is too short, the algorithm may be too reactive, while if it is too long, it may be too slow to respond to changes in the system. Furthermore, tuning the optimization parameters and cost functions is required to achieve the desired control performance. Therefore, expert knowledge is also required. Besides, MPC algorithms are usually sensitive to disturbances and may require additional robustness measures, such as state and input constraints, to ensure stability. It is concluded that implementing MPC algorithms can be challenging, and there are many implementation details that can affect the performance and stability of the control system.

An overview of the advantages and disadvantages of MPC is provided in Table 3.4

Table 3.4: Overview of the advantages and disadvantages of MPC. Own elaboration.

Advantages of MPC	Disadvantages of MPC
• Optimality	• Computational complexity
• Handling constraints	• Modeling complexity
• Explicit consideration of trade-offs	• Initialization issues
• Precise reference tracking	• Tuning (e.g. prediction horizon)
• Flexibility	
• Real time control	

Examples of MPC in robotics
As in the case of FLC, MPC can be applied in trajectory tracking and obstacle avoidance. It can be used to implement a control strategy that tracks a desired reference trajectory for the position and orientation of the robot, considering constraints on the speed and acceleration of the robot. Carron et al. [65] apply this knowledge to a robotic arm, showing that their data-driven MPC controller yields improvements compared to traditional PID controllers and even non-data-driven MPC schemes. Furthermore, it can be used to implement a control strategy that avoids obstacles in the environment while tracking a desired reference trajectory. In doing this, it uses obstacle information to generate control inputs that steer the robot around obstacles while maintaining a smooth trajectory, as in the work of Lim et al. [66].

One of the most distinguishing features of MPC is that it can be used to implement trade-offs, including control strategies that optimize the power consumption of robots. The MPC algorithm can consider the

energy consumption of the robot, speed, and other constraints to generate control inputs that result in the most energy-efficient motion. Cho et al. [67] apply this reasoning to saving energy consumption in hydraulic-legged robots. The proposed method showed significantly improved energy efficiency compared to the existing pump control methods in simulation and laboratory experiments.

As can be seen from the examples in the current subsection, MPC provides a solid framework for controlling robots and optimizing their operation by implementing trade-offs, making it a state of the art control approach in robotics.

3.4.3. Search algorithms

Search algorithms are a family of algorithms designed to locate and sort data. They have been successfully used in a wide range of contexts. The most significant developments in this field date to the 1970s with the rise of computer science. In this chapter, the fundamentals of search algorithms in general, grid-based methods, and sampling-based methods are given in Section 3.4.3, Section 3.4.3 and Section 3.4.3, respectively. Then, Section 3.4.3 goes through the advantages and disadvantages of using search algorithms for path planning. Lastly, Section 3.4.3 provides examples of how search algorithms can be used to plan a path to be followed by robots.

Fundamentals of search algorithms

A search algorithm is a set of rules or instructions to find specific information from a collected dataset. They have been developed over several centuries, and their applications range from mathematics to computer science and operations research. In robotics, search algorithms are commonly used to find the shortest path connecting two graph points. Fu et al. [68] have reviewed the state of the art algorithms for solving the shortest path problem. Most algorithms used for this purpose are heuristic, as they make use of heuristic functions to estimate the cost of reaching the goal from each node in the graph.

A heuristic is a general cognitive framework designed to solve a complex problem with an intuitive approach. Heuristic methods include using trial and error or rules of thumb to speed up finding a satisfactory solution. Thus, they are relevant when a short-term approximation is prioritized over the perfect solution. In computer science, they are widely used to ease the computational complexity of algorithms. According to Edward A. Silver et al. [69], reasons to use heuristic methods include mathematical problems that do not have analytic solutions or situations where it is computationally prohibitive to reach analytic or iterative solutions. Heuristic techniques are also appropriate in cases where it is beneficial for the decision-maker to understand how the solution is reached and to develop an intuitive feeling as to which variables are particularly important.

Grid-based methods

Grid-based methods are algorithms that discretize the space by dividing it into a grid of cells. They are suitable for small-scale environments with well-defined boundaries where a discrete representation of the space is feasible. They are commonly used for tasks where the robot must navigate a maze-like environment. Examples of grid-based methods include Dijkstra's algorithm, A*, the Bellman-Ford algorithm, Breadth-First Search (BFS), and Depth-First Search (DFS). This subsection focuses on Dijkstra's and A* algorithms due to their increased relevance in robotics applications.

Dijkstra's algorithm Dijkstra's algorithm has been named after its inventor, Dutch computer scientist Edsger W. Dijkstra. As inputs, it requires a weighted, directed graph G and a starting vertex s. The outputs consist of the length of the shortest path from s to v for every vertex v in G, and the shortest path tree (SPT), which is a subgraph of G that includes all vertices reachable from s and the edges that form the shortest paths from s to each reachable vertex.

The algorithm works by maintaining a set of visited nodes and a set of unvisited nodes. Initially, all nodes are unvisited, and the algorithm assigns an initial distance value of infinity to every node except for the starting node, which has a distance value of zero. The unvisited node with the smallest distance value is selected at each algorithm step and added to the visited node-set. Then, all of its neighbors in the adjacent edges are examined, and their distance values are updated if a shorter path is found. The described steps are repeated until the destination node is visited or until there are no more nodes to visit. Through the steps, a data structure is used to keep the nodes in order based on their distance values. Thus, this priority queue is the mechanism that keeps track of the node with the smallest distance value.

The algorithm is summarized in its procedural form below.

