ACTIVE VISION AND PERCEPTION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jake Richard Gemerek

August 2020

ACTIVE VISION AND PERCEPTION

Jake Richard Gemerek, Ph.D.

Cornell University 2020

Active vision and perception for resource-constrained autonomous vehicles, such as small ground robots and quadrotors, are limited in their allowable algorithmic complexity and slow reaction times. For an autonomous mobile robot to safely and reliably perform a useful task or behavior, real-time visual perception that informs a controller with a fast reaction time is needed. This dissertation covers new research developments in the areas of active vision, planning, and control for directional sensors with a focus on event-cameras and RGB cameras. Event-cameras, also known as neuromorphic cameras, are biologically inspired visual sensors that measure local changes in light intensity, mitigating latency and redundant data. Several high-level active vision algorithms, interfaced with autonomous vehicle controllers, are developed for event-cameras and quantitatively compared to analogous RGB camera algorithms, in terms of both accuracy and computational cost. In particular, motion-based perception algorithms for object recognition and tracking, action recognition, and depth estimation are developed for use on a moving quadrotor tasked with reacting to the perceived environment. Novel active vision algorithms for RGB cameras are also developed in which an autonomous ground vehicle or quadrotor interact with a human target of interest using novel action recognition and tracking perception capabilities paralleled with new control methods for target following. Furthermore, a novel occlusion-avoiding path planning algorithm that is applicable to both event-cameras and RGB cameras is developed. The proposed method computes a closed-form collection of subsets of

the sensor's configuration space, referred to as visibility regions, that quantify the visibility of targets subject to the sensor field of view geometry and line of sigh visibility. This method is quantitatively compared to several existing sensor path planning methods in terms of analytical computational complexity, experimental path performance, and experimental computational cost analysis. The results of this work enable active vision, perception, and planning for resource-constrained mobile robots equipped with directional sensors such as an event-camera or RGB camera.

BIOGRAPHICAL SKETCH

Jake started the Mechanical Engineering PhD program at Cornell in the Fall of 2016 after graduating with a dual Aerospace and Mechanical Engineering degree from the University at Buffalo. His research interest is in the area of autonomous robotic systems with a focus on real-time applications. He is an active member of the IEEE, and has published conference papers at the top-tier IEEE robotics and autonomous systems meetings. At Cornell, Jake was named a 2019 Commercialization fellow by the college of engineering, in partnership with the Johnson Business school, and he was awarded the National Science Foundation Innovation-Corps grant to pursue commercial applications of his research.

To my loving and supportive wife, Kathryn.

ACKNOWLEDGEMENTS

I would like to acknowledge the members of the Laboratory for Intelligent Systems and Controls for their long conversations about my research which has helped me to make this research effort ever more powerful. I would also like to thank my advisor and mentor, Silvia Ferrari, who is also the director of the Laboratory for Intelligent Systems and Controls. Silvia has been a great technical advisor to me by allowing me to choose my own area of research while simultaneously monitoring my work closely such that I was able to finish my degree in a short amount of time. She furthermore allowed me to explore my research interests as well as nurtured my professional career development that has ultimately led to me having an innovative position in industry secured by the final few months of my PhD. I would also like to acknowledge the teaching team for the Commercialization fellowship at Cornell since this program helped to steer my research direction to always focus on solving technical problems that also provide value to an end-user of my research developments.

Furthermore, I would like to acknowledge the Office of Naval Research for providing the grants that partially funded my PhD.

Finally, I would like to acknowledge and thank my family, including my wife Kathryn, and my parents, Mark and Julie, and my dog, Apollo. I would not have been able to be where I am now without their unwavering support throughout my PhD and my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Active vision for autonomous systems involves integrating several areas of research including computer vision and machine learning, as well as dynamical systems, control theory, and information-driven path planning [33]. In particular, an autonomous system with active vision capabilities is equipped with one or more vision sensors, and possibly many other sensors, that are used to perceive its surrounding environment and, based on those perceptions, react in some way. A reaction may include modifying the camera's field-of-view (FOV) such that the amount of valuable perception information is maximized, for example, when tracking a moving target. Examples of active perception tasks that are investigated in this dissertation include object recognition and tracking, scene depth estimation, and human action recognition.

The term vision sensor in this dissertation refers, primarily, to directional sensors that have a bounded FOV and are subject to line-of-sight (LOS) visibility; That is, the sensor cannot obtain measurements through opaque objects that cause occlusions. The most well-studied vision sensor is a digital camera that measures red-green-blue (RGB) color light intensity at each pixel in the sensor array, referred to as an RGB camera. An RGB camera is exactly the type of camera used on mobile devices, web-cameras, closed-circuit TV (CCTV) security cameras, and many more daily real-world applications. RGB cameras have recently been shown to be capable of providing incredibly valuable information that can be processed to produce high-level autonomous perception results for tasks such as scene depth

estimation, object recognition and tracking, and human action recognition. However, to employ many of these algorithms, especially those based on deep learning convolutional neural networks (CNN), an incredible amount of processing power is required. For this reason, this dissertation focuses on two types of vision sensors for active vision: (i) RGB cameras, and (ii) event-cameras, which are discussed in further detail later.

Active perception refers to the autonomous capability of a robotic system to perceive its surroundings using onboard sensors and react or make decisions based on those perceptions. With significant strides in computer vision capabilities recently coupled with well-developed theory of control systems and robotics, active perception is at the frontier of robotics research. Recent state of the art active perception autonomous systems are equipped with one or often multiple standard RGB cameras that record sequential images, or frames, based on light detecting sensors such as a charge-coupled device (CCD) or active pixel complementary metal-oxide semiconductor (CMOS) sensors, at a fixed frequency, or frame rate. Examples of such systems are autonomous quadrotors capable of navigating an environment to identify targets [40] or track and follow a human of interest [38, 39], and even understand and react to the actions or gestures that the human gives to the quadrotor [37]. However, even with these impressive capabilities, these autonomous systems are significantly less capable than even simple biological active perception systems, such as birds and insects. This restriction often requires multiple processors (CPU) and graphical processors (GPU) that consume incredible amounts of energy and are not suitable for small or even most medium-sized robotic platforms, such as quadrotors and micro-aerial vehicles (MAVs) due to payload and energy constraints. Part of this shortcoming is due to the computational cost of today's active perception algorithms as well as the temporally-discrete underlying

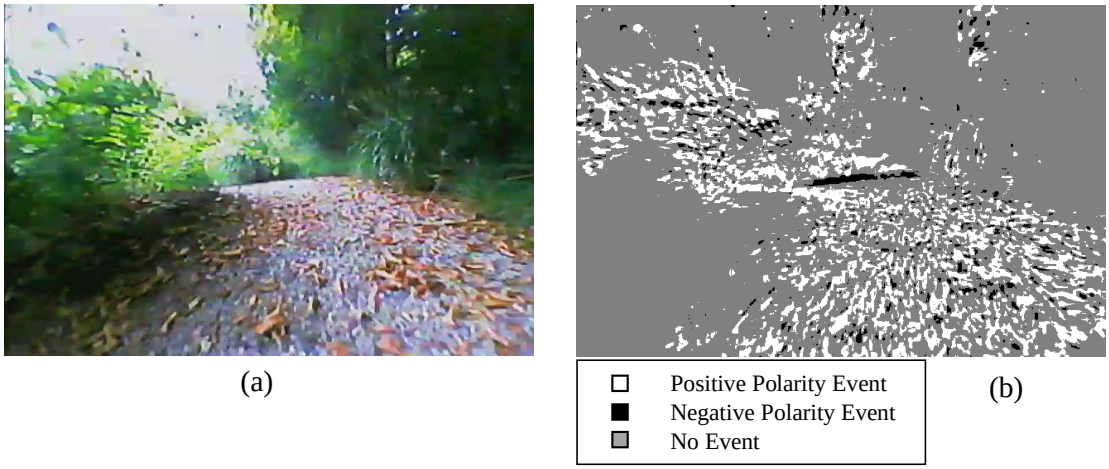


Figure 1.1: RGB camera (a) and event camera (b) measurement signals for a camera onboard a moving quadrotor. The standard camera measuremnt signal results in a sequence of RGB images and the event camera signal results in a collection of events with positive (white) and negative (black) polarities

framework of CCD and CMOS video cameras. Throughout this dissertation, we implement a standard CMOS vision sensor and refer to this sensor as the RGB camera, for the red-green-blue (RGB) data representation of every pixel in each frame, and this class of cameras applies to all traditional digital RGB camera methodologies that the algorithms discussed in this dissertation can be applied to.

This dissertation proposes the use of event cameras, also known as neuromorphic sensors, to alleviate the aforementioned shortcomings of RGB cameras for active perception. Event cameras are biologically inspired sensors that are fundamentally different than RGB cameras, in that they asynchronously measure per-pixel brightness changes over time with micro-second temporal resolution [87]. Event cameras themselves are new hardware developments that have only recently had perception algorithms developed for simple tasks such as collision avoidance and motion detection for fast-moving objects [31, 32]. This article provides the novel development and application of several active perception techniques for event

cameras, and furthermore compares each of these methods to the analogous RGB camera algorithm, in terms of both accuracy and computational cost. This significant research contribution not only provides several novel active perception algorithms for event cameras, but will also allow for clear and concise decisions to be made for future active perception research in terms of the tradeoff between event camera benefits and limitations when compared to RGB camera capabilities.

In addition to active vision, which encompasses visual perception and control, this dissertations furthermore develops path planning techniques that applies to all directional sensors. More precisely, presents a novel and systematic methodology for planning the trajectory of a mobile robotic sensor deployed to classify multiple fixed targets located in an obstacle-populated workspace. Existing robotic sensor path planning methods are not directly applicable to robots whose primary objective is to gather sensor measurements using a sensor with a bounded field of view and subject to line of sight visibility. The method developed in this dissertation is a novel path planning development in which the obstacles, sensor field of view, and robotic platform, are represented as closed and bounded subsets of an Euclidean workspace. The proposed method computes a collection of subsets of the directional sensor's configuration space, referred to as visibility regions, that quantify the visibility of every subset of targets subject to the sensor field of view geometry and line of sigh visibility. The visibility regions are then used to construct a connectivity graph that represents the connectivity of the sensor's configuration space. The computational performance of the algorithm is analyzed and an approximation algorithm with significantly improved efficiency is also presented which leverages visibility regions in which multiple targets are visible. The effectiveness of the sensing strategies computed by this method are demonstrated through multiple simulation and physical experiments, in which the method devel-

oped in this dissertation is compared to traditional Traveling Salesman Problem approaches, as well as cell decomposition, and coverage methods. The proposed algorithm outperforms the comparison algorithms in both physical and simulated experiments.

1.2 Literature Review

A significant amount of work has become recently available in the literature with respect to event-based perception. Event cameras are a biologically inspired sensor, and it has been shown that several small insects and other biological systems perceive their environment based on changes in their FOV rather than raw intensity [36]. Biologically inspired collision avoidance is a subject of significant study, in particular. Many flying animals use event-like sensing to navigate at incredibly high speeds in very cluttered environments, in such a manner that is impossible for RGB cameras, but has been shown to be achievable for event cameras [31, 32]. Other collision avoidance studies show how nocturnal flying insects use alternative sensing modalities outside of vision to navigate in the dark, when vision is not an option [73]. Some basic studies have been performed showing the control of a quadrotor using event cameras [28], but few of these studies look at high-level tasks such as object recognition, action recognition, and human interaction with event-driven perception algorithms onboard active robotic systems. The efficiency of event cameras has been widely speculated in the literature and the importance of temporal information has also been explored [50, 60] leading to the hypothesis in this dissertation. Based on the current available studies, event cameras are expected to be highly more computationally efficient than RGB cameras and capable of performing many navigational tasks and potentially even high-level active per-

ception tasks at a similar performance to RGB cameras. This dissertation explores both the computational cost and accuracy between event and RGB cameras in a rigorous methodology making use of both perfectly repeatable and controllable simulations and physical experiments on multiple vehicle platforms.

Recent advancements in computer vision, particularly video-based object detection and classification, pave the way for future autonomous systems comprised of camera-equipped mobile robots that can decide how to obtain and process images or videos without human intervention [98]. The development of autonomous mobile cameras have recently been shown to impact a variety of applications that include automated surveillance, [4], intelligent cinematography [72], and autonomous social navigation [19]. The challenge of detecting, tracking, and following a mobile human target of interest is critical to all of the aforementioned applications. As a result, several human tracking algorithms have been developed, some of which make use of carefully designed hand-crafted features, such as Histograms of Oriented Gradients (HoG) for detection with simultaneous KLT (Kanade-Lucas-Tomasi) feature-tracking [4], and optical flow-based human tracking methods using multiple cameras [94]. More recent human tracking algorithms take advantage of the advancements of deep convolutional networks, and instead use convolutional features for tracking human appearance in videos [71]. Human tracking in video has become one of the challenging problems at the forefront of computer vision, leading to the development of benchmark datasets [56]. The state-of-the-art methods according to such benchmarks base their hypotheses on multiple cues, such as appearance, kinematics, and interactions [83].

Although the problem of human detection and tracking has been studied extensively over recent years, almost all of the tracking algorithms do not incorporate

any type of control over the camera FOV, and instead assume the video sequence is recorded *a priori*. The few works that do actively control the camera FOV simplify the detection and tracking of the human target [42], [72]. This dissertation considers the problem of controlling a mobile camera with a bounded FOV that is rigidly attached to a mobile robot capable of onboard computing for real-time video processing. The camera is controlled such that a target (human) of interest is detected, tracked, and followed while moving through a complex, unstructured environment. The novel approach in this disseration leverages a state-of-the-art deep learning algorithm for detecting a human target in the video ([61], whose output is processed to compute a control input command for the mobile robot without requiring a 3-dimensional position estimate or kinematic model of the target. This is advantageous since human motion is generally very difficult, if not impossible, to accurately predict.