Algorithm 1: Dijkstra's Algorithm

Input: G, s
Output: distSet, shortest path tree formed by edges with shortest distances in distSet
$SPTSet \leftarrow \{s\}$
distSet $\leftarrow$ initialize all distances to ∞, except distSet$[s] \leftarrow 0$
while $SPTSet$ *does not contain all vertices in* G **do**
  $u \leftarrow$ vertex in G with minimum distance in distSet that is not already in $SPTSet$
  Add u to $SPTSet$
  for *each neighbor* v *of* u *that is not in* $SPTSet$ **do**
    $dist \leftarrow$ distSet$[u]$ + weight(u, v)
    if $dist <$ *distSet*$[v]$ **then**
      distSet$[v] \leftarrow$ dist
    end
  end
end
return *distSet*

Dijkstra's algorithm is guaranteed to find the shortest path between the starting node and all other nodes in the graph, provided that no edges have a negative weight. In the case that the weights represent real distances, as in the classic case of planning a path for a robot to follow, this condition is guaranteed to be respected, and the effectiveness of the algorithm is proved. However, under specific formulations, Dijkstra's algorithm would not be suitable. A meaningful, practical example in path planning is the case where a driver has the option to pay a toll which can reduce the overall travel time or distance. If the weights represent the offset to a base distance, negative weights can represent traveling less than expected, whereas positive weights may refer to distances longer. Thus, to use this algorithm, it is essential to formulate the problem such that no negative edge weights exist, to avoid causing infinite loops where the algorithm keeps revisiting the same nodes.

A* algorithm The A* algorithm is a popular heuristic search algorithm that expands on the advantages of Dijkstra's algorithm. It uses an admissible heuristic function to guide the search toward the goal node while exploring the search space efficiently. An admissible heuristic is a function that underestimates the distance to the goal node, i.e., it never overestimates the distance. By using an admissible heuristic, the A* algorithm can avoid exploring paths that are unlikely to lead to the goal node and focus on those paths that are more promising. The A* algorithm maintains two nodes during the search, an open and a closed set. The open set contains the nodes that have been discovered but not explored, while the closed set contains the ones that have already been explored.

Initially, the open set contains only the starting node. At each step of the algorithm, the A* algorithm selects the node with the lowest estimated cost $f(n) = g(n) + h(n)$, where $g(n)$ is the cost of the path from the starting node to node n and $h(n)$ is the estimated cost from node n to the goal node. The algorithm then expands the selected node by examining its neighbors and adding them to the open set if they have not been explored yet. Furthermore, it updates the cost values from the starting node to node n and from node n to the goal for each neighbor node as they are being examined. The A* algorithm terminates when the goal node is found or the open set becomes empty. If the goal node is found, the algorithm reconstructs the optimal path from the starting node to the goal node by following the parent pointers of each node, starting from the goal node and ending at the starting node. The optimal path has the lowest cost among all possible paths.

The procedural form of the algorithm can be found below.

One of the main advantages of the A* algorithm is its efficiency, especially when the heuristic function is well-designed and informative. An excellent heuristic function can significantly reduce the number of nodes that need to be explored, making the algorithm much faster than other search algorithms. However, designing a good heuristic function can be challenging and require domain-specific knowledge and insights.

Algorithm 2: A* algorithm

Input: start, goal, successors, cost, heuristic
Output: path, g(goal)
$open \leftarrow$ {start}
$closed \leftarrow \{\}$
g(start) $\leftarrow 0$
f(start) $\leftarrow$ heuristic(start)
while $open \neq \{\}$ **do**
  $current \leftarrow$ node in $open$ with lowest f score
  if $current =$ *goal* **then**
    $path \leftarrow$ reconstruct path from start to goal
    return *path*, g(*goal*)
  end
  move $current$ from $open$ to $closed$
  for $neighbor$ ***in*** *successors*(*current*) **do**
    if $neighbor$ ***in*** $closed$ **then**
      continue
    end
    $tentative\_g \leftarrow g$(current) + cost(current, neighbor)
    if $neighbor$ ***not in*** $open$ **then**
      add $neighbor$ to $open$
    end
    else if $tentative\_g \geq g$(*neighbor*) **then**
      continue
    end
    $parent$(neighbor) $\leftarrow$ current
    g(neighbor) $\leftarrow tentative\_g$
    f(neighbor) $\leftarrow g$(neighbor) + heuristic(neighbor)
  end
end
return *failure*

Sampling-based methods
Sampling-based methods sample the environment and then use probabilistic algorithms to plan a path that avoids obstacles. They are suitable for large-scale environments where creating a complete map in advance is challenging. They can handle complex and dynamic environments and are particularly suitable for tasks where the robot needs to explore new or unknown areas. Examples of such algorithms include Rapidly-exploring Random Tree (RRT) and Probabilistic Roadmap (PRM). For this report, RRT is explained in more detail due to its predominance in literature and to exemplify how sampling-based methods differ from grid-based methods.

RRT algorithm The RRT algorithm is designed to efficiently explore a high-dimensional space to find a feasible path between a start and goal configuration. The basic idea behind the RRT algorithm is to construct a tree of configurations that grows from the start configuration towards the goal configuration. At each iteration of the algorithm, a random configuration is generated in the state space, and the nearest configuration in the tree is found. An attempt is made to extend the tree from the nearest configuration toward the random configuration. If the extension leads to a new configuration close enough to the goal configuration, the algorithm terminates and returns a viable path.

The algorithm in its procedural form is given below.

The RRT algorithm has several advantages over other motion planning algorithms, including its ability to handle high-dimensional spaces and explore narrow passages in the state space. However, in some cases, the algorithm may require many iterations to find a viable path, and it may not always find the optimal path. There are several variants of the RRT algorithm, including the RRT* algorithm, which is an

Algorithm 3: RRT algorithm

```
Function RRT(start, goal, max_iter, step_size, goal_radius):
    tree.add_node(start)
    for i ← 1 to max_iter do
        q_rand ← sample random config
        q_near ← nearest neighbor(q_rand, tree)
        q_new ← extend(q_near, q_rand, step_size)
        if distance(q_new, goal) < goal_radius then
            tree.add_node(q_new)
            tree.add_edge(q_new, goal)
            return construct path(start, goal, tree)
        end
        tree.add_node(q_new)
        tree.add_edge(q_near, q_new)
    end
    return None
```

extension of the basic RRT algorithm that seeks to improve the quality of the solution by using a different cost function and rewiring the tree structure during the search. Other variants of the RRT algorithm include the bidirectional RRT, which searches for a path from both the start and goal configurations simultaneously, and the informed RRT*, which uses a heuristic function to guide the search toward the goal configuration.