Another active perception task investigated in this dissertation is that of Human pose estimation (HPE) and human action recognition (HAR), which are are two closely related tasks for computer vision and robotics applications, and until very recently, have been treated separately in the literature [21,66]. HPE refers to the task of estimating the positions of human joints (e.g., wrist, elbow, etc.) in the image plane (2D HPE, [16,51]), or in the 3D scene that the camera is embedded in (3D HPE, [18,66]), given a single image or video sequence. HAR refers to the task of localizing and classifying actions performed by a particular human of interest (e.g., idling, walking, running, bike riding, etc.) [59]. HPE and HAR are closely related since both tasks are influenced by the relative positions and trajectories of the human joints. However, only a small number of recent works have addressed these problems simultaneously [58,66,77]. Unlike previous HAR algorithms, the method developed in this dissertation produces a 3D pose representation for action features

that are, by definition, independent of the camera viewpoint, or *viewpoint invariant*. Viewpoint invariant HAR for real-time autonomous systems is motivated by myriad applications such as autonomous driving [44], video surveillance [82], person tracking and following [39], and virtual/augmented reality [67], for example. Invariant features have been developed for computer vision tasks for several decades in order to create robust perception systems that are reliable in unconstrained video sequences. Scale invariant feature transforms (SIFT), histograms of oriented gradients (HOG), are a few examples of invariant features for early computer vision tasks, such as image classification and object recognition [25]. Other related works attempt to generate viewpoint invariant features for action recognition using ad-hoc and purely data driven approaches [58]. The method proposed in this dissertation presents a systematic approach to viewpoint invariance by assuming known body proportions of the target human and a rotation-based representation of human pose that is derived analytically and is inherently independent of the camera viewpoint. Several authors have concluded that although 2D HPE in the image plane performs very well, 3D HPE has proven to be challenging and remains an open problem, particularly when deep end-to-end frameworks attempt to learn depth information directly from raw pixel data [18, 69]. Unlike purely data-driven deep learning methods, the proposed method leverages kinematics and geometry in order to accurately estimate the human's 3D pose without the need for massive amounts of 3D pose data, which is extremely difficult to obtain without expensive equipment, and is still less precise than 2D pose data. Additionally, HAR has been widely studied in the literature, but many of the approaches are only applicable in the case of multiple sensors [58], or the performance degrades with camera motion [8].

The second portion of the dissertation relates to occlusion avoidance in path

planning for directional sensors. Directional sensor planning refers to the problem of developing a sensing strategy for a mobile robot tasked with perceiving multiple targets that are distributed in a workspace. A directional sensor is characterized by a bounded field of view (FOV) whose measurements are subject to line of sight (LOS) visibility, or occlusions caused by opaque obstacles in the workspace. Recent work in the area of robot and sensor planning has focused on motion-planning [20, 54, 55, 85, 93], trajectory optimization [5, 34, 46, 84], treasure hunt [14, 63, 92, 100, 101], and next-best view planning [27, 76, 89]. However, robot motion-planning and trajectory optimization methods do not typically focus on the sensor performance, and instead are concerned with collision avoidance and planning acceptable robot paths regardless of the sensing objective. In addition, while treasure hunt and next-best view planning methods emphasize a sensing objective, LOS visibility is often ignored in these formulations. This dissertation presents a novel method that augments these developed research areas to include a planning methodology for the general and widely applicable directional sensor planning problem. Because many of the sensors being used today for active perception, such as monocular cameras [39, 98], stereo cameras [23, 35], radar [91], sonar [105], neuromorphic cameras [15, 104], all fall within the definition of directional sensors, the methods presented in this dissertation are widely applicable and valuable to the field of active perception. Some applications that will benefit from the new directional sensor planning methods presented in this dissertation include autonomous vehicle safety [74], digital agriculture [45], person/target tracking and following [39, 99], mine counter measure planning [105], and fugitive gas leak detection and estimation [1, 41].

A similar class of path planning problems is the Traveling Salesman Problem (TSP) [2] which refers to the problem of determining the minimum distance

path that visits multiple point targets and returns to the starting point. However, the original TSP formulation cannot account for the sensor FOV geometry and may produce highly undesirable solutions and possibly fail in the presence of large FOV geometries and obstacles in the workspace. The formulation of the TSP has been extended to include non-overlapping Euclidean neighborhoods in the augmented problem referred to as the Traveling Salesman Problem with Neighborhoods (TSPN) [29]. The TSPN refers to the problem of finding the minimum-length path that passes through at least one point of each continuous Euclidean target neighborhood and returns to the original starting point. The TSPN, however, is only applicable to a special case of the directional sensor planning problem in which a maximum of only one target is visible from any sensor configuration. Furthermore, the TSP formulation has been extended to the Group Traveling Salesman Problem (GTSP) [29], in which the path is required to only visit at least one target of each target group, which contains a finite set of targets. The GTSP and TSPN have also been combined in the literature, to produce the Group Traveling Salesman Problem with Neighborhoods (GTSPN) [29]. The proposed approach for directional sensor path planning is formulated more generally in this dissertation than any variation of the TSP problems, and thus each of the TSP variations could be derived as a special case of the method proposed in this dissertation. Additionally, the TSP literature is focused on the optimization step rather than the construction of the geometry from a practical sensing challenge. In contrast, this dissertation formulates a general optimization framework that stems from a common problem in directional sensor planning.

The path planning method in this dissertation focuses on planning in the presence of occlusions induced by opaque obstacles in the workspace. Similar works in the literature that are relevant for visibility planning include coverage-based

planning algorithms with visibility constraints, such as the Art Gallery Problem (AGP) [96] and the Watchman's Route Problem (WRP) [43]. The AGP refers to the problem of determining the minimum number of stationary and omnidirectional sensors required to completely cover a polygonal workspace with no obstacles. Although the AGP is one of the few problem formulations that includes the geometrical notion of occlusions, the solution approaches are not applicable to sensors with a bounded FOV or sensors that are onboard a mobile robot. The WRP refers to the problem of determining the optimal closed path of a mobile omnidirectional sensor tasked with completely covering a workspace that may be populated with obstacles. However, solutions to the WRP are also not applicable to bounded sensor FOVs. Additionally, both the AGP and WRP are coverage problems, while the directional sensor planning problem in this work only requires the sensor can view a finite set of target positions.

1.3 Overview of Research Contributions

This dissertation describes new methods that enable event cameras to perform the following active perception tasks: (i) Depth Estimation, (ii) Object Recognition, (iii) Object Tracking, and (iv) Action Recognition. Although this dissertation presents these tasks as four separate problem formulations, many autonomous systems requiring active perception capabilities rely on some or all of these capabilities simultaneously to accomplish a high-level objective, such as autonomous driving, or human robot interaction. For example, a robot may be tasked with maintaining a person in its field of view and must follow the person as they walk an unknown path, while simultaneously analyzing the human's actions such that the robot can obey the human's commands all while using depth estimation for collision avoid-

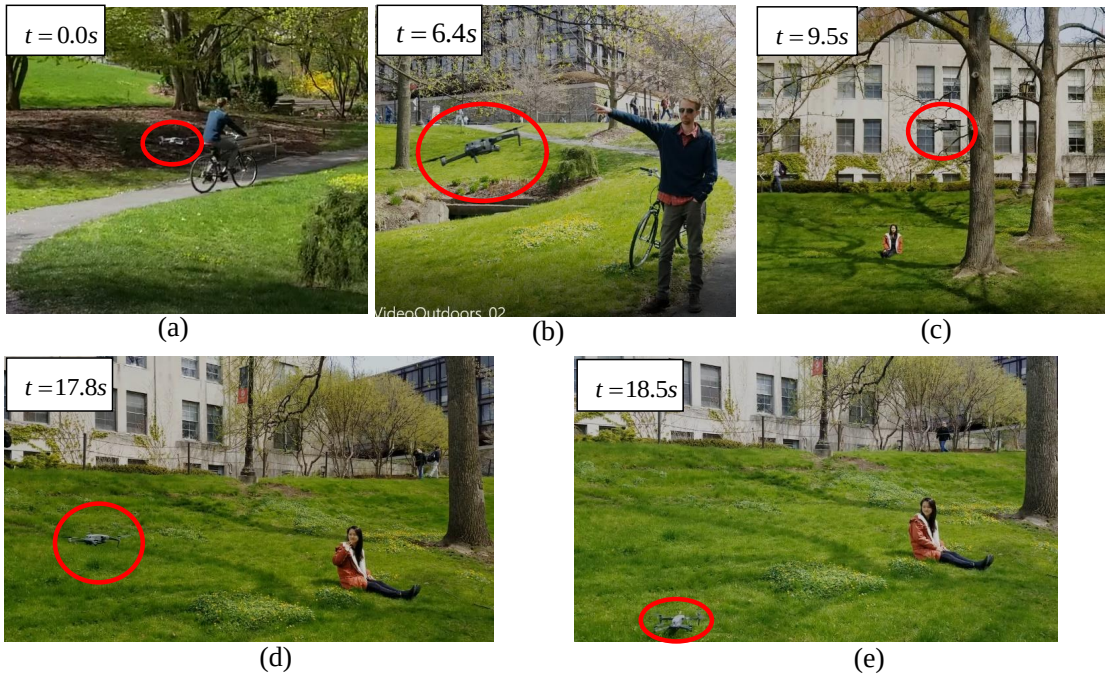


Figure 1.2: A demonstration of the event-based active perception algorithms performed on an autonomous quadrotor. The quadrotor is initially tracking and following a human on a bicycle (a), and upon perceiving the action of the human pointing (b), the quadrotor quickly flies in the prescribed direction (c). The quadrotor continues autonomously until in that direction over complex terrain that many vehicles cannot traverse until another human takes control of the quadrotor by waving in the sensor's field of view (d), and commanding the quadrotor to land safely (e).

ance and perceiving the distance to the person of interest. Multiple of these types of demonstrations are presented in this dissertation, both in physical experiments and controlled simulations. Figure 1.2 demonstrates such an experiment that highlights the capabilities of the active perception methods developed for event cameras in this dissertation. In addition, the raw sensor data for an RGB camera and event camera are show in correspondence in Figure 1.3 and Figure 1.4, respectively.

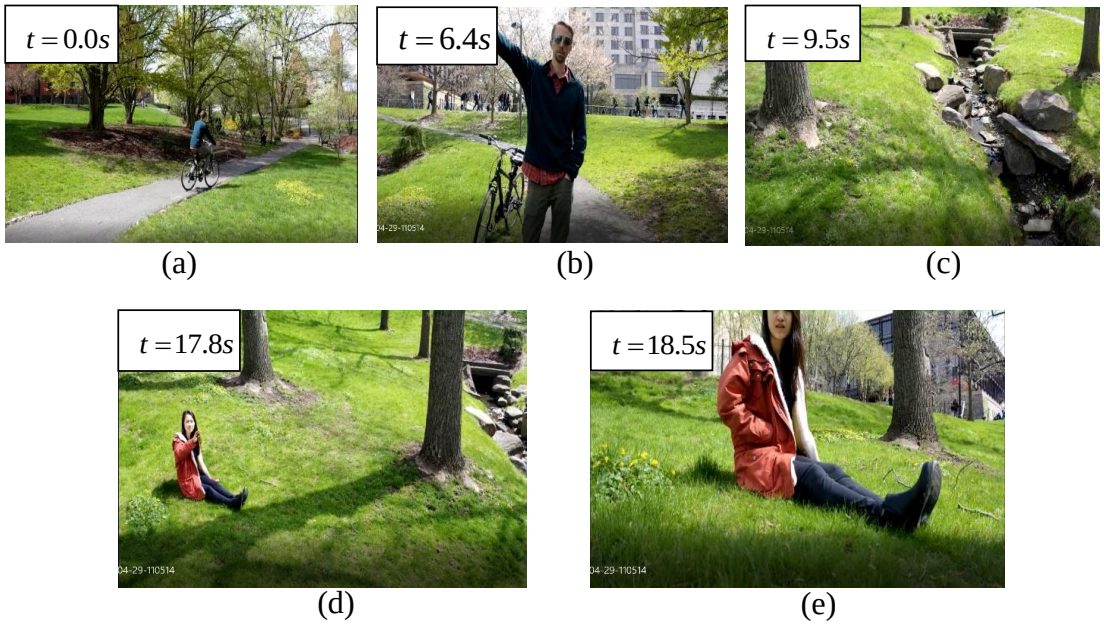


Figure 1.3: RGB camera FOV images corresponding to the active perception experiments on an autonomous quadrotor

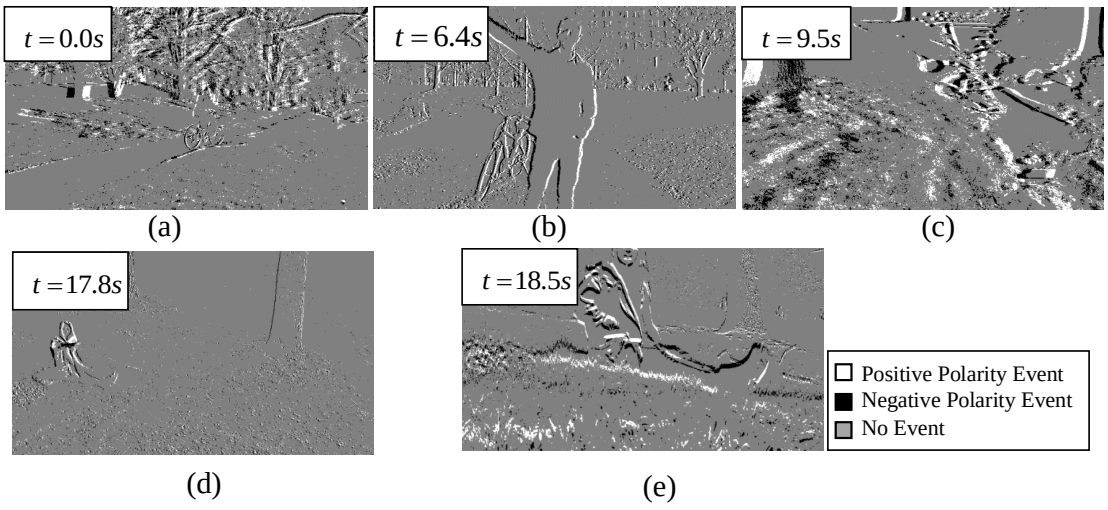


Figure 1.4: Event camera output corresponding to the active perception experiments on an autonomous quadrotor.

1.3.1 List of Research Contributions

This dissertation presents novel contributions on several fronts of active perception for robotics. To summarize these findings, a list of research developments is presented here and each of the contributions is discussed in detail throughout the remainder of the dissertation.

- Event-camera optical flow algorithm development and comparison to an RGB camera optical flow algorithm
- Event-camera depth estimation algorithm development and comparison to an RGB camera depth estimation algorithm
- Event-camera object recognition algorithm development and comparison to an RGB camera object recognition method
- Event-camera object tracking algorithm development and comparison to an RGB camera object tracking algorithm
- Event-camera human action recognition algorithm development and comparison to an RGB camera human action recognition algorithm
- Real-time 3D human pose estimation based on perspective geometry
- Person following Controller design for nonholonomic ground vehicles
- Person following controller design for quadrotors
- Human-directed flight control from pointing recognition
- Occlusion avoidance in directional sensor path planning

1.4 Thesis Organization

This thesis is divided into two primary components: Active vision, and occlusion avoidance. The first part on active vision is presented in Chapter 2, which encompasses new active vision research and algorithms for RGB cameras and event cameras. The chapter is broken down by the active vision tasks that include optical flow, depth estimation, object recognition, object tracking and following, and action recognition. Each of these tasks have a new event-camera algorithm proposed and many of them also have new developments for RGB cameras. Physical experiments are presented at the end of the chapter along with quantitative performance results that showcase the algorithm accuracy as well as the computational cost.

Chapter 3 provides detailed theoretical work on the development of occlusion avoiding path planning for directional sensors. A closed form representation of a directional sensor's configuration space is developed and a highly accurate path planning method is developed. Furthermore, alternative approximation path planning algorithms are developed based on the theoretical developments of the visibility regions in configuration space. The various methods are qualitatively and quantitatively compared amongst themselves as well as approaches inspired by existing algorithms such as the travelling salesman problem (TSP) and coverage algorithm. The dissertation then ends with a concluding chapter that summarizes the findings and provides recommendations for future work.

CHAPTER 2

ACTIVE VISION

2.1 Visual Perception and Control Design Interface

Active perception encompasses both visual perception and active control by coupling the perceived environment with a desired autonomous vehicle behavior that is achieved through controller design. The developments in this thesis are exclusively designed with active perception tasks in mind such that each perception algorithm is designed to be achievable on a real-time embedded system whose behavior in an environment is driven by the coupled perception and control methodologies. Figure 2.1 illustrates the integration of the perception algorithms with control and planning algorithms. A desired behavior is input to an active vision system and using the resulting perceived scene from the perception algorithms a high-level control behavior is computed, such as to 'take-off', 'land', or 'track and follow an object of interest'. The high-level controller outputs this desired behavior into a classical controller that is capable of achieving the desired behavior. Then, the control input is provided to the autonomous vehicle dynamical system which in turn adjusts the camera state and sensor FOV. By adjusting the sensor FOV and vehicle state, the sensor perception algorithm is altered by physically changing the position an possibly the geometry of the sensor FOV to improve the sensor's scene awareness and understanding. An example of such a system is shown in Figure 2.2 in which a micro-aerial vehicle (MAV) performs high-level perception tasks including action and object recognition which is input to a high-level controller that determines the vehicle behavior to be 'object follow mode'. The low-level controller then provides the input signals to the MAV, which follows the person-of-interest.

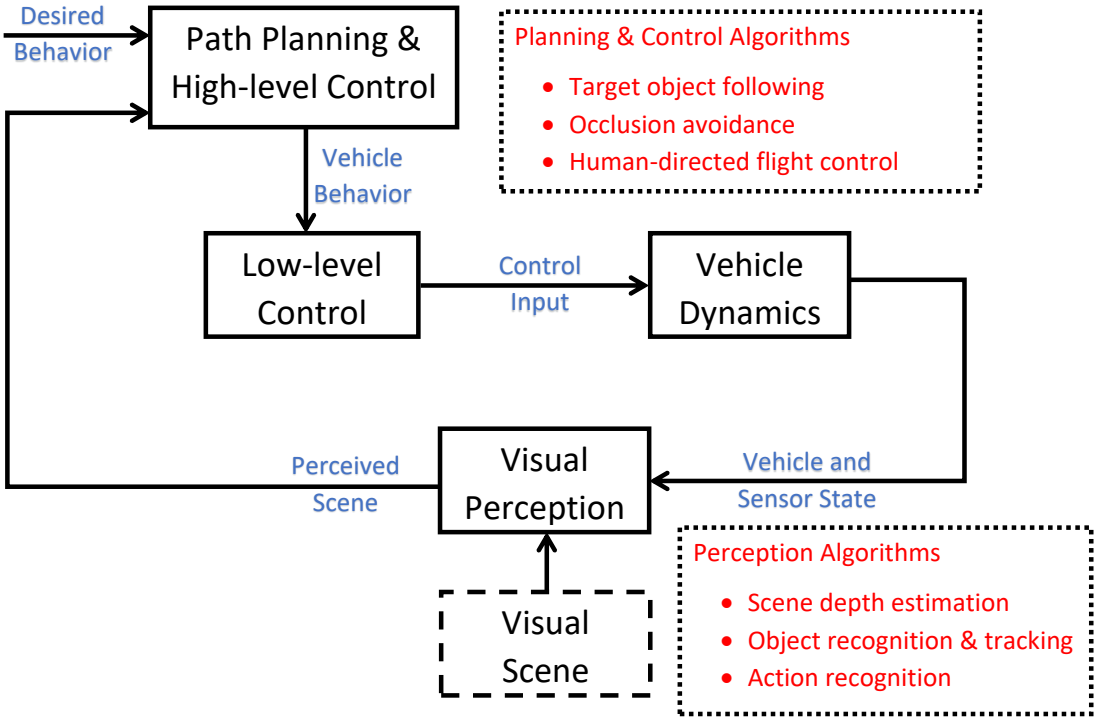
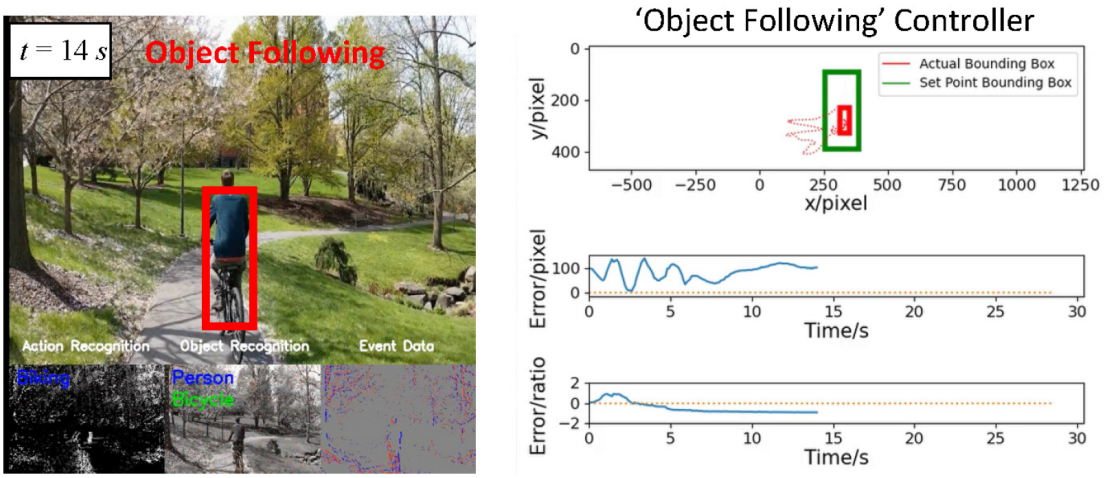


Figure 2.1: Active perception block diagram.



(a) (b)

Figure 2.2: Integrated MAV perception and control. High-level perception including action and object recognition (a) is put into action by a switched controller shown in 'object following mode' (b) that reacts to the actions of the person-of-interest identified earlier as the 'human' 'waving' who is presently biking.

2.2 Event-camera and RGB Camera Physics

Cameras are visual directional sensors with a bounded field of view, denoted by $\mathcal{S} \subset \mathbb{R}^3$ (FOV) defined as a subset of the robot workspace $\mathcal{W} \subset \mathbb{R}^3$ [1]. Opaque objects in $\mathcal{S}$ are projected onto the sensor image plane $\mathcal{I} \subset \mathcal{S}$, defined as a 2D plane that is orthogonal to the sensor's optical axis and is located at a distance from the sensor's focal point, known as the focal length. For an RGB camera, the signal recorded from the image plane is a discrete synchronous sequence of 3-channel 8-bit integer $(h \times w)$ image arrays, or frames, each denoted by the matrix,

$$I(t_k) = \begin{bmatrix} L_{11}(t_k) & L_{12}(t_k) & \cdots & L_{1w}(t_k) \\ L_{21}(t_k) & L_{22}(t_k) & \cdots & L_{2w}(t_k) \\ \vdots & \vdots & \ddots & \vdots \\ L_{h1}(t_k) & L_{h2}(t_k) & \cdots & L_{hw}(t_k) \end{bmatrix} \tag{2.1}$$

recorded at time $t_k = t_0 + k\Delta t$, where each element of the RGB camera frame is a 3-channel 8-bit light intensity value for the red, green, and blue color components, $L_{ij}(t_k) = [r(t_k) \quad g(t_k) \quad b(t_k)]^T$, and $r(t_k), g(t_k), b(t_k) \in \{0, 1, ..., 255\}$ candelas. The RGB camera frames are sampled at a constant frequency, known as the frame rate, measured in frames-per-second (fps). The time interval between sampled frames, also known as the inverse of the frame rate, is denoted by $\Delta t > 0$. An event camera, sometimes referred to as a neuromorphic camera or dynamic vision sensor, is an imaging sensor that records events characterized by significant local changes in brightness. Each pixel in the event camera sensor array operates independently and asynchronously from the others, storing a reference intensity level at time $t \in [t_0, t_f]$, denoted by $\tilde{L}_{ij}(t)$, such that the internal reference intensities recorded

by the full sensor array can be represented by the matrix,

$$E(t) = \begin{bmatrix} \tilde{L}_{11}(t) & \tilde{L}_{11}(t) & \cdots & \tilde{L}_{11}(t) \\ \tilde{L}_{11}(t) & \tilde{L}_{11}(t) & \cdots & \tilde{L}_{11}(t) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{L}_{11}(t) & \tilde{L}_{11}(t) & \cdots & \tilde{L}_{11}(t) \end{bmatrix} \tag{2.2}$$

At time t, where the subscripts $(\cdot)_{ij}$ denote the pixel coordinate (i, j). Over time, the reference intensity level $\tilde{L}_{ij}(t)$ is continuously compared to the intensity level observed at the pixel coordinate denoted by $L_{ij}(t)$ and an event is fired at (i, j) if and only if the change in the log intensity exceeds a user-defined threshold $\theta > 0$, i.e.,

$$\| \ln L_{ij}(t) - \ln \tilde{L}_{ij}(t) \| \geq \theta \tag{2.3}$$

Then, the event polarity is defined as,

$$p_{ij} = \begin{cases} 1 & \text{if } [\ln L_{ij}(t) - \ln \tilde{L}_{ij}(t)] \geq \theta \\ -1 & \text{if } [\ln L_{ij}(t) - \ln \tilde{L}_{ij}(t)] < \theta \end{cases} \tag{2.4}$$

As the event-camera moves in $\mathcal{W}$ and operates over a period of time $[t_0, t_f]$ a stream of asynchronous events is recorded and organized into the set of events,

$$\mathcal{E} = \{\mathbf{e}(t_1), \mathbf{e}(t_2), ...\} \tag{2.5}$$

where,

$$\mathbf{e}(t_k) \triangleq [i \quad j \quad t_k \quad p_k]^T, \quad i \in \{0, ..., w\}, \quad j \in \{0, ..., h\}, \quad t_k \in [t_0, t_f], \quad p_k \in \{-1, 1\} \tag{2.6}$$

Similarly, as the RGB camera moves in $\mathcal{W}$ and operates over the same time period $[t_0, t_f]$ a stream of synchronous RGB frames is recorded and organized in the set array,

$$\mathcal{V} = \{I(t_0), I(t_0 + \Delta t), I(t_0 + 2\Delta t), ...\} \tag{2.7}$$

A fundamental and important difference between the stream of events $\mathcal{E}$ and the video stream $\mathcal{V}$ is that each frame $I(t_0 + k\Delta t) \in \mathcal{V}$ is collected synchronously with a frequency that depends on the video frame rate, while each event $\mathbf{e}(t_k) \in \mathcal{E}$ fires asynchronously at time $t_k \in [t_0, t_f]$. The following sections describe the new methods for active perception with event cameras with analogous high-level capabilities as standard RGB cameras.

2.3 Optical Flow

The event-camera depth estimation method takes advantage of a new event-camera optical flow algorithm that uses the sensor firing rate to capture the spatial-temporal image gradient information with respect to events in the image plane. The computation of the firing rate is discussed in the Materials and Methods section. The event-camera optical flow is computed from the velocity vector of every pixel in the image plane. In the case where there is no event at a given pixel, the optical flow there is zero since the lack of events firing is indicative of a lack of motion. For stationary objects and a static environment, the optical flow vector of a pixel at $(i, j) \in \mathcal{I}$ is defined as,

$$\dot{\mathbf{p}}_{ij}(t) = \frac{1}{z_{ij}(t)}\mathbf{G}_{ij}\mathbf{v}(t) + \mathbf{H}_{ij}\boldsymbol{\omega}(t) \tag{2.8}$$

where $z_{ij}(t)$ is the depth at pixel coordinate (i, j) at time t, and $\mathbf{G}_{ij}, \mathbf{H}_{ij} \in \mathbb{R}^{2\times 3}$ are matrices that depend on (i, j) and on constant physical camera parameters that are discussed in further detail in the Materials and Methods section. The optical flow is a vector field comprised of all optical flow vectors obtained from the pixel

in $\mathcal{I}$, and organized in the matrix,

$$\mathbf{F}(t) = \begin{bmatrix} \dot{\mathbf{p}}_{11}(t) & \dot{\mathbf{p}}_{11}(t) & \cdots & \dot{\mathbf{p}}_{11}(t) \\ \dot{\mathbf{p}}_{11}(t) & \dot{\mathbf{p}}_{11}(t) & \cdots & \dot{\mathbf{p}}_{11}(t) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{\mathbf{p}}_{11}(t) & \dot{\mathbf{p}}_{11}(t) & \cdots & \dot{\mathbf{p}}_{11}(t) \end{bmatrix} \tag{2.9}$$

The event-camera optical flow algorithm relies on a quantity known as the event firing rate, denoted by $f(\xi, \eta, t)$, where $\xi \in \mathbb{R}$ and $\eta \in \mathbb{R}$ denote the horizontal and vertical spatial position on the image plane, respectively. The positions ξ and η are related to the pixel coordinates i and j through the physical camera parameter known as pixels per inch (PPI). The event firing rate is a smooth and differentiable function over a spatial-temporal volume that represents the event density in that volume. The event firing rate is used to construct the event-camera optical flow equation,

$$\frac{\partial f(\xi, \eta, t)}{\partial t} = -\frac{\partial f(\xi, \eta, t)}{\partial \xi}\dot{p}_{\xi}(t) - \frac{\partial f(\xi, \eta, t)}{\partial \eta}\dot{p}_{\eta}(t) \tag{2.10}$$

where the horizontal and vertical optical flow vector components at the spatial position (ξ, η) are denoted by $\dot{p}_{\xi}(t)$ and $\dot{p}_{\eta}(t)$, respectively, and together with the camera PPI can be used to compute the optical flow vector $\dot{\mathbf{p}}_{ij}(t) = [\dot{p}_i(t) \quad \dot{p}_j(t)]^T$ at a pixel coordinate (i, j). The algorithm then computes the optical flow vector for each pixel coordinate (i, j) that have a corresponding position (ξ_i, η_j), using an optimization derived from the event-camera optical flow equation. Because the event-camera optical flow equation is a scalar with two unknown optical flow variables, the additional assumption is made that the spatial gradient of the firing rate remains constant along the trajectory of moving points through the image plane. Then, the event-camera optical flow optimization problem is given as,

$$\mathbf{F}(t) = \arg\min_{\mathbf{F}(t)} \left(\sum_{i=0}^{w} \sum_{j=0}^{h} \left(\Delta\psi_{ij}^2(t) + \Delta f_{ij}^2(t) \right) \right) \tag{2.11}$$