Advantages and disadvantages of search algorithms

Search algorithms are a powerful tool for path planning in robotics. These algorithms offer several advantages; the main advantage is that they guarantee completeness and optimality under certain conditions. In this context, being complete means that it can find a feasible solution if one exists, whereas being optimal refers to finding the best solution possible. For example, Dijkstra's algorithm is complete and optimal when used with a non-negative cost function, as described in Section 3.4.3. Furthermore, search algorithms are flexible and can be adapted to various problem domains. Grid-based methods and sampling-based methods can be applied to path planning in two-dimensional spaces and three-dimensional spaces with varying degrees of complexity. It is also worth noting that they can be highly efficient in finding solutions to path-planning problems. The A* algorithm is an excellent example of an algorithm that has been tailored to be efficient, as it uses a heuristic function to guide the search toward the goal, improving on Dijkstra's algorithm.

Nevertheless, there are some disadvantages to using search algorithms for path planning. Firstly, routing problems typically have combinatorial complexity, meaning that the number of possible solutions grows exponentially with the size of the problem. Thus, search algorithms struggle to find optimal solutions in such cases and can become computationally expensive. Moreover, these algorithms can be sensitive to the environment and the problem formulation. For instance, grid-based methods are sensitive to the size and resolution of the grid, while sampling-based methods are sensitive to the distribution of samples. Lastly, they struggle to handle uncertainty and are not designed to re-plan in real-time when obstacles or other environmental factors change.

Despite these challenges, search algorithms remain a valuable tool for path planning in robotics. Only by carefully considering the advantages and disadvantages of search algorithms is it possible to select the best approach for the specific problem domain and application. Table 3.5 summarizes the mentioned advantages and disadvantages of search algorithms.

Table 3.5: Overview of the advantages and disadvantages of search algorithms. Own elaboration.

Advantages of search algorithms	Disadvantages of search algorithms
• Completeness and Optimality	• Combinatorial Complexity
• Flexibility	• Sensitivity to Environment
• Efficiency	• Difficulty in Handling Uncertainty

Examples of search algorithms in robotics
Search algorithms are essential tools for solving various problems in robotics. They have been applied in various robotic systems to facilitate path planning, obstacle avoidance, and motion planning. They offer a robust and efficient approach to addressing the challenges of robot navigation in complex environments.

The biggest application of search algorithms in robotics is path planning. Algorithms such as A*, Dijkstra, RRT, and PRM are commonly used to find the optimal or feasible path in high-dimensional configuration space. Examples include the work of Kavraki et al. [70], where PRM, combined with a learning algorithm, is used to plan paths in high-dimensional environments in fractions of a second. In obstacle avoidance, RRT has been used frequently to find a collision-free path for robots. In the study by Bouzid et al. [71], an extension of RRT, named multi-RRT* fixed node, is successfully used to plan the motion of a quadrotor in a cluttered environment. In addition, search algorithms have been applied to motion planning in robotics, as in the case of the study conducted by Guo and Parker [72], that opted for a search method in an attempt to plan not only paths for multiple robots but also to plan their velocities. The algorithm was then implemented and successfully run on indoor ground robots.

Search algorithms have been successfully applied in various robotic systems to facilitate path planning, obstacle avoidance, and motion planning. Due to their efficiency, it is expected that the use of search algorithms will continue to grow in the field of robotics and play an increasingly important role in developing autonomous systems.

3.4.4. Discussion
Fuzzy Logic Control, Model Predictive Control, and Search Algorithms are all powerful techniques in robotics that enable intelligent decision-making in various tasks. The dissected literature shows that they are mainly used for path planning, motion planning, and obstacle avoidance. Applications further include object manipulation, adaptive control, and human-robot interaction.

FLC has been applied in numerous robotic applications due to its ability to handle uncertainty and imprecision. It provides a flexible way to design control systems based on expert knowledge or empirical data due to its simple and straightforward formulation. Since its introduction in 1965, FLC has been applied to numerous fields. It shows great promise in robotics applications requiring sensors, actuators, and algorithms to achieve the desired outcome. An identified research gap is the need to bridge the gap between FLC and deep learning, as suggested by Fan [73]. FLC and deep learning can benefit from this collaboration, as fuzzy logic can assist in re-examining heuristics and reformulating neural computing to increase efficiency, whereas deep learning can tune the initial parameterization conducted by an expert.

MPC is another popular control technique widely used in robotics applications because it handles non-linear dynamics and constraints. It is a powerful optimization-based control approach that can plan and execute a sequence of control actions over a finite horizon while considering constraints and uncertainties. Compared to FLC, it offers a more rigorous mathematical formulation and the possibility to consider trade-offs explicitly. In the context of SaR, this is particularly useful to minimize the search time for victims and find the most significant possible number of victims while avoiding obstacles. Its primary disadvantage in real-life applications is its computational complexity. It is expected that MPC remains one of the most solid and robust control algorithms and that research in this area is oriented toward reducing its computational cost.

Search Algorithms are used in robotics mainly for path optimization. They provide a systematic way to explore the configuration space and find an optimal or feasible path to reach the goal. They can mathematically guarantee completeness and optimality in certain conditions, making them a robust solution. Nevertheless, it is crucial to review their formulation when applying them to large-scale problems, as combinatorial complexity tends to increase exponentially with the increase of the graph size. The literature review shows that recent work in heuristic search algorithms involves combining them with learning algorithms to look for similarities between different problems and decrease computational time.

In conclusion, FLC, MPC, and search algorithms are all promising algorithms for intelligent decision-making in robotics. The choice of technique depends on the desired mathematical rigor, the availability of a model, and computational power. Research gaps in control strategies and path planning algorithms include integrating learning algorithms and bridging the gap between theory and practice to prove their effectiveness in recent complex problems, as highlighted in [70], [73], [74].

3.5. Conclusion

The Search and Rescue (SaR) industry stands to benefit substantially from the usage of robotics. Robots have the potential to offer a fast and automated response in the aftermath of disasters, which is vital in saving human lives. Furthermore, they are able to access remote areas and mitigate the risks to human rescuers by substituting them in traversing hazardous environments. Nevertheless, assessing their performance and reliability in laboratory settings is essential before trusting robots with such a vital task. Recently, the focus of researchers has included creating heterogeneous teams of robots that complement their characteristics for the benefit of the mission and implementing and testing these setups. With this in mind, this literature study aims to provide supporting knowledge to fulfill the research objective of **developing an affordable autonomous UGV-UAV collaborative team to create a global map of the locations of victims in SaR scenarios**.