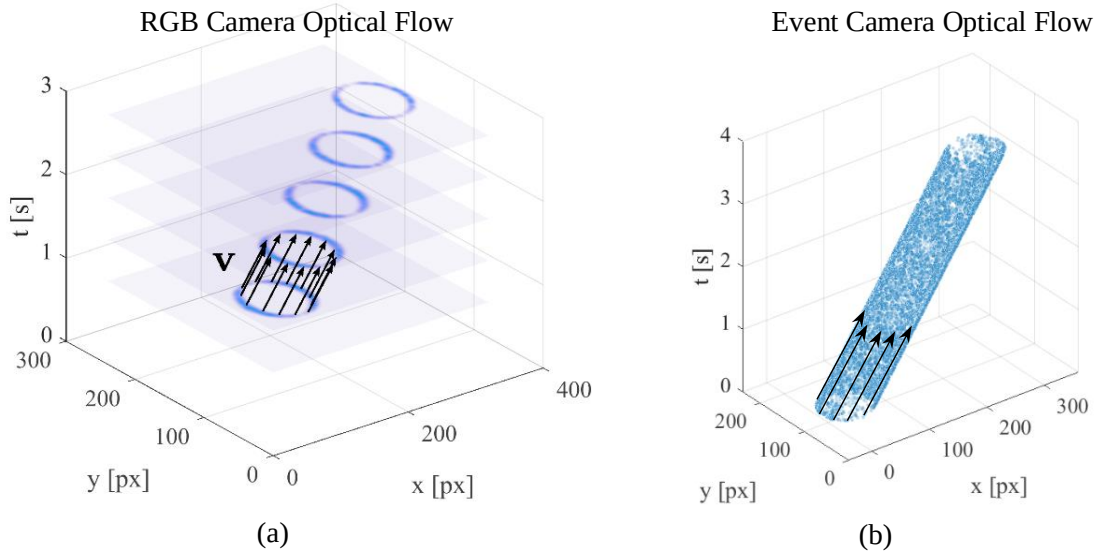


Figure 2.3: Optical flow comparison for a translating circle between RGB frame algorithm and event-camera algorithm

where

$$\Delta\psi_{ij}(t) \triangleq \psi(\xi_i + \dot{p}_i(t)\Delta\tau, \eta_j + \dot{p}_j(t)\Delta\tau, t + \Delta\tau) - \psi(\xi_i, \eta_j, t) \tag{2.12}$$

$$\Delta f_{ij}(t) \triangleq f(\xi_i + \dot{p}_i(t)\Delta\tau, \eta_j + \dot{p}_j(t)\Delta\tau, t + \Delta\tau) - f(\xi_i, \eta_j, t) \tag{2.13}$$

where the orientation of the spatial gradient of the event firing rate is defined as,

$$\psi \triangleq \arctan\left(\frac{f_\eta}{f_\xi}\right) \tag{2.14}$$

and the shorthand notation $f_\xi = \partial f/\partial\xi$, $f_\eta = \partial f/\partial\eta$ is used for brevity. The optical flow function is updated every $\Delta\tau$ seconds, where $\Delta\tau$ depends on the available computing resources. The complete derivation and linearization of the event-camera optical flow algorithm is described in the Materials and Methods section.

2.4 Depth Estimation

The problem of depth estimation consists of determining the distance between the moving camera and opaque objects in the scene, based on their projection onto the image plane for every pixel location. All cameras measure the three-dimensional workspace by relying on perspective projection onto a two-dimensional image plane, resulting in the loss of depth information that cannot be reconstructed from the signals recorded at a single time instant. Although stereoscopic depth estimation algorithms exist for RGB and event cameras [3, 64, 65, 68], the methods presented this dissertation are developed to solve the more parsimonious problem of monocular depth estimation. It is assumed that knowledge of the robot velocity $\mathbf{v} \in \mathbb{R}^3$ and angular rate $\boldsymbol{\omega} \in \mathbb{R}^3$ is available either from GPS and IMU data or from an onboard localization and mapping algorithm [13]. Then the robot kinematics can be used to perform depth estimation based only on two subsequent RGB camera frames using structure from motion [95] or optical flow [106] algorithms. For comparison, in this dissertation an optical-flow depth estimation algorithm is implemented on the moving RGB camera in stationary environments. This RGB algorithm is chosen for comparison to the event camera because of its high computational efficiency. Taking inspiration from motion-based biological navigation [7, 64], a new event-camera depth estimation algorithm is developed in this dissertation. In particular, an event-camera optical flow algorithm is developed that is computationally efficient and enables real-time event-camera depth estimation.

From the event-camera optical flow, the depth $z_{ij}(t)$ for each pixel coordinate (i, j) is calculated using the matrix pseudo-inverse,

$$z_{ij}(t) = \left((\mathbf{v}^T \mathbf{G}_{ij}^T \mathbf{G}_{ij} \mathbf{v})^{-1} (\mathbf{v}^T(t) \mathbf{G}_{ij}^T)(\dot{\mathbf{p}}_{ij}(t) - \mathbf{H}_{ij} \boldsymbol{\omega}(t))\right)^{-1} \tag{2.15}$$

2.5 Object Recognition

Autonomous robots require reliable object-recognition capabilities. Object recognition consists of classifying objects in the scene by estimating a categorical variable with a range comprised of a countable and finite set of semantic labels, e.g.:

Performing object recognition requires first detecting relevant objects in the camera FOV, and then classifying their semantic label. Although many methods for object detection and recognition have been developed for RGB cameras to date, including convolutional neural networks (CNN) [80] and Histogram of Oriented Gradient (HOG) [25], to the authors' knowledge, there currently exist no object recognition algorithms for event-cameras. The event-camera object recognition algorithm presented in this dissertation is composed of two steps: object detection, and object classification. The object detection in this dissertation is done using the optical flow output and detecting individual objects in the image by clustering regions that have a large optical flow magnitude. Then, the image around the detected moving object is cropped and the optical flow in the cropped image is used to compute the HOF features that are used in the SVM for object classification, thereby completing the object recognition. The approach developed in this dissertation computes the same dense event-based optical flow field described in the previous subsection, and then computes the Histogram of Optical Flow (HOF) to extract a feature vector for image classification. HOF has been originally proposed in the literature for performing object and action recognition with RGB cameras [26], among other tasks [17]. The HOF features extracted from the event-camera image of a moving person are shown in Figure 2.4. Following the step of HOF feature extraction, a trained SVM is used to classify the object, as explained in the Materials and Methods section. However, object recognition

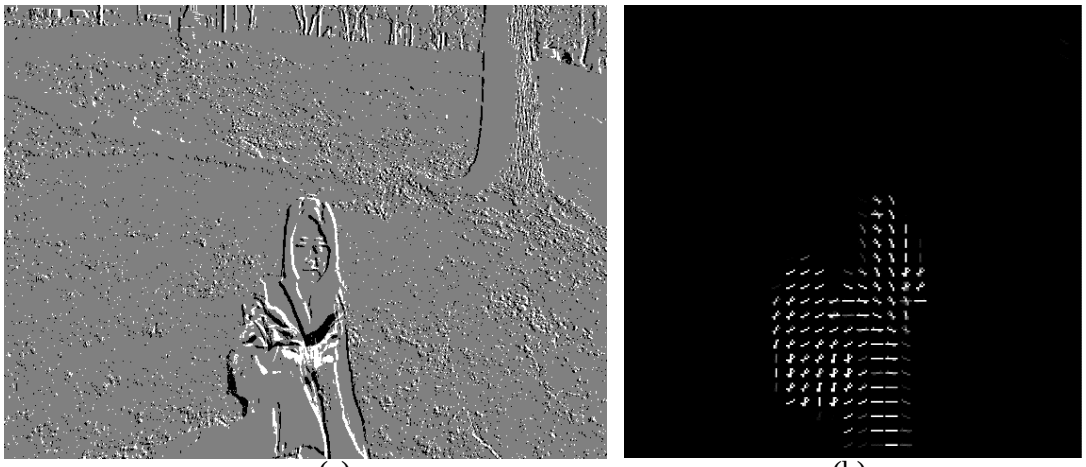


Figure 2.4: HOF features for object recognition. A moving person in the event-camera FOV (a) produces HOF features (b) that are used in a linear SVM for person recognition.

alone is often not enough for active perception, and a particular object must be tracked and even followed by a mobile robot over time for a monitoring task.

2.6 Object Tracking and Active Following

Having recognized objects in their surroundings, autonomous robotics may be required to track and follow an object of interest over time. The example used throughout this dissertataion is that of detecting, recognizing, and tracking a human in an environment populated with natural or man-built objects, vehicles, and animals. Once a human is recognized within the camera FOV, the robot is tasked with taking off and following the human within a specified distance. In order for the target to be consistently recognized over time, the target must first be detected and recognized within the sensor FOV.

2.6.1 Event-camera Object Tracking and Quadrotor Control for Active Target Following

The object detection algorithm for event-cameras is motion based, and the set of pixel coordinates where motion occurs at time , referred to as a motion mask, is computed from the optical flow as,

$$I_D(t) \triangleq \{(i,j) \in \mathcal{I} | \|\dot{\mathbf{p}}_{ij} > \mu\|\} \tag{2.16}$$

where $\mu > 0$ is a user-defined threshold that controls the detection sensitivity. Then, a density-based clustering algorithm [30] is employed to distinguish between multiple objects in $I_D(t)$. The output of the clustering algorithm provides a collection of N mutually disjoint moving object masks, denoted by $\mathcal{D}(t) = \{D_1(t), ..., D_N(t)\}$, such that the union of the object masks is a subset of the motion mask,

$$\bigcup_{i=1}^{N} D_i(t) \subseteq I_D(t), \quad \bigcap_{i=1}^{N} D_i(t) = \emptyset \tag{2.17}$$

Each object mask can then be classified using the event-camera object recognition algorithm and furthermore, the events associated with the detected human must be tracked and properly associated with the same person, in the presence of multiple objects appearing in the sensor FOV. A data association algorithm based on the Hungarian algorithm [6, 53] is used to solve the following optimization problem. Given a set of $N > 0$ detections at time t_1 and $M > 0$ detections at time t_2, find the Boolean matrix $\mathbf{X}$ whose elements $x_{ij} = 1$ if and only if the i-th detection at

t_1 is associated with the j-th detection at t_2, such that

$$\min_{\mathbf{X}} \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} x_{ij} \tag{2.18}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} x_{ij} \leq 1, \quad \forall j \tag{2.19}$$

$$\text{s.t.} \quad \sum_{j=1}^{M} x_{ij} \leq 1, \quad \forall i \tag{2.20}$$

where d_{ij} is the Euclidean distance between the i-th detection at time t_1 and the j-th detection at time t_2. While many tracking algorithms have been developed to date for RGB cameras, this dissertation presents the first event-based tracking algorithm. Let the target human's geometric center position be denoted by a vector $\mathbf{r}_T(t) \in \mathcal{W}$, and the event-camera position be denoted by a vector $\mathbf{r}(t) \in \mathcal{W}$ in inertial frame at time t, and let the constant specified distance be denoted by $\delta > 0$. Then, an onboard controller can be used to ensure that $\|\mathbf{r}_T(t) - \mathbf{r}(t)\| \to \delta$. The relative distance between the event-camera and the target is estimated using the depth estimation result for the pixel (i, j) that best aligns with the target detection geometric center, such that $\|\mathbf{r}_T(t) - \mathbf{r}(t)\| \approx z_{ij}(t)$. This approximation is most accurate when the sensor heading is such that the human is centered in the camera FOV. Therefore, two proportional-integral-derivative (PID) controllers are used to maintain the target at the specified distance and centered in the camera FOV,

$$u_1(t) = k_{p1}(z_{ij}(t) - \delta) + k_{i1} \int_{t_0}^{t} (z_{ij}(\tau) - \delta) d\tau + k_{d1} \frac{d}{dt}(z_{ij}(t) - \delta) \tag{2.21}$$

$$u_2(t) = k_{p2}(j(t) - \frac{w}{2}) + k_{i2} \int_{t_0}^{t} (j(\tau) - \frac{w}{2}) d\tau + k_{d2} \frac{d}{dt}(j(t) - \frac{w}{2}) \tag{2.22}$$

where the gains $k_{p1}, k_{p2}, k_{i1}, k_{i2}, k_{d1}, k_{d2} \geq 0$ control the weight of each term in the PID controllers, and $u_1(t)$ maintains the specified distance to the human and $u_2(t)$

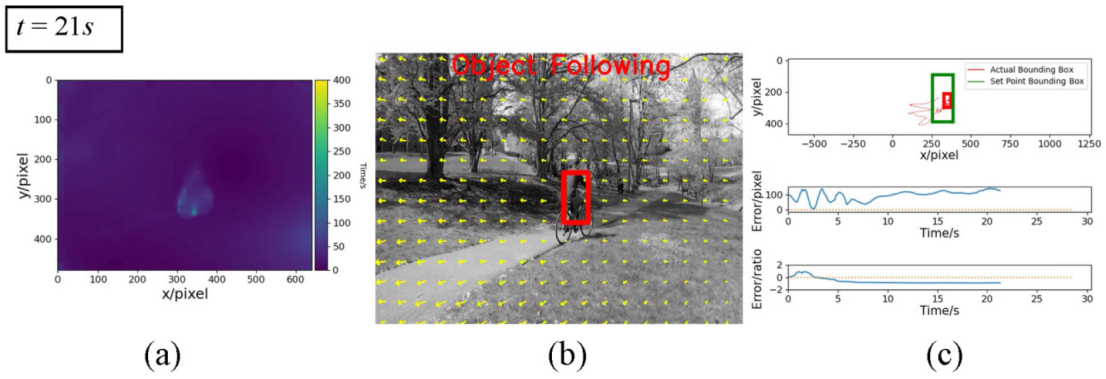


Figure 2.5: MAV active perception controller. The depth estimate is used for obstacle avoidance (a) while the camera feed and optical flow (yellow arrows) in (b) are used for object and action recognition, as well as for following the person-of-interest using the real-time onboard switched controller (c) shown in 'object following' mode at the present time.

maintains the human in the center of the FOV. An example of this system is shown in Figure 2.5.