The literature review provides a comprehensive overview of the state of the art of image processing methods, one of the main focuses of the work. In order to ensure that the UGV and UAV collaborate effectively, it is essential to understand how they can detect victims and obstacles and merge the information to create a global map for the mission. The literature study then delves into communication protocols, guaranteeing that information is transmitted and exchanged as expected. As the reader becomes familiar with image processing techniques and communication frameworks that allow a global map of the SaR scenario to be created and transmitted, it is then indispensable to plan the path and motion for the UGV to assist the victims while avoiding obstacles. The literature study reviews the state of the art of control strategies and path-planning algorithms for that purpose. By synthesizing these topics, the literature study offers a thorough understanding of the potential of implementing autonomous UGV-UAV collaborative SaR teams.

Image Processing Advancements in computer science have allowed computers to detect and classify objects in photographs. Initially, this was exclusively accomplished using traditional computer vision methods that required expert knowledge in defining features that characterize each object to be searched in the images. Recently, artificial intelligence has been revolutionizing the field of computer vision by creating Convolutional Neural Networks (CNNs), a type of neural network specifically designed for image processing. Deep learning algorithms are able to extract outputs directly from raw data accurately, provided that they are trained with significant amounts of labeled images. The comparison conducted by O'Mahony et al. [36] clearly illustrates the differences between the two workflows. Since collecting labeled images can be too time-consuming for some applications, deep learning has still not made traditional computer vision algorithms obsolete. Nevertheless, current research aims to develop even more sophisticated and efficient deep learning algorithms, including Faster Regional-based Convolutional Neural Networks (R-CNNs) and You Only Look Once (YOLO). Regarding fusing images taken from two different viewpoints, the state of the art includes Siamese networks, which automatically detect how similar two inputs are. In object scaling, stereo vision is state of the art, as it analyses the disparities between different images to estimate the depth of the objects in the scene. The relevance of the photographs depends on the image quality, the context in which they were captured, and the scope. Combining Support Vector Machines (SVMs) and metrics such as Peak Signal-to-noise Ratio (PSNR) or Structure Similarity Index (SSIM) constitutes novelty in the field by mapping objective quality to subjective quality assessments.

Communication Protocols Without effective communication, combining robots to achieve a common goal is impossible. Communication protocols should be designed to enable the fast and robust exchange of information. In order to choose a suitable communication protocol for a given application, it is crucial to consider if communication is unidirectional or bidirectional, the available bandwidth, maximum latency allowed, scalability, security, flexibility, and robustness to transmission failures. Sequential deployment of robots does not require strict communication considerations, as information must not be exchanged in real time. In the case of concurrent deployment of robots, communications are more dynamic and complex, as robots communicate in real time, and latency should be minimal. The work of Fankhauser et al. [4] showcases the challenges of switching from a sequential to a concurrent deployment setting. Communication protocols between two robots can involve direct communications between them or also with an external computer. Wi-Fi remains the most common means of communication due to its availability and simplicity. However, investment in the field of IoT led to the creation of more sophisticated frameworks for

real time communication, including Message Queuing Telemetry Transport (MQTT), Extensible Messaging and Presence Protocol (XMPP), and Wireless Sensor and Actuator Network (WSAN).

Control Strategies and Path Planning In any SaR mission, it is necessary to establish a path to rescue all victims in the most efficient way, to save precious time. In the case of autonomous SaR robotics, it is even more crucial to define a suitable path for the robots to follow and to plan their motion to track it as closely as possible. Given that the world of control theory is vast, the literature study focused on the most promising solutions that have been applied to SaR robotics in the past, namely Fuzzy Logic Control (FLC), Model Predictive Control (MPC) and search algorithms. The literature study clarifies that all three control methods are encouraging in the field of intelligent decision-making in robotics. The selection depends mainly on the desired mathematical rigor and available computational power. FLC is the most suitable choice in cases where simplicity is a priority. If the problem requires handling hard constraints and making trade-offs, MPC is the adequate solution. In cases where the problem is exclusively path planning and not motion planning, and the size of the problem is not too large, search algorithms can often guarantee completeness and optimality and therefore provide a reliable solution. Therefore, all of them are currently actively researched due to their substantial potential in the field of robotics.

Closing Remarks In conclusion, this literature review has successfully addressed all research questions and has presented a comprehensive overview of SaR robotics. In addition, it has identified significant research gaps that future researchers can explore to advance the field further. The research gaps identified include the need to optimize sophisticated deep-learning object detection and matching algorithms to require fewer data. Such an achievement will make traditional computer vision algorithms obsolete and will allow fast and reliable image processing, even for non-experts. Regarding communication protocols, research gaps include the need to develop safe and scalable communication frameworks for IoT applications. In the field of robotics, it is increasingly popular to use multi-robot architectures. Scalability issues are even more prone to occur due to the expected miniaturization of robots in the future, as discussed by Sitti [75]. With the current development of society, various robotic applications are raising ever higher requests for solving constrained optimization problems. Thus, more efforts should be made in MPC research to bridge the gap between MPC theory and applications, as highlighted by Xi et al. [74]. Another interesting research gap is to bridge the gap between FLC and deep learning. According to Fan [73], research in FLC has declined since 1998, partially due to a lack of convincing applications in complex machine learning problems. However, there is an opportunity to use FLC to re-examine heuristics and reformulate neural computing to achieve more efficient algorithms in practice. The literature review also suggests combining search algorithms with learning algorithms to look for similarities between different problems and decrease computational times. The present review is valuable for researchers and industry professionals seeking to broaden their knowledge in this domain.

Part III

Closure

4

Conclusion

In this chapter, the main conclusions of the thesis are provided. Firstly, in Section 4.1, the research objective and research questions are revisited to reflect on the extent to which they can be answered. Subsequently, in Section 4.2, concluding remarks on the project are given in a general tone.