2.6.2 Ground Vehicle Control for Active Target Following

This dissertation considers the problem of controlling a nonholonomic mobile ground robot equipped with an onboard camera characterized by a bounded field-of-view, tasked with detecting and following a potentially moving human target using onboard computing and video processing in real time. Computer vision algorithms have been recently shown highly effective at object detection and classification in images obtained by vision sensors. Existing methods typically assume a stationary camera and/or use pre-recorded image sequences that do not provide a causal relationship with future images. The control method developed in this dissertation seeks to improve the performance of the computer vision algorithms, by planning the robot/camera trajectory relative to the moving target based on the desired size and position of the target in the image plane, without the need to

estimate the target's range. The method is tested and validated using a highly realistic and interactive game programming environment, known as Unreal Engine™, that allows for closed-loop simulations of the robot-camera system. Results are further validated through physical experiments using a Clearpath™Jackal robot equipped with a camera which is capable of following a human target for long time periods. Both simulation and experimental results show that the proposed vision-based controller is capable of stabilizing the target object size and position in the image plane for extended periods of time.

This dissertation also proposes the use of a game development software, known as Unreal Engine™, for simulation of the vision-based control algorithm in a photo-realistic virtual space. Most robot simulation and visualization software emphasize physical accuracy and lack visual realism. Furthermore, simulation in a visually-realistic environment provides the ability to quickly and efficiently recreate and redefine an infinite set of testing conditions, while providing a deterministically repeatable test environment. This work further validates the proposed controller as well as the reliability of the Unreal Engine™as a control algorithm simulator through physical experiments which demonstrate the proposed methods are capable of real-world implementation.

Problem Formulation

Consider a region of interest, $\mathcal{W} \subset \mathbb{R}^3$, populated with human target $\mathcal{T} \subset \mathcal{W}$, and a mobile robot $\mathcal{A} \subset \mathcal{W}$. The robot is equipped with a camera which has a bounded FoV, characterized by a focal length $\lambda \in \mathbb{R}$, a half-angle $\alpha \in \mathbb{S}^1$, and an aspect ratio A_R. The image plane, $\mathcal{S} = [0, w] \times [0, h]$, is the perspective projection of $\mathcal{W}$ as seen through the camera FoV, where w and h are the width and height of

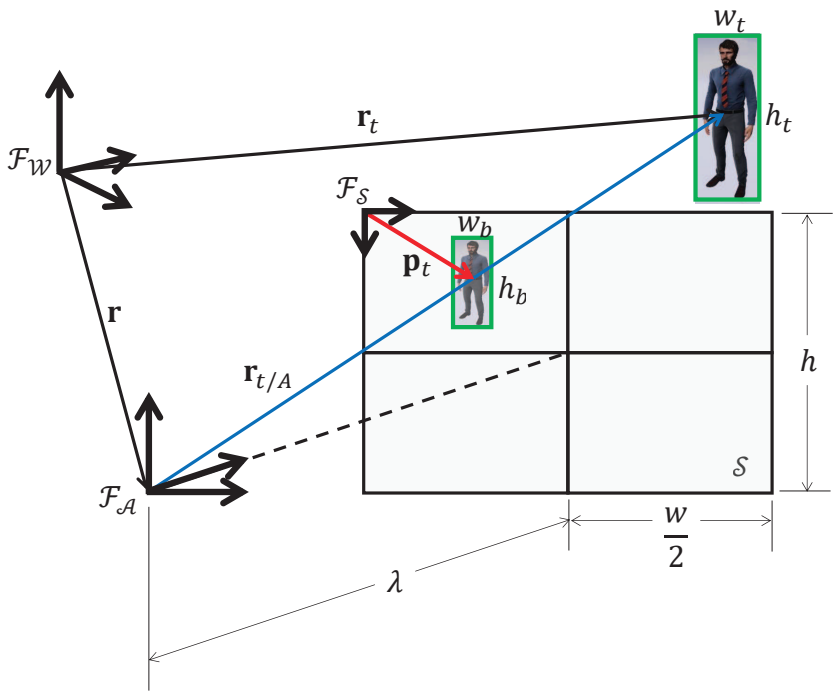


Figure 2.6: The inertial, body, and image reference frames are illustrated, along with relative position vectors and dimensions. The target is shown in the 3-dimensional world being transformed, via perspective projection, onto the image plane.

the image, respectively, which may be computed from the camera parameters λ, α, and A_R, that is $w = 2\lambda \tan \alpha$, and $h = w/A_R$ Fig. 2.6.

A reference frame $\mathcal{F}_{\mathcal{A}}$ is embedded in $\mathcal{A}$, such that the 1st axis is aligned with the camera optical axis, the 3rd axis points vertically, and the 2nd axis completes the right-hand rule. The origin of $\mathcal{F}_{\mathcal{A}}$ is known as the focal point of the camera, whose position $\mathbf{r}(t) \in \mathbb{R}^3$ is expressed relative to an inertial reference frame $\mathcal{F}_{\mathcal{W}}$, as illustrated in Fig. 2.6. The coordinates of the focal point position, expressed in $\mathcal{F}_{\mathcal{W}}$ are given as $\mathbf{r}(t) = [x(t), y(t), z]^T$, where z is the constant height of the focal point. Assuming the z-axes of $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_{\mathcal{A}}$ remain parallel, the rotation matrix

which maps vectors expressed in $\mathcal{F}_{\mathcal{W}}$ to $\mathcal{F}_{\mathcal{A}}$ is given as

$$\mathbf{R}_W^A = \begin{bmatrix} \cos\theta(t) & \sin\theta(t) & 0 \\ -\sin\theta(t) & \cos\theta(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.23}$$

where $\theta(t)$ is the yaw angle of the robot.

The perspective projection $\mathbf{P}_A^S$ maps vectors in $\mathbb{R}^3$ expressed with respect to $\mathcal{F}_{\mathcal{A}}$, whose origin is the focal point of the camera, to vectors in $\mathbb{R}^2$ expressed with respect to $\mathcal{F}_{\mathcal{S}}$. Fig. 2.6 shows the vector $\mathbf{r}_{t/A} \in \mathcal{W}$ being mapped to the vector $\mathbf{p}_t \in \mathcal{S}$ via $\mathbf{P}_A^S$. $\mathbf{p}(t) = [x_b(t), y_b(t)]^T$ is the position vector to the center of the bounding box in the image plane. The perspective projection can be written as a scaled linear operation on $\mathbf{r}_{t/A}$ using homogeneous coordinates ([**?**]). The perspective projection mapping of an arbitrary vector $\mathbf{r} \in \mathcal{W}$ to the associated image plane vector $\mathbf{p} \in \mathcal{S}$ is

$$\mathbf{P}_A^S(\mathbf{r}) = \frac{1}{2}\begin{bmatrix} w \\ b \\ 0 \end{bmatrix} - \gamma \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{r} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}, \tag{2.24}$$

where $\gamma \in \mathbb{R}$ is a scaling parameter used to enforce the 3rd element to equal unity. Therefore, all of the required transformations have been defined which take the target position $\mathbf{r}_t$, and the robot position $\mathbf{r}$, expressed in inertial frame, and define a position vector on the image plane $\mathbf{p}_t$. This complete transformation is illustrated in Fig. 2.6 and may be expressed as

$$\begin{bmatrix} \mathbf{p}_t \\ 1 \end{bmatrix} = \mathbf{P}_A^S \left(\mathbf{R}_W^A (\mathbf{r}_t - \mathbf{r}) \right). \tag{2.25}$$

Assuming $\mathcal{A}$ is rigid, and the camera is rigidly attached to $\mathcal{A}$, the robot configuration (state) vector can be described as $\mathbf{q}(t) = [x(t)\ y(t)\ \theta(t)]^T$, which is governed

by the nonholonomic unicycle kinematic model,

$$\dot{\mathbf{q}}(t) = \begin{bmatrix} \cos\theta(t) & 0 \\ \sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \mathbf{G}(\mathbf{q}(t))\mathbf{u}(t), \tag{2.26}$$

where $v(t) \in \mathbb{R}$ is the forward velocity, and $\omega(t) \in \mathbb{R}$ is the yaw rate. The control input vector is defined as $\mathbf{u}(t) = [v(t)\ \omega(t)]^T \in \mathcal{U}$, and $\mathcal{U} \subset \mathbb{R}^2$ is the set of admissible control inputs.

A bounding box $\mathbf{b}(t) \in \mathbb{R}^4$ associated with the target $\mathcal{T}$ is extracted from the video and projected on $\mathcal{S}$. The elements composing $\mathbf{b}(t) = [x_b(t), y_b(t), w_b(t), h_b(t)]^T$ are the coordinates of the bounding box center $\mathbf{p}_t(t) = [x_b(t), y_b(t)]^T \in \mathcal{S}$, expressed in the image frame $\mathcal{F}_\mathcal{S}$, and the width $w_b(t) \in [0, w]$ and height $h_b(t) \in [0, h]$ of the bounding box. Several computer vision algorithms exist which extract such a bounding box containing an object of interest from an image, some of which will be reviewed in the following section.

Control Objective

In order to maintain the target within the camera FoV, a reliable target bounding box must be consistently extracted from the video sequence. Therefore, it is desirable to maintain the target not only within the FoV, but at a reliable range for accurate image processing. The control objective is to drive the bounding box $\mathbf{b}(t)$ to a desired set point $\tilde{\mathbf{b}}$ by suitable choice of the control input vector $\mathbf{u}(t)$ over some time interval of interest $[t_0, T) \subseteq \mathbb{R}$. That is,

$$[x_b, y_b, w_b, h_b]^T \to [\tilde{x}_b, \tilde{y}_b, \tilde{w}_b, \tilde{h}_b]^T. \tag{2.27}$$

where $\tilde{x}_b$, $\tilde{y}_b$ are the desired (constant) coordinates of the bounding box center, and $\tilde{w}_b$, $\tilde{h}_b$ are the desired (constant) width and height of the bounding box, respec-

tively. Typically, the set point $\tilde{\mathbf{b}}$ is chosen such that the bounding box remains in the center of the image at a sufficient scale for reliable image processing. However, since there is no control over the pitch of the camera or the relative height of the camera focal point and the target center, $y_b(t)$ is not controllable. As long as the target and robot both remain on a level surface with a reasonable relative height between the camera focal point and the target center, this uncontrollable variable will not affect the proposed algorithm's performance. A more complex model may assume the camera is mounted on a gimbal, enabling the camera to rotate with respect to the robot, which would provide a means to control $y_b(t)$, but such a model is not considered in this work.

Similarly, the width $w_b(t)$ and height $h_b(t)$ of the bounding box provide some redundant information since the robot cannot control the orientation of the target. Therefore, the width and height are combined into a single size metric $\Delta_b(t)$ of the bounding box, which is a measure of the length of the bounding box diagonal, $\Delta_b(t) = \|[w_b(t), h_b(t)]^T\|_2$. Therefore, after post-processing the extracted bounding box $\mathbf{b}(t) = [x_b(t), y_b(t), w_b(t), h_b(t)]^T$ into a useful output vector $\mathbf{y}(t) = [\Delta_b(t), x_b(t)]^T$, the control objective is reduced to

$$\mathbf{y}(t) \to \tilde{\mathbf{y}}, \tag{2.28}$$

where $\tilde{\mathbf{y}} = [\tilde{\Delta}_b, \tilde{x}_b]^T$, and $\tilde{\Delta}_b = \|[\tilde{w}_b, \tilde{h}_b]^T\|_2$.

Methodology

This dissertation presents a novel unified video processing and control approach for detecting and pursuing a human target in a complex unstructured environment, which is accomplished by stabilizing the control objective defined in the previous

section. Due to recent advancements of object recognition tasks in computer vision, the presented methodology employs a state-of-the-art deep convolutional neural network (CNN) [61] to detect and classify objects within the image plane. The output of the CNN is then processed to extract a bounding box $\mathbf{b}(t)$ associated with the target. This bounding box is then used to compute a control law designed to maintain the target human in a desirable position within the image for reliable future detections.

Target Detection and Identification

Over recent years algorithms for multi-class object detection in images have become extremely accurate, mostly due to the use of deep CNNs [49]). Three recent well-known architectures are Faster R-CNN proposed by [81], the Single Shot Multibox Detector (SSD) developed by [61], and the Region-based Fully Convolutional Network (R-FCN) by [24]. It is difficult to disambiguate the *best* architecture due the use of interchangable feature extraction and classification techniques. Furthermore, the work by [49] present a comprehensive study of the speed-accuracy tradeoff between different CNN architectures and feature extractors. The object detection algorithm used in this work is chosen to be as accurate as possible while simultaneously being capable of real-time implementation on a physical robot with onboard computing. This work uses a MobileNet ([47]) implementation of the SSD architecture ([61]) in order to satisfy real-time resource constraints. The CNN is pre-trained on the Microsoft COCO data set of [57].

The CNN takes as input an image $\mathcal{S}(t)$ at time t and outputs a set of detection-confidence pairs $\mathcal{B}(t) = \{(\mathbf{b}_i(t), c_i(t))\}_{i=1}^{N_{det}(t)}$, where $\mathbf{b}_i(t)$ are bounding boxes containing objects of the same class as the target (i.e., human), and $c_i(t) \in [0, 1)$ are

the associated confidence scores of the bounding box, and $N_{det}(t) \in \mathcal{N}$ is the number of detections. The target bounding box $\mathbf{b}(t)$ is then computed as the bounding box with the highest associated confidence score, when multiple detections are extracted. If no detections are extracted, i.e., $\mathcal{B}(t) = \emptyset$, the target bounding box $\mathbf{b}(t)$ is set equal to the set point bounding box $\tilde{b}$ in order to stop the robot. This processing step is expressed as

$$\mathbf{b}(t) = \begin{cases} \arg\max_{\mathbf{b}_i}\{c_i : (\mathbf{b}_i, c_i) \in \mathcal{B}(t)\} & \mathcal{B}(t) \neq \emptyset \\ \tilde{\mathbf{b}} & \mathcal{B}(t) = \emptyset \end{cases}. \tag{2.29}$$

This formulation guarantees that $\mathbf{b}(t)$ exists and is unique by construction. The bounding box $\mathbf{b}(t)$ is then transformed into the output vector $\mathbf{y}(t)$ which is used as the control variable in the controller to be designed in the following subsection.

Controller Design

The control law for tracking and following the target based on video frames obtained by the robot camera, and processed according to the previous subsections, is developed by considering properties of the perspective projection, and noting how points in three dimensions move across an image in two-dimensions while the camera is moving. Due to the properties of the perspective projection, objects which are closer to the focal point appear larger, and objects that are farther from the focal point appear smaller. This provides a natural method for controlling the size of the bounding box $\Delta_b(t)$ without requiring kinematic estimations in the 3-dimensional world. Similarly, the position of the bounding box $x_b(t)$ in the image provides a natural error signal for the yaw rate $\omega(t)$ of the robot. Because the target human may be moving, the use of integral compensation is proposed in order to reduce steady state errors that would be present if the control input

were simply proportional to these error signals. Then, the proposed video-guided control input is designed using the following proportional-integral compensation

$$\mathbf{u}(t) = -\mathbf{K}_1 \Delta\mathbf{y}(t) - \mathbf{K}_2 \int_0^t \Delta\mathbf{y}(\tau) d\tau, \tag{2.30}$$

where $\mathbf{K}_1, \mathbf{K}_2 \succeq 0$ are diagonal gain matrices of reasonable dimension, and $\Delta\mathbf{y}(t) = \mathbf{y}(t) - \tilde{\mathbf{y}}$. The proposed control law is validated using photo-realistic simulations in highly complex environments, as well as through physical experiments in a laboratory setting.