4.1. Revisiting Research Formulation

The research objective, as well as the research questions posed in Chapter 1 are repeated below for convenience. A reflection is then provided one by one to get an overview of the success of the work. The research objective included leveraging the complementary capabilities of flying and ground robots to deliver an affordable and effective SaR solution in mapping unknown disaster areas. For this purpose, a few typical SaR tasks were performed experimentally, namely estimating the distance of a human from the robot from photographs using pose estimation, tracking the trajectory of an object by applying data fusion, and estimating the terrain elevation. To ensure a simplistic setup and to keep the hardware costs low, these tasks were performed by requiring visual depiction only through photographs and videos. Due to the emergence of YOLO as one of the leading object detection algorithms, as concluded from the literature study, it was incorporated into the framework. Therefore, the research objective was formulated as follows:

Research Objective

Investigate the attainable accuracy of a collaborating flying and a ground robot team equipped with RGB cameras and employing YOLO in performing conventional SaR mapping tasks.

The research objective led to the development of a modular Python program consisting of several functionalities divided by task. The evolution of the program occurred hand-in-hand with frequent testing, making use of real photographs and videos from the robots and the laboratory facilities to validate the algorithms. Prior to the experiments, several programs were installed and tested to ensure the smooth flow of information and command exchange between the external computer, the Parrot Bebop 2, and the Parrot Jumping Sumo. In addition, safety briefings were attended to guarantee the safety of the experiments, as well as the safety of the participants. The first research question, related to the detection and depth estimation of participants, making use of pose estimation as a supporting algorithm, was formulated as given below.

Research Question 1

Is leveraging pose estimation techniques and camera equations a robust and accurate approach to performing human depth estimation in photographic images?

The experimental sessions involved 24 participants in the CyberZoo, and estimating their depths (i.e., their distance from the SUMO Jumping Robot) for different poses and distances to the camera of the UGV. The experiment revealed the potential of the algorithm to accurately estimate human depth from photographs, yet unveiled its susceptibility to variations in poses. Even though most of the average

relative errors originating from the algorithm are below 10%, they can also soar above 20% in specific circumstances. Nevertheless, auspicious results include a module to estimate missing key points, which indeed shows increased robustness. Even in situations where some body parts of the participants were occluded, such as the shoulders and the face, the algorithm practically did not showcase a reduction in accuracy. Compared to more simplistic algorithms that simply take the width or height of the object into consideration when estimating its depth, which is much more prone to faulted predictions due to occlusion, the proposed solution represents a significant improvement. Moreover, the fact that the estimation can be done through several different reference measurements and can be fused also offers increased robustness, as the algorithm is less dependent on a specific measurement (e.g., the height). However, the accuracy values obtained are contingent on well-lit scenarios and were only tested in distances up to 3 m. It is expected that the accuracy significantly drops in poorly lit scenarios, scenarios with several occlusions, or even at larger distances where the resolution of the camera plays a bigger role. Thus, in the eventuality that the UGV performs this task by itself, the need to integrate the proposed system into a broader depth estimation module, which includes other forms of sensing, such as thermal imaging and acoustic sensing, remains a priority. In order to tackle this difficulty, a UAV assists the UGV in tracking the trajectories of objects, giving origin to the second research question, specified below.

Research Question 2

How much improvement does a UGV-UAV collaboration offer in performing object tracking compared to an individual operation within realistic SaR scenarios?

The estimated trajectories were evaluated based on three performance metrics, namely the Root Mean Square Error (RMSE), the extreme deviation from the true trajectory, and the area ratio, which represents the portion of the area under the true trajectory that is covered by the estimated trajectory. When tracking the trajectories of objects, the collaborative advantage of deploying both aerial and ground robots is evident in keeping the overall shape of the trajectory close to the true trajectory. Nevertheless, this is not always reflected in the metrics, reason why visual inspection of the trajectories should accompany the analysis. The collaborative setup is more fault-safe in keeping a low RMSE, as it is frequently close to the average of the RMSE obtained with the deployment of each of the robots individually. The collaboration is extremely effective at preventing large extreme deviations, especially in cluttered environments. Overall, for unobstructed narrow-area trajectories, the deployment of one robot suffices as the joint deployment of the robots offers practically no improvements. Particularly, the standalone UAV offers more encouraging results when compared to the standalone UGV. However, for wide-area trajectories or in the presence of obstacles, the advantages of deploying both robots are evident. In the widest trajectory simulated and in the presence of obstacles to the aerial view, the deployment of both robots showcased a reduction of 28% in the extreme deviation of the true trajectory and an increase of 42% in the area covered ratio compared to the best-performing robot. The ability to navigate obstacles and to maintain precision in diverse environments remains a challenge that demands further attention, as it is expected to further improve the collaborative setup. Lastly, the setup could be extended to compute an elevation profile of the terrain, giving rise to the third research question, as formulated below.

Research Question 3

To what extent can a collaborative UGV-UAV system generate a dependable and comprehensive elevation map of diverse terrains?

The proposed setup for elevation mapping faces two primary elevation error sources: depth estimation and subsequent elevation estimation. Depth estimation inaccuracies stem from simplified camera models and low image resolution, impacting object detection, compounded by occasional object occlusion. Elevation errors also result from poor homography calibration in specific regions, exacerbated by the double application of the simplified camera model. Without UAV assistance, elevation relative errors range from 0% to 35%, dropping to under 13% with UAV help, except for one instance reaching 35%. This instance occurs in a poorly calibrated area regarding homography calibration, highlighting the importance of having diverse reference points in the calibration step. These findings, observed in well-lit scenarios with aligned UGV-object settings, are expected to degrade in more challenging environments. The suitability of the

approach varies: it does not meet high-precision needs but could find utility in SaR missions contingent on terrain characteristics and coverage area. Lastly, in the current approach, the UAV carries only one object at a time and needs to pick it up and transport it somewhere else for the elevation at a different point to be estimated. Therefore, future work is essential to answer about the scalability of the approach.

4.2. Closing Remarks

Collectively, these experiments represent notable progress in leveraging visual data and robotic capabilities for SaR. However, they also serve as a poignant reminder of the multifaceted nature of real-world scenarios, where dynamic and unpredictable conditions demand adaptable, versatile, and resilient systems. While running these experiments in real-time seems realistic, running them onboard is more daunting, especially for the elevation mapping task. Moving forward, bridging the gap between experimental findings and real-world deployment remains imperative. Addressing the identified limitations by refining algorithms, enhancing collaborative robot functionalities, and devising adaptable strategies tailored to diverse SaR scenarios will be pivotal. In essence, while these experiments mark significant strides in harnessing visual-based techniques within robotic systems for SaR missions, they contribute to an ongoing journey towards more effective, adaptable, and accessible solutions vital for saving lives.

References

[1] A. Pretto, S. Aravecchia, W. Burgard, N. Chebrolu, C. Dornhege, T. Falck, F. Fleckenstein, A. Fontenla, M. Imperoli, R. Khanna, F. Liebisch, P. Lottes, A. Milioto, D. Nardi, S. Nardi, J. Pfeifer, M. Popović, C. Potena, C. Pradalier, E. Rothacker-Feder, I. Sa, A. Schaefer, R. Siegwart, C. Stachniss, A. Walter, W. Winterhalter, X. Wu, and J. Nieto, "Building an aerial–ground robotics system for precision farming: An adaptable solution," *IEEE Robotics & Automation Magazine*, volume 28, number 3, pages 29–49, Sep. 2021 (cited on page 1).

[2] C. Potena, R. Khanna, J. Nieto, R. Siegwart, D. Nardi, and A. Pretto, "Agricolmap: Aerial-ground collaborative 3d mapping for precision farming," *IEEE Robotics and Automation Letters*, volume 4, number 2, pages 1085–1092, Apr. 2019 (cited on page 1).

[3] B. Hament and P. Oh, "Unmanned aerial and ground vehicle (uav-ugv) system prototype for civil infrastructure missions," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA: IEEE, Mar. 2018, pages 1–4 (cited on page 1).

[4] P. Fankhauser, M. Bloesch, P. Krüsi, R. Diethelm, M. Wermelinger, T. Schneider, M. Dymczyk, M. Hutter, and R. Siegwart, "Collaborative navigation for flying and walking robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea (South): IEEE, Dec. 2016, pages 2859–2866 (cited on pages 1, 33, 36, 37, 39, 52).

[5] G.-J. M. Kruijff, F. Pirri, M. Gianni, P. Papadakis, M. Pizzoli, A. Sinha, V. Tretyakov, T. Linder, E. Pianese, S. Corrao, F. Priori, S. Febrini, and S. Angeletti, "Rescue robots at earthquake-hit mirandola, italy: A field report," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, College Station, TX, USA: IEEE, Nov. 2012, pages 1–8 (cited on page 1).

[6] S. Hood, K. Benson, P. Hamod, D. Madison, J. M. O'Kane, and I. Rekleitis, "Bird's eye view: Cooperative exploration by ugv and uav," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, Miami, FL, USA: IEEE, Jun. 2017, pages 247–255 (cited on page 1).

[7] R. T. Newkirk, "The increasing cost of disasters in developed countries: A challenge to local planning and government," *Journal of Contingencies and Crisis Management*, volume 9, number 3, pages 159–170, Dec. 2001 (cited on pages 1, 26).

[8] M. Zorn, *Natural Disasters and Less Developed Countries*, S. Pelc and M. Koderman, Eds. Springer International Publishing, Aug. 2018, pages 59–78 (cited on pages 2, 26).

[9] J. P. Zelten, "Digital photography and the dynamics of technology innovation," Ph.D. dissertation, Feb. 2002 (cited on page 2).

[10] I. C. Condotta, T. M. Brown-Brandl, S. K. Pitla, J. P. Stinn, and K. O. Silva-Miranda, "Evaluation of low-cost depth cameras for agricultural applications," *Computers and Electronics in Agriculture*, volume 173, page 105 394, Jun. 2020 (cited on page 2).

[11] A. P. Hill, P. Prince, J. L. Snaddon, C. P. Doncaster, and A. Rogers, "Audiomoth: A low-cost acoustic device for monitoring biodiversity and the environment," *HardwareX*, volume 6, e00073, Oct. 2019 (cited on page 2).

[12] H. H. Titi, *Feasibility Study for a Freeway Corridor Infrastructure Health Monitoring (HM) Instrumentation Testbed*. Wisconsin DOT Research & Library Unit, Jul. 2012 (cited on page 2).

[13] L. Zhaohua and G. Bochao, "Radar sensors in automatic driving cars," in *2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT)*, Nanchang, China: IEEE, Oct. 2020, pages 239–242 (cited on page 2).

[14] S. Hummel, A. Hudak, E. Uebler, M. Falkowski, and K. Megown, “A comparison of accuracy and cost of lidar versus stand exam data for landscape management on the malheur national forest,” *Journal of Forestry*, volume 109, pages 267–273, Jul. 2011 (cited on page 2).

[15] D. Van Nam and K. Gon-Woo, “Solid-state lidar based-slam: A concise review and application,” in *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju Island, Korea (South): IEEE, Jan. 2021, pages 302–305 (cited on page 2).

[16] R. R. Murphy, S. Tadokoro, and A. Kleiner, *Disaster Robotics*, B. Siciliano and O. Khatib, Eds. Springer International Publishing, Jan. 2016, pages 1577–1604 (cited on page 26).

[17] J. Qi, D. Song, H. Shang, N. Wang, C. Hua, C. Wu, X. Qi, and J. Han, “Search and rescue rotary-wing uav and its application to the lushan ms 7.0 earthquake,” *Journal of Field Robotics*, volume 33, number 3, pages 290–321, Jul. 2016 (cited on page 26).

[18] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, “Deep learning for smart manufacturing: Methods and applications,” *Journal of manufacturing systems*, volume 48, pages 144–156, Jul. 2018 (cited on page 28).

[19] R. Thendral, A. Suhasini, and N. Senthil, “A comparative analysis of edge and color based segmentation for orange fruit recognition,” in *2014 International Conference on Communication and Signal Processing*, Melmaruvathur, India: IEEE, Apr. 2014, pages 463–466 (cited on page 29).

[20] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, “F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds,” in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, Arlington, VA, USA: Association for Computing Machinery, Nov. 2019, pages 88–100 (cited on page 29).

[21] J. Gautham, A. Sharma, S. Dhanalakshmi, and K. Ramamoorthy, “3d scene reconstruction and mapping with real time human detection for search and rescue robotics,” in *AIP Conference Proceedings*, volume 2427, Address to be determined: AIP Publishing LLC, Feb. 2023, page 020 002 (cited on page 29).

[22] E. Karami, M. Shehata, and A. Smith, “Image identification using sift algorithm: Performance analysis against different image deformations,” Oct. 2017 (cited on page 29).