Simulation Experiments

The Unreal Engine™is a leading game development software capable of advanced open-source environment development and manipulation. These capabilities have recently been exploited by several industries outside of the game development community, including architectural visualization, film-making, and virtual reality training simulations. The use of Unreal Engine™for simulation of computer vision-based automatic control algorithms makes no sacrifice to physical accuracy, but has the advantage of a vast user community composed of artists and developers who create visually realistic environments, characters, behaviors, and objects, which may be used in simulations. This ease of access to a diverse set of environments and scenarios helps test the robustness of proposed methods in ways not feasible in real-world experiments or conventional robotics software.

Similar game development softwares have been used in previous works for generating synthetic data to train deep CNNs. Unreal Engine™has also been used for similar computer vision tasks by several authors, such as semantic segmentation by [79], and simulating stereo-vision applications by [102]. Furthermore,

the work by [88] uses Unreal Engine™to develop realistic quadrotor and vehicle simulations for autonomous vehicle simulations. This work, proposes the use of Unreal Engine™for the novel task of simulating a fully autonomous robot using visual feedback for tracking a target in the realistic virtual space.

Human Tracking and Following Simulation Results

The control law developed in the previous section is tested in the visually realistic and complex subway environment using Unreal Engine™, in which a mobile robot equipped with a camera tracks and follows a potentially moving human without knowledge of the target dynamics or environment geometry. The subway environment consists of realistic lighting, a moving subway, and other moving objects.

A number of simulations are conducted in order to analyze the proposed control algorithm: (1) a step response of the velocity input $v(t)$, (2) a ramp response of the velocity input $v(t)$, (3) a step response of the yaw rate input $\omega(t)$, (4) a ramp response of the yaw rate input $\omega(t)$. The set point used for all simulations is chosen such that the bounding box stabilizes to a desired size for reliable image processing at the center of the camera FoV. The single setting for the diagonal gain matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ is manually chosen and not changed between simulations.

The first simulation tests the controller response to a step input in the size error of the target, i.e., $\Delta_b(t) - \tilde{\Delta}_b$. This is done by initially placing the target human in the center of the camera FoV, such that $x_b(t_0) = \tilde{x}_b$, and at a distance away from the robot such that $\Delta_b(t_0) < \tilde{\Delta}_b$. The bottom of Fig. 2.7 shows the initial configuration of the robot and target in a geometrically simplified visualization of the subway, along with the visual input to the robot at the initial time t_0. The set point bounding box $\tilde{\mathbf{b}}$ is illustrated as the orange bounding box and the

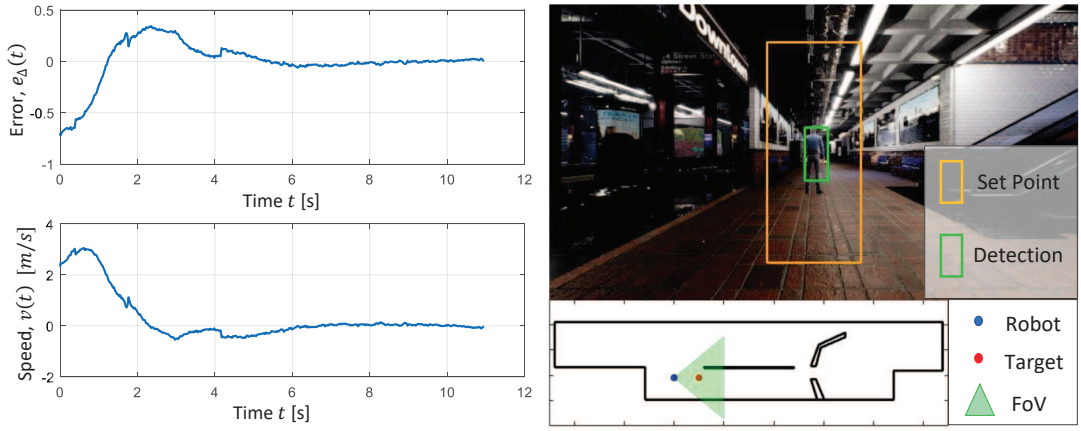


Figure 2.7: Simulated step response to an initial error in the size of the target, $e_{Delta} = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$. The bottom of the figure illustrates the initial configuration of the target and robot in a geometrically simplified visualization of the subway, along with the initial visual input at the initial configuration.

estimated target bounding box $\mathbf{b}(t_0)$ output from the CNN is illustrated as the green bounding box. The top of Fig. 2.7 shows the resulting error signal $e_\Delta(t) = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$ and control input $v(t)$. The step response slightly overshoots the desired position and stabilizes in roughly five seconds.

The second simulated experiment tests the controller response to a ramp input in the size error of the target, $\Delta_b(t) - \tilde{\Delta}_b$. This is done by initially placing the target human in the center of the camera FoV, such that $x_b(t) = \tilde{x}_b$, and programming the human target to walk at a constant velocity in the direction of the initial camera optical axis. The human is programmed to walk at 1.3 m/s, which is a typical walking speed of a human. Fig. 2.8 illustrates the controller response as well as several snapshots throughout the simulation showing the robot-target configuration and the associated visual input. The response of the controller to such an input again stabilizes without any steady state error, due to the integral term of the control law in 2.30.

The next simulated experiment tests the controller response to a step input in

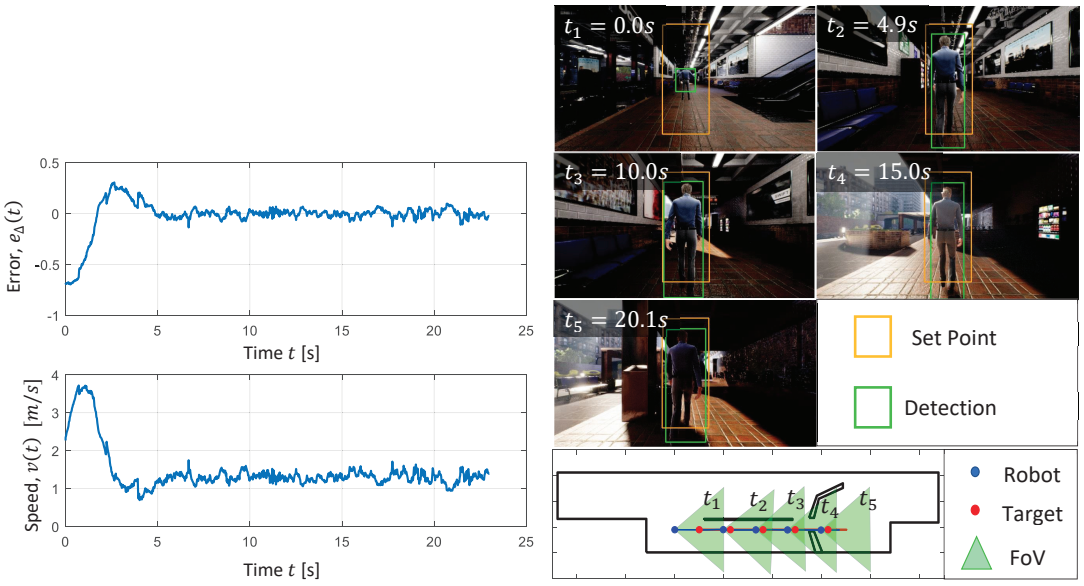


Figure 2.8: Simulated ramp response of the error in the size of the target, $e_\Delta = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$. The bottom of the figure illustrates the configuration of the target and robot in a geometrically simplified visualization of the subway at a number of snapshots throughout the simulation, along with the visual input at the time of these snapshots.

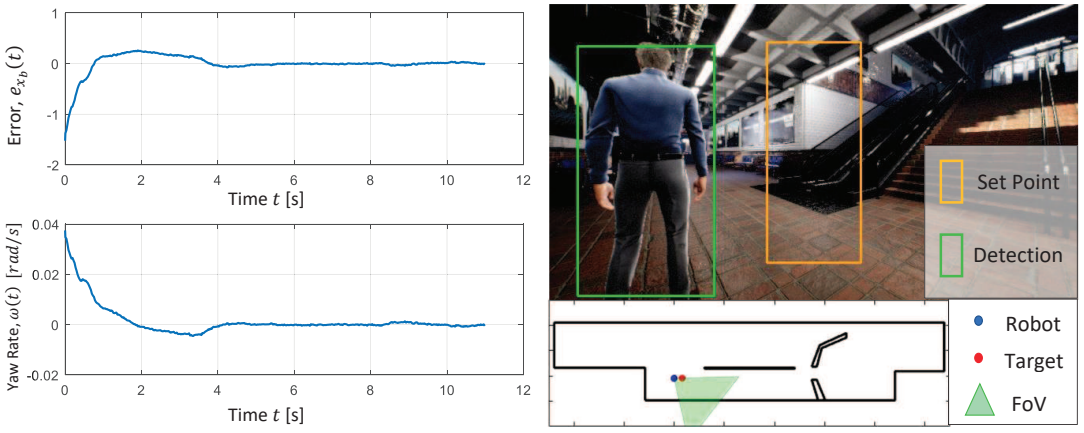


Figure 2.9: Simulated step response to an initial error in the position of the target, $e_{x_b} = (x_b(t) - \tilde{x_b})/\tilde{x_b}$. The bottom of the figure illustrates the initial configuration of the target and robot in a geometrically simplified visualization of the subway, along with the initial visual input at the initial configuration.

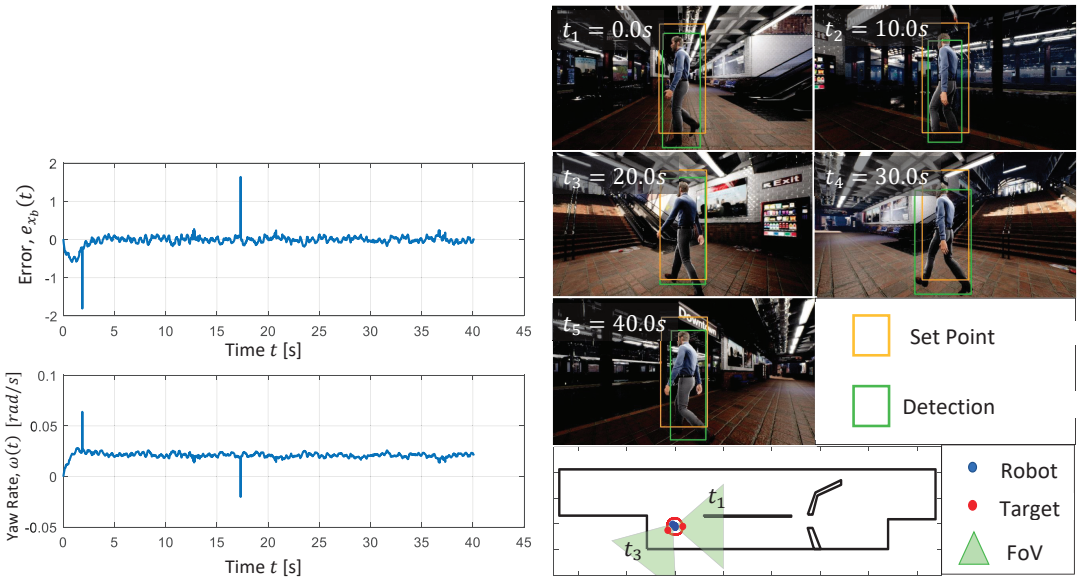


Figure 2.10: Simulated ramp response of the error in the position of the target, $e_{x_b} = (x_b(t) - \tilde{x_b})/\tilde{x_b}$. The bottom of the figure illustrates the configuration of the target and robot in a geometrically simplified visualization of the subway at a number of snapshots throughout the simulation, along with the visual input at the time of these snapshots.

the lateral position of the target, i.e., $x_b(t) - \tilde{x}_b$. This is done by initially placing the target human at a distance from the robot such that $\Delta_b(t_0) = \tilde{\Delta}_b$, but offset from the optical axis such that $x_b(t_0) < \tilde{x}_b$. The bottom of Fig. 2.9 shows the initial configuration of the robot and target in a geometrically simplified visualization of the subway, along with the visual input to the robot at the initial time t_0. The top of Fig. 2.7 shows the resulting error signal $e_{x_b}(t) = (x_b(t) - \tilde{x}_b)/\tilde{x}_b$ and control input $\omega(t)$. The step response slightly overshoots the desired position but quickly stabilizes about the set point configuration.

The final simulated experiment tests the controller response to a ramp input in the lateral position of the target, $x_b(t) - \tilde{x}_b$. This is done by programming the human target to walk in a circular motion centered at the camera focal point at a constant speed. The radius of the target's circular path is chosen such that $\Delta_b(t) = \tilde{\Delta}_b$. Fig 2.10 illustrates the controller response as well as several snap-

shots throughout the simulation showing the robot-target configuration and the associated visual input. The response of the controller to this ramp input very rapidly stabilizes without any steady state error. Some high frequency oscillations in the error signal are visible, and are caused by the periodic nature of the human walking as viewed from the side. That is, the bounding box slightly changes in shape and position due to swinging arms and legs of a walking human. Two of the spikes in the signal are caused by errors in the CNN detection algorithm, but are only present at single frames, which does not affect performance.

The four simulations performed in this study all show the controller stabilizing about the desired set point. Furthermore, the robot was programmed to follow the human through the subway environment using the proposed controller, while the target moved arbitrarily through the environment. Even in this case, where the robot was subject to small disturbances such as brief/partial occlusions, lighting variations, and change in the target motion the robot successfully stabilized about the set point. In this case, the robot was able to follow the human through the subway environment for several minutes, and possibly longer. Therefore, these simulation results suggest that, as long as the human target does not intentionally evade the robot, then the proposed controller will be capable of following the human indefinitely under reasonable conditions.