[23] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision*, volume 3951, Graz, Austria: Springer, May 2006, pages 404–417 (cited on page 29).

[24] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision*, Graz, Austria: Springer, May 2006, pages 430–443 (cited on page 29).

[25] A. Reiter, P. K. Allen, and T. Zhao, “Articulated surgical tool detection using virtually-rendered templates,” in *Computer assisted radiology and surgery (CARS)*, Pisa, Italy: Springer, Jun. 2012, pages 1–8 (cited on page 29).

[26] S. Xu, T. Fang, D. Li, and S. Wang, “Object classification of aerial images with bag-of-visual words,” *IEEE Geoscience and Remote Sensing Letters*, volume 7, number 2, pages 366–370, Dec. 2010 (cited on page 29).

[27] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, volume 20, pages 273–297, Sep. 1995 (cited on page 29).

[28] J. Kavitha and A. Suruliandi, “Texture and color feature extraction for classification of melanoma using svm,” in *2016 International conference on computing technologies and intelligent data engineering (ICCTIDE’16)*, Kovilpatti, India: IEEE, Jan. 2016, pages 1–6 (cited on page 30).

[29] J.-j. Wang, D.-a. Zhao, W. Ji, J.-j. Tu, and Y. Zhang, “Application of support vector machine to apple recognition using in apple harvesting robot,” in *2009 International Conference on Information and Automation*, Zhuhai/Macau, China: IEEE, Aug. 2009, pages 1110–1115 (cited on page 30).

[30] Y. Liu, X. Wang, L. Li, S. Cheng, and Z. Chen, “A novel lane change decision-making model of autonomous vehicle based on support vector machine,” *IEEE Access*, volume 7, pages 26 543–26 550, Feb. 2019 (cited on page 30).

[31] J. Kober and R. Babuška, *Knowledge-Based Control Systems*. Jul. 2017 (cited on pages 30, 31, 40–42, 45).

[32] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” Nov. 2015 (cited on page 31).

[33] A. Dhillon and G. K. Verma, “Convolutional neural network: A review of models, methodologies and applications to object detection,” *Progress in Artificial Intelligence*, volume 9, number 2, pages 85–112, Dec. 2020 (cited on page 31).

[34] A. G. Villa, A. Salazar, and F. Vargas, “Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks,” *Ecological informatics*, volume 41, pages 24–32, Sep. 2017 (cited on page 31).

[35] K. Shan, J. Guo, W. You, D. Lu, and R. Bie, “Automatic facial expression recognition based on a deep convolutional-neural-network structure,” in *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, London, United Kingdom: IEEE, Jun. 2017, pages 123–128 (cited on page 31).

[36] N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, “Deep learning vs. traditional computer vision,” in *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC)*, Las Vegas, NV, USA: Springer, Apr. 2020, pages 128–144 (cited on pages 31, 32, 52).

[37] J. Du, “Understanding of object detection based on cnn family and yolo,” in *Journal of Physics: Conference Series*, volume 1004, Hong Kong, China: IOP Publishing, Feb. 2018, page 012 029 (cited on page 33).

[38] I. Melekhov, J. Kannala, and E. Rahtu, “Siamese network features for image matching,” in *2016 23rd international conference on pattern recognition (ICPR)*, Cancun, Mexico: IEEE, Dec. 2016, pages 378–383 (cited on page 33).

[39] M. Marques, M. Stošić, and J. Costeira, “Subspace matching: Unique solution to point matching with geometric constraints,” in *2009 IEEE 12th International Conference on Computer Vision*, Kyoto, Japan: IEEE, Oct. 2009, pages 1288–1294 (cited on page 33).

[40] R. Lagisetty, N. K. Philip, R. Padhi, and M. S. Bhat, “Object detection and obstacle avoidance for mobile robot using stereo camera,” in *2013 IEEE International Conference on Control Applications (CCA)*, Hyderabad, India: IEEE, Aug. 2013, pages 605–610 (cited on page 34).

[41] P. Kuhad, A. Yassine, and S. Shimohammadi, “Using distance estimation and deep learning to simplify calibration in food calorie measurement,” in *2015 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, Shenzhen, China: IEEE, Jun. 2015, pages 1–6 (cited on page 34).

[42] D. R. I. M. Setiadi, “Psnr vs ssim: Imperceptibility quality assessment for image steganography,” *Multimedia Tools and Applications*, volume 80, number 6, pages 8423–8444, Nov. 2021 (cited on page 34).

[43] W. Ding, Y. Tong, Q. Zhang, and D. Yang, “Image and video quality assessment using neural network and svm,” *Tsinghua Science and Technology*, volume 13, number 1, pages 112–116, Feb. 2008 (cited on page 34).

[44] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, “Active autonomous aerial exploration for ground robot path planning,” *IEEE Robotics and Automation Letters*, volume 2, number 2, pages 664–671, Jan. 2017 (cited on page 36).

[45] P. De Petris, S. Khattak, M. Dharmadhikari, G. Waibel, H. Nguyen, M. Montenegro, N. Khedekar, K. Alexis, and M. Hutter, “Marsupial walking-and-flying robotic deployment for collaborative exploration of unknown environments,” *arXiv preprint arXiv:2205.05477*, May 2022 (cited on page 36).

[46] Z. Wang, Z. Hu, Y. Man, and M. Fjeld, “A collaborative system of flying and ground robots with universal physical coupling interface (pci), and the potential interactive applications,” in *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, New Orleans, LA, USA: Association for Computing Machinery, Apr. 2022, pages 1–7 (cited on page 37).

[47] T. Miki, P. Khrapchenkov, and K. Hori, “Uav/ugv autonomous cooperation: Uav assists ugv to climb a cliff by attaching a tether,” in *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada: IEEE, May 2019, pages 8041–8047 (cited on pages 37, 38).

[48] N. Lissandrini, C. K. Verginis, P. Roque, A. Cenedese, and D. V. Dimarogonas, “Decentralized nonlinear mpc for robust cooperative manipulation by heterogeneous aerial-ground robots,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA: IEEE, Feb. 2020, pages 1531–1536 (cited on page 38).