Human Tracking and Following Experimental Results

The experimental validation of the proposed control algorithm is done using a Clearpath™Jackal robot equipped with a camera and onboard computing capabilities. The robot can be accurately modeled by the nonholonomic unicycle model 2.26. A Vicon motion capture system is used to provide ground truth measure-

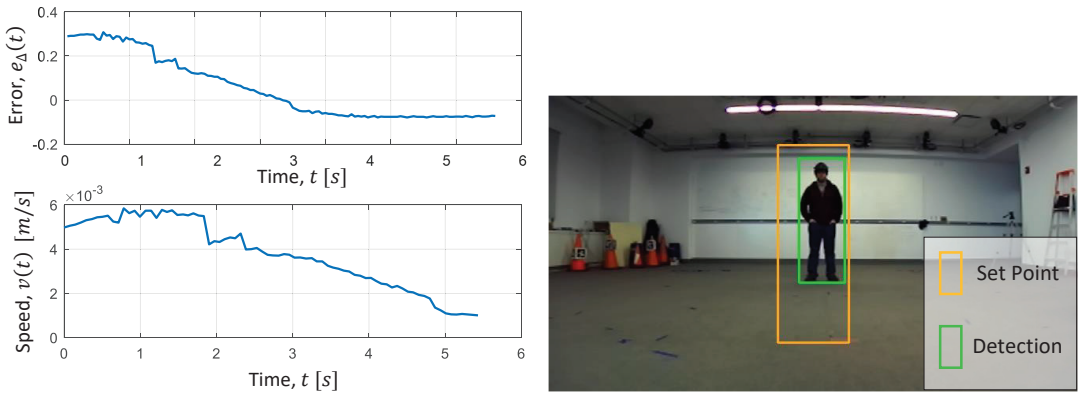


Figure 2.11: Simulated step response to an initial error in the size of the target, $e_{Delta} = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$. The bottom of the figure illustrates the initial visual input.

ments of the robot and target states. It should be made clear that the Vicon data is never made available to the robot, and is only used to collect accurate pose data for results visualization. The four physical experiments presented here are exactly the same as the four simulated experiments in the previous section. That is : (1) a step response of the velocity input $v(t)$, (2) a ramp response of the velocity input $v(t)$, (3) a step response of the yaw rate input $\omega(t)$, (4) a ramp response of the yaw rate input $\omega(t)$

The first physical experiments tests the controller response to a step input in the size error of the target, i.e., $\Delta_b(t) - \tilde{\Delta}_b$. This is done by initially placing the target human in the center of the camera FoV, such that $x_b(t_0) = \tilde{x}_b$, and at a distance away from the robot such that $\Delta_b(t_0) < \tilde{\Delta}_b$. The bottom of Fig. 2.11 shows the visual input to the robot at the initial time t_0. The set point bounding box $\tilde{\mathbf{b}}$ is illustrated as the orange bounding box and the estimated target bounding box $\mathbf{b}(t_0)$ output from the CNN is illustrated as the green bounding box. The top of Fig. 2.11 shows the resulting error signal $e_\Delta(t) = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$ and control input $v(t)$. The step response slightly overshoots the desired position then quickly stabilizes.

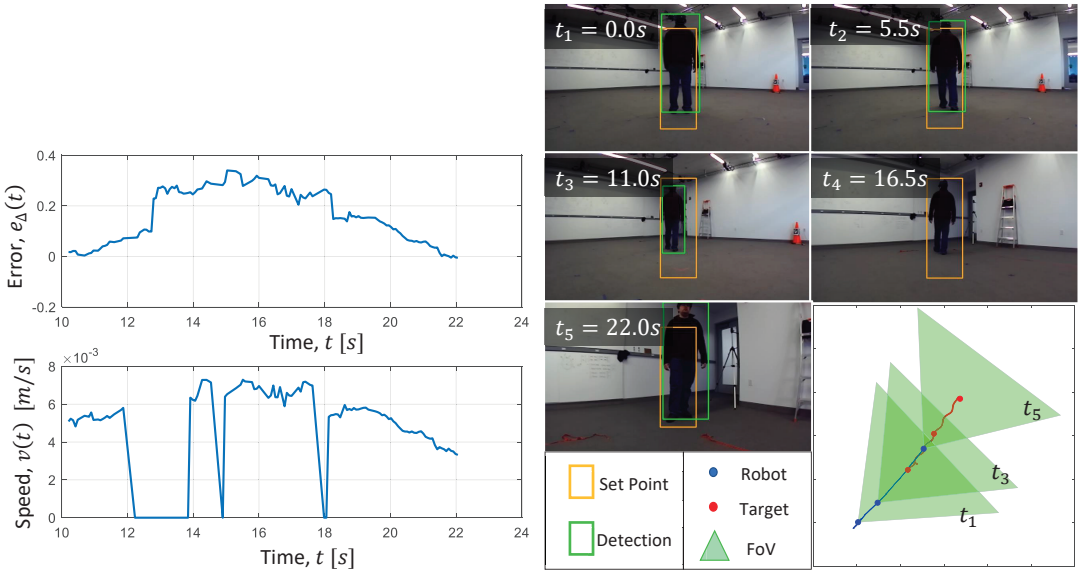


Figure 2.12: Simulated ramp response of the error in the size of the target, $e_\Delta = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$. The bottom of the figure illustrates the configuration of the target and robot at a number of snapshots throughout the simulation, along with the visual input at the time of these snapshots.

The second physical experiment tests the controller response to a ramp input in the size error of the target, $\Delta_b(t) - \tilde{\Delta}_b$. This is done by initially placing the target human in the center of the camera FoV, such that $x_b(t) = \tilde{x}_b$, and programming the human target to walk at a constant velocity in the direction of the initial camera optical axis. Fig. 2.12 illustrates the controller response as well as several snapshots throughout the simulation showing the robot-target configuration and the associated visual input. The response of the controller to such an input again stabilizes. However, due to the physical limitations of the laboratory setup (i.e., Size of the Vicon area) the human cannot walk far enough to allow the robot to fully reach steady state, but extrapolation of the available response is promising. This further illustrates the power of visually-realistic simulation in Unreal Engine™.

The next physical experiment tests the controller response to a step input in

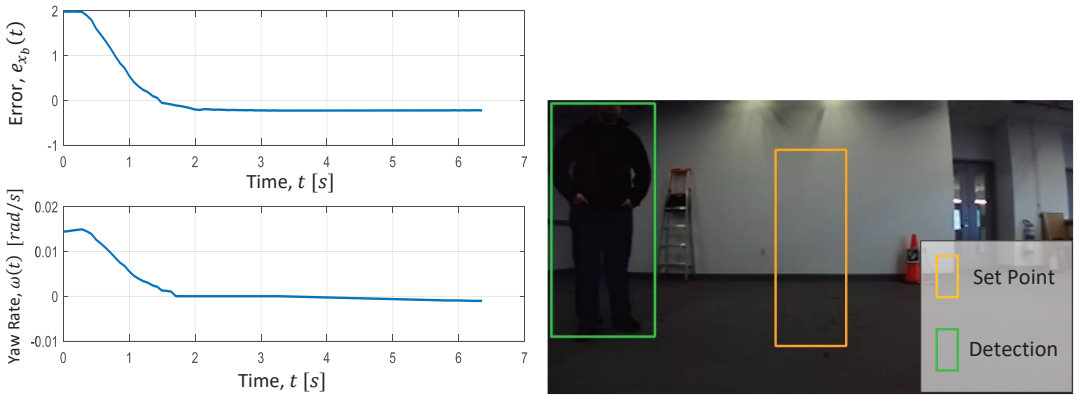


Figure 2.13: Simulated step response to an initial error in the position of the target, $e_{x_b} = (x_b(t) - \tilde{x_b})/\tilde{x_b}$. The bottom of the figure illustrates the initial visual input.

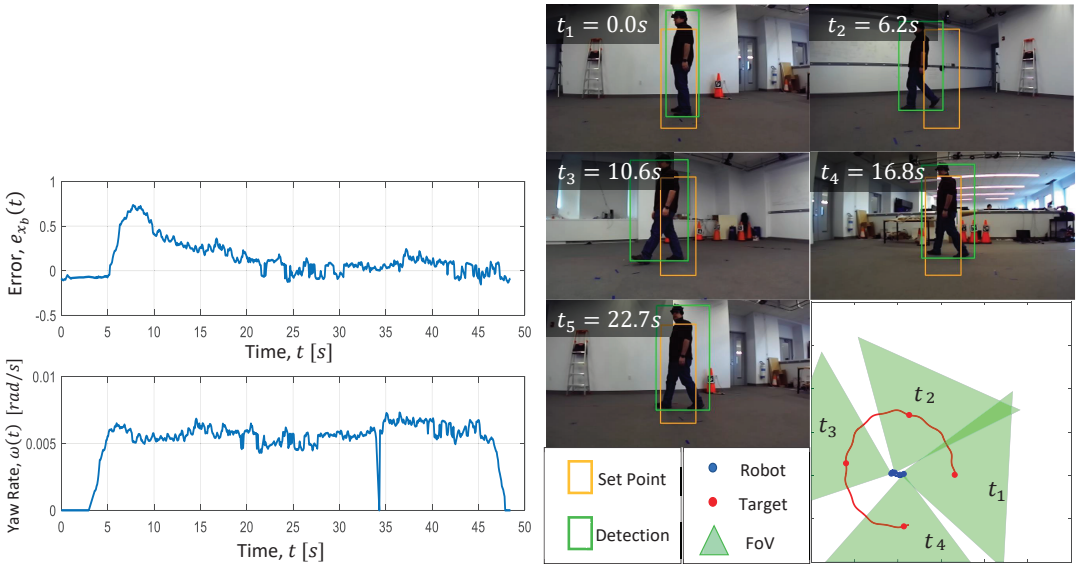


Figure 2.14: Simulated ramp response of the error in the position of the target, $e_{x_b} = (x_b(t) - \tilde{x_b})/\tilde{x_b}$. The bottom of the figure illustrates the configuration of the target and robot at a number of snapshots throughout the simulation, along with the visual input at the time of these snapshots.

the lateral position of the target, i.e., $x_b(t) - \tilde{x}_b$. This is done by initially placing the target human at a distance from the robot such that $\Delta_b(t_0) = \tilde{\Delta}_b$, but offset from the optical axis such that $x_b(t_0) < \tilde{x}_b$. The bottom of Fig. 2.13 shows the initial visual input to the robot at t_0. The top of Fig. 2.7 shows the resulting error signal $e_{x_b}(t) = (x_b(t) - \tilde{x}_b)/\tilde{x}_b$ and control input $\omega(t)$. The step response slightly

has no overshoot and quickly stabilizes about the set point configuration.

The final physical experiment tests the controller response to a ramp input in the lateral position of the target, $x_b(t) - \tilde{x}_b$. This is done by the human target walking in a circular motion centered at the camera focal point at a constant speed. Fig 2.14 illustrates the controller response as well as several snapshots throughout the simulation showing the robot-target configuration and the associated visual input. The response of the controller to this ramp input very rapidly stabilizes without any steady state error. These results confirm that the proposed controller as well as future computer vision-based controllers can be readily simulated in Unreal Engine™and then successfully implemented on physical robotic platforms.

2.7 Action Recognition and Human-directed Quadrotor Control

The previously discussed tasks enable an event-camera-equipped robot to navigate and track a particular object using only event-driven algorithms. Action recognition enables mobile robots to obey simple gestures and commands, such as to move in a specified direction based on the direction of a human pointing. Action recognition is incredibly useful for autonomous systems that share an environment with humans. Many algorithms exist in the literature for human action recognition, which is an active area of research for machine learning and deep learning algorithms [90]. However, the majority of these algorithms require significant computational resources or rely on RGB camera image intensity values. This work employs a purely motion-based action recognition algorithm using Temporal Templates [9], and more particularly the Motion Energy Image (MEI) and

Motion History Image (MHI) representations of actions. The event-camera action recognition algorithm constructs the MEI and MHI from event data. Given the set of events collected from a motion sequence over the time interval $[t-\tau, t]$, where $\tau > 0$ is a temporal sliding window, the MEI and MHI are computed, respectively, as,

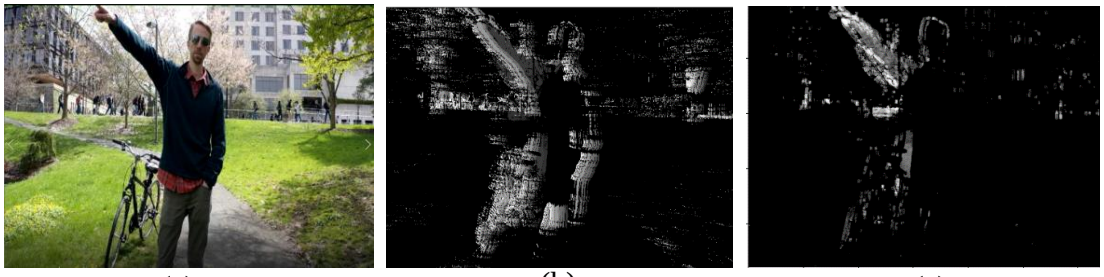
$$E_\tau(\mathcal{E}) = \bigcup_{\mathbf{e}(t)\in\mathcal{E}} \{(i,j)|\mathbf{e}(t) = [i \quad j \quad t \quad p]\} \tag{2.31}$$

$$h_{ij} = \max_{\mathbf{e}(t)\in\mathcal{E}} (\{0\} \cup \{\tau - t|\mathbf{e}(t) = [i \quad j \quad t \quad p]\}) \tag{2.32}$$

where h_{ij} represents a single element of the MHI matrix,

$$H_\tau = \begin{bmatrix} h_{11} & h_{11} & \cdots & h_{11} \\ h_{11} & h_{11} & \cdots & h_{11} \\ \vdots & \vdots & \ddots & \vdots \\ h_{11} & h_{11} & \cdots & h_{11} \end{bmatrix} \tag{2.33}$$

where the dependency on the event set is omitted for brevity. The resulting MHI matrix has larger component values where more recent motion has occurred and the values decay as the motion occurred longer ago. Figure 2.15 shows an example of the temporal templates for a human pointing computed using an event-camera and an RGB camera, for comparison. The temporal templates naturally lend themselves to event data since they rely on image differences as a main part of the method. The shape of MEI and MHI can effectively represent human motion and they are used to extract a feature representation of actions. We use the first 7 Hu moments [48] as a global shape descriptor of MEI and MHI that are invariant to scale, translation, and rotation. Then, a recognition scheme matching the distance between the moments of testing data and those of stored actions is developed. The distance metric is Mahalanobis distance, denoted by D_M. The distance D_M



(a) (b) (c)

Figure 2.15: Temporal Templates for a human pointing from event and RGB camera. The action of a human pointing (a) results in the MHI computed from RGB camera (b) and event camera (c).

between a test motion sequence and the combined distribution of training motion sequences is given by:

$$D_M = ((\boldsymbol{\mu} - \bar{\boldsymbol{\mu}})^T \mathbf{K}_s^{-1} (\boldsymbol{\mu} - \bar{\boldsymbol{\mu}}))^{1/2} \tag{2.34}$$

where $\mathbf{K}_s$ and $\bar{\boldsymbol{\mu}}_s$ are respectively the covariance matrix and mean of the Hu moments of training motions, and $\boldsymbol{\mu}$ is the Hu moments of testing motion. Using distance D_M, testing data can be matched to the closest training data to determine the best matching action label.