[49] B. Zhou, H. Xu, and S. Shen, “Racer: Rapid collaborative exploration with a decentralized multi-uav system,” *arXiv preprint arXiv:2209.08533*, Feb. 2022 (cited on page 38).

[50] S. Ren, Y. Chen, L. Xiong, Z. Chen, and M. Chen, “Path planning for the marsupial double-uavs system in air-ground collaborative application,” in *2018 37th Chinese Control Conference (CCC)*, Wuhan, China: IEEE, Oct. 2018, pages 5420–5425 (cited on page 38).

[51] M. Mukhandi, D. Portugal, S. Pereira, and M. S. Couceiro, “A novel solution for securing robot communications based on the mqtt protocol and ros,” in *2019 IEEE/SICE International Symposium on System Integration (SII)*, Paris, France: IEEE, Jan. 2019, pages 608–613 (cited on page 38).

[52] S. Bendel, T. Springer, D. Schuster, A. Schill, R. Ackermann, and M. Ameling, “A service infrastructure for the internet of things based on xmpp,” in *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, San Diego, CA, USA: IEEE, Mar. 2013, pages 385–388 (cited on page 38).

[53] D.-I. Curiac, “Towards wireless sensor, actuator and robot networks: Conceptual framework, challenges and perspectives,” *Journal of Network and Computer Applications*, volume 63, pages 14–23, Mar. 2016 (cited on page 38).

[54] A. Ohta, “Sky magic: Drone entertainment show,” in *ACM SIGGRAPH 2017 Emerging Technologies*, Los Angeles, CA, USA: Association for Computing Machinery, Jul. 2017 (cited on page 40).

[55] L. A. Zadeh, “Fuzzy sets,” *Information and Control*, volume 8, number 3, pages 338–353, Jun. 1965 (cited on page 40).

[56] T. Vasconcelos Cabanas Ramos Ascensão, “Adaptive fuzzy logic control applied to socially assistive drones: A case study,” Nov. 2021 (cited on pages 41, 44).

[57] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” *IEEE Transactions on Systems, Man, and Cybernetics*, volume SMC-15, number 1, pages 116–132, Feb. 1985 (cited on page 43).

[58] P. Albertos, A. Sala, and M. Olivares, “Fuzzy logic controllers. methodology. advantages and drawbacks,” in *Congreso Español sobre Tecnologías y Lógica Fuzzy*, Seville, Spain: Publisher not specified, Sep. 2000, pages 1–11 (cited on page 43).

[59] H. Seraji and A. Howard, “Behavior-based robot navigation on challenging terrain: A fuzzy logic approach,” *IEEE transactions on robotics and automation*, volume 18, number 3, pages 308–321, Jun. 2002 (cited on page 44).

[60] A. Pandey, R. K. Sonkar, K. K. Pandey, and D. R. Parhi, “Path planning navigation of mobile robot with obstacles avoidance using fuzzy logic controller,” in *2014 IEEE 8th International Conference on Intelligent Systems and Control (ISCO)*, Coimbatore, India: IEEE, May 2014, pages 39–41 (cited on page 44).

[61] V. Ciobanu and N. Popescu, “Tactile controller using fuzzy logic for robot inhand manipulation,” in *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, Cheile Gradistei, Romania: IEEE, Nov. 2015, pages 435–440 (cited on page 44).

[62] T. Das and I. N. Kar, “Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots,” *IEEE Transactions on Control Systems Technology*, volume 14, number 3, pages 501–510, May 2006 (cited on page 44).

[63] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, “Generalized predictive control—part i. the basic algorithm,” *Automatica*, volume 23, number 2, pages 137–148, Mar. 1987 (cited on page 44).

[64] C. de Koning, “Hierarchical cooperative mission planning of non-homogeneous autonomous search-and-rescue robots,” Jan. 2022 (cited on page 44).

[65] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, “Data-driven model predictive control for trajectory tracking with a robotic arm,” *IEEE Robotics and Automation Letters*, volume 4, number 4, pages 3758–3765, Jul. 2019 (cited on page 46).

[66] H. Lim, Y. Kang, C. Kim, J. Kim, and B.-J. You, “Nonlinear model predictive controller design with obstacle avoidance for a mobile robot,” in *2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, Beijing, China: IEEE, Oct. 2008, pages 494–499 (cited on page 46).

[67] B. Cho, S.-W. Kim, S. Shin, J.-H. Oh, H.-S. Park, and H.-W. Park, “Energy-efficient hydraulic pump control for legged robots using model predictive control,” *IEEE/ASME Transactions on Mechatronics*, volume 28, number 1, pages 3–14, Aug. 2023 (cited on page 47).

[68] L. Fu, D. Sun, and L. Rilett, “Heuristic shortest path algorithms for transportation applications: State of the art,” *Computers & Operations Research*, volume 33, number 11, pages 3324–3343, Nov. 2006 (cited on page 47).

[69] E. A. Silver, R. Victor, V. Vidal, and D. de Werra, “A tutorial on heuristic methods,” *European Journal of Operational Research*, volume 5, number 3, pages 153–162, Sep. 1980 (cited on page 47).

[70] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE transactions on Robotics and Automation*, volume 12, number 4, pages 566–580, Aug. 1996 (cited on page 51).

[71] Y. Bouzid, Y. Bestaoui, and H. Siguerdidjane, “Quadrotor-uav optimal coverage path planning in cluttered environment with a limited onboard energy,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada: IEEE, Dec. 2017, pages 979–984 (cited on page 51).

[72] Y. Guo and L. Parker, “A distributed and optimal motion planning approach for multiple mobile robots,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation*, volume 3, Washington, DC, USA: IEEE, Aug. 2002, pages 2612–2619 (cited on page 51).

[73] L. Fan, “Revisit fuzzy neural network: Bridging the gap between fuzzy logic and deep learning,” Nov. 2017 (cited on pages 51, 53).

[74] Y.-G. XI, D. Li, and S. Lin, “Model predictive control — status and challenges,” *Acta Automatica Sinica*, volume 39, pages 222–236, Mar. 2013 (cited on pages 51, 53).

[75] M. Sitti, “Microscale and nanoscale robotics systems [grand challenges of robotics],” *IEEE Robotics & Automation Magazine*, volume 14, number 1, pages 53–60, Mar. 2007 (cited on page 53).