The event-camera active perception algorithms presented in this dissertation are compared to the analogous RGB camera algorithms that use standard algorithms in the following sections. The event-camera is a biologically inspired sensor that is expected to require significantly less computational resources while performing as good, if not better, than RGB camera methods for tasks that are heavily influenced by motion in the camera image plane. This dissertation quantitatively and qualitatively analyzes the comparison of event and RGB cameras in terms of both accuracy and computational cost for each of the high-level active perception tasks discussed in this section.

2.7.1 3D Human Pose and Action Recognition

This section develops a novel and systematic approach to real-time three-dimensional human pose estimation for *viewpoint invariant* action recognition from a monocular camera onboard a quadrotor tasked with actively tracking a human of interest. Leveraging recent advancements in two-dimensional human pose estimation, a novel geometric approach developed here characterizes the three-dimensional pose of each link in the human skeleton using rotations, and the three-dimensional joint angles are derived analytically. A rotational representation of pose is invariant to the absolute scale of the human, depending only on body proportion which is consistent for adult humans. Furthermore, the joint angles can be expressed with respect to a human-fixed reference frame, thereby making the pose representation invariant to the camera viewpoint. A human action feature representation is also developed, taking advantage of the viewpoint invariant pose and a weighted sliding temporal window, that is used for viewpoint invariant human action recognition. Finally, a switched controller informed by the target human action is developed enabling a quadrotor to maintain the human in camera's field of view. Results of the three-dimensional human pose estimation are shown to accurately represent the true pose of a human. Additionally, the action-based switched controller is shown to maintain the human in the camera's field of view while the human performs various actions that would otherwise require controller tuning or re-design.

The 3D HPE results are shown to accurately compute the orientation of multi-link human limbs and skeletons in real-time for use onboard a mobile robot. In addition, the proposed viewpoint invariant feature representation is shown to be capable of capturing temporal information and to be more robust than existing

Figure 2.16: Two camera viewpoints of the same human walking sequence. The viewpoint invariant pose angles developed in this dissertation are shown on the right of both images for the human's left arm. The invariant pose angles results in the viewpoint invariant action recognition result of 'walking'.

feature representations of the task of HAR. The switched controller developed in this dissertation is also shown to perform well for a human performing several different actions that would likely require re-tuning of control parameters of non-action based controllers.

Problem Formulation

This dissertation considers the problem of a mobile quadrotor tasked with maintaining a human target within the camera's FOV by taking advantage of the perceived actions of the human. The region of interest (ROI), or workspace $\mathcal{W} \subset \mathbb{R}^3$, contains a quadrotor that is equipped with a monocular RGB camera with FOV

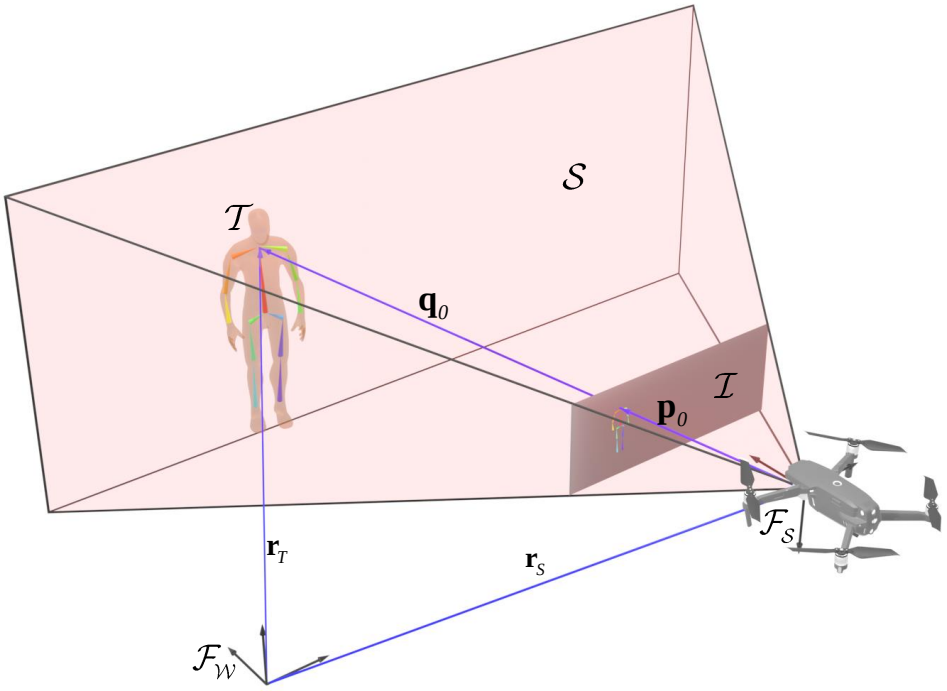


Figure 2.17: A quadrotor with an onboard camera with FOV $\mathcal{S}$ observes a human target $\mathcal{T}$. The target skeleton is shown projected from the workspace $\mathcal{W}$ onto the image plane $\mathcal{I}$ using perspective projection. Additionally, the target root joint κ_0, which has position $\mathbf{q}_0$ relative to the camera frame $\mathcal{F}_\mathcal{S}$, is projected to the position $\mathbf{p}_0$ on the image plane. The camera focal point and target root joint have positions $\mathbf{x}$ and $\mathbf{x}_T$, respectively, defined relative to the fixed coordinate frame $\mathcal{F}_\mathcal{W}$.

$\mathcal{S} \subset \mathcal{W}$. The camera FOV $\mathcal{S}$ is modeled as a right-rectangular pyramid with apex position $\mathbf{r}_S \in \mathcal{W}$ defined relative to a fixed coordinate frame $\mathcal{F}_\mathcal{W}$. The apex $\mathbf{r}_S$ of the camera FOV $\mathcal{S}$ coincides with the camera's focal point following the pinhole camera model [98]. A camera-fixed reference frame $\mathcal{F}_\mathcal{S}$ is embedded in the sensor FOV with its origin $\mathcal{O}_\mathcal{S}$ located at the camera focal point $\mathbf{r}_S$. The camera-fixed frame $\mathcal{F}_\mathcal{S}$ can then be used to describe every point in the sensor FOV $\mathcal{S}$ relative to the fixed reference frame $\mathcal{F}_\mathcal{W}$. The image plane $\mathcal{I} \subset \mathcal{S}$ is a planar manifold that is normal to the camera's optical axis and points in the camera FOV are projected onto the camera image plane from perspective projection as shown in Fig. 2.17.

The quadrotor has a state vector $\mathbf{x} \in \mathbb{R}^{12}$ that is composed of the quadrotor's

position $\mathbf{r}_S \in \mathbb{R}^3$, velocity $\dot{\mathbf{r}}_S \in \mathbb{R}^3$, attitude $(\phi_S, \theta_S, \psi_S)$, and angular rate $\boldsymbol{\omega} \in \mathbb{R}^3$. Where the attitude of the quadrotor is characterized by the three Euler angles composed of the aircraft's roll $\phi_S \in \mathbb{S}^1$, pitch $\theta_S \in \mathbb{S}^1$ and yaw $\mathbb{S}^1$. These Euler angles characterize the attitude of the reference frame $\mathcal{F}_\mathcal{S}$ with respect to the inertial frame $\mathcal{F}_\mathcal{W}$. The nonlinear quadcopter dynamics is modeled in general as,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2.35}$$

where $\mathbf{u} \in \mathcal{U}$ is the quadcopter control input and $\mathcal{U}$ is the set of admissible control inputs.

In addition to the quadrotor and sensor FOV, the ROI is also occupied by a human target $\mathcal{T} \subset \mathcal{W}$ whose pose is represented by 14 joint positions $\kappa_i \in \mathcal{T},\ i = 0, ..., 13$ as shown in Fig. 2.18. Any pair of two connected joints $(\kappa_i,\ \kappa_j)$ is referred to as a *link*. The vector from the camera focal point $\mathcal{O}_S$ to the joint κ_i is defined as $\mathbf{q}_i = [x_i\ y_i\ z_i]^T$, expressed in terms of the camera-fixed reference frame $\mathcal{F}_\mathcal{S}$. The relative position of joint κ_j with respect to κ_i is defined as $l_{ij}\hat{\mathbf{e}}_{ij} \triangleq \mathbf{q}_j - \mathbf{q}_i$, where the length of the link connecting κ_j to κ_i is $l_{ij} = \|\mathbf{q}_i - \mathbf{q}_j\|$ and the unit vector $\hat{\mathbf{e}}_{ij}$ is in the direction from κ_i to κ_j. The vector that describes the position of a joint κ_i projected onto the image plane $\mathcal{I}$ is denoted by $\mathbf{p}_i = [\xi_i\ \eta_i\ \lambda]^T$, whose coordinates are expressed in terms of the camera-fixed reference frame $\mathcal{F}_\mathcal{S}$. $\lambda > 0$ is the camera's known focal length, and ξ_i, η_i are the horizontal and vertical positions of κ_i projected onto the image plane, i.e., the pixel location in the image. From perspective projection, $\mathbf{p}_i$ can be expressed as a scaled vector of the position $\mathbf{q}_i$,

$$\mathbf{p}_i = \frac{\lambda}{z_i}\mathbf{q}_i \tag{2.36}$$

The joint κ_0, which references the approximate neck position of the human, is referred to as the *root* joint for the target human which is used to describe the target's position $\mathbf{r}_T$ relative to $\mathcal{F}_\mathcal{W}$, or $\mathbf{q}_0$ relative to $\mathcal{F}_S$, as shown in Fig. 2.17.

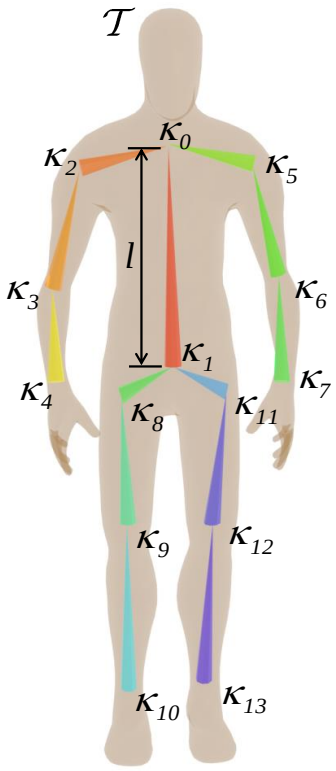


Figure 2.18: A human target $\mathcal{T} \in \mathcal{W}$ with skeleton represented by 14 joint positions $\kappa_i,\ i = 0, ..., 13$. κ_0 is referred to as the root joint and is used to define the target position $\mathbf{r}_T$ relative to the fixed reference frame $\mathcal{F}_\mathcal{W}$.

Because the perspective projection is a mapping from $\mathbb{R}^3$ to $\mathbb{R}^2$, the depth component cannot easily be computed from the camera measurements. However, in this dissertation it is assumed that the human target is a reasonably proportioned human, meaning that the ratio of any link's length over the torso length, defined as $l = \|\mathbf{q}_1 - \mathbf{q}_0\|$, can be accurately estimated. This assumption is commonly used in many industries and dates back to the work of Leonardo Di Vinci's Vitruvian Man [86]. Mathematically, the following link ratios are assumed known,

$$\gamma_{ij} \triangleq \frac{l_{ij}}{l} \tag{2.37}$$

for all $i,\ j = 0, ..., 13$ such that κ_i and κ_j are connected joints according to Fig. 2.18 and $i \neq j$.

In this dissertation, the pose of a link $(\kappa_i,\ \kappa_j)$ is defined using two Euler angles, denoted by φ_{ij} and ϑ_{ij}. In order to ensure that the estimated pose is viewpoint invariant, $\varphi_{ij},\ \vartheta_{ij}$ are defined relative to a target-fixed reference frame $\mathcal{F}_{\mathcal{T}}$. $\mathcal{F}_{\mathcal{T}}$ has origin at κ_0 and is composed of three unit vectors $(\hat{\mathbf{e}}_{T1},\ \hat{\mathbf{e}}_{T2},\ \hat{\mathbf{e}}_{T3})$ as shown in Fig. 2.19. The direction of the unit vector $\hat{\mathbf{e}}_{T1}$ is defined to be normal to the plane formed by the three joints $\kappa_1,\ \kappa_2,\ \kappa_5$. This normal vector is computed using the vector cross product,

$$\hat{\mathbf{e}}_{T1} = \hat{\mathbf{e}}_{51} \times \hat{\mathbf{e}}_{21} \tag{2.38}$$

The unit vector $\hat{\mathbf{e}}_{T3}$ is in the direction from the root joint κ_0 to the joint κ_1 projected into the plane formed by $\kappa_1,\ \kappa_2,\ \kappa_5$, or

$$\hat{\mathbf{e}}_{T3} \triangleq \frac{\hat{\mathbf{e}}_{10} - (\hat{\mathbf{e}}_{10} \cdot \hat{\mathbf{e}}_{T1})\, \hat{\mathbf{e}}_{T1}}{\|\hat{\mathbf{e}}_{10} - (\hat{\mathbf{e}}_{10} \cdot \hat{\mathbf{e}}_{T1})\, \hat{\mathbf{e}}_{T1}\|} \tag{2.39}$$

where $(\cdot)$ denotes the vector dot product. The final unit vector $\hat{\mathbf{e}}_{T2}$ is defined using the right hand rule and therefore the target reference frame is completely defined as shown in Fig. 2.19.

Having defined the target reference frame $\mathcal{F}_{\mathcal{T}}$, the Euler angles $\varphi_{ij},\ \vartheta_{ij}$ that define the pose of the link $(\kappa_i,\ \kappa_j)$ may be defined in terms of two consecutive rotations. The first rotation is about the $\hat{\mathbf{e}}_{T3}$ axis followed by a rotation about the intermediate $\hat{\mathbf{e}}_{T2}$ axis, as shown in Fig. 2.20. By this convention, if both $\varphi_{ij} = \vartheta_{ij} = 0$, then the vector $\mathbf{s}_{ji}$ points in the direction of $\hat{\mathbf{e}}_{T1}$. The full 3D human pose can now be represented as a 26-dimensional viewpoint invariant vector of Euler angles, composed of two angles for each of the 13 links,

$$\begin{aligned} \mathbf{p}_T \triangleq [&\varphi_{01}\ \vartheta_{01}\ \varphi_{02}\ \vartheta_{02}\ \varphi_{23}\ \vartheta_{23}\ \varphi_{34}\ \vartheta_{34} \ldots \\ &\varphi_{05}\ \vartheta_{05}\ \varphi_{56}\ \vartheta_{56}\ \varphi_{67}\ \vartheta_{67} \ldots \\ &\varphi_{18}\ \vartheta_{18}\ \varphi_{89}\ \vartheta_{89}\ \varphi_{9,10}\ \vartheta_{9,10} \ldots \\ &\varphi_{1,11}\ \vartheta_{1,11}\ \varphi_{11,12}\ \vartheta_{11,12}\ \varphi_{12,13}\ \vartheta_{12,13}]^T \end{aligned} \tag{2.40}$$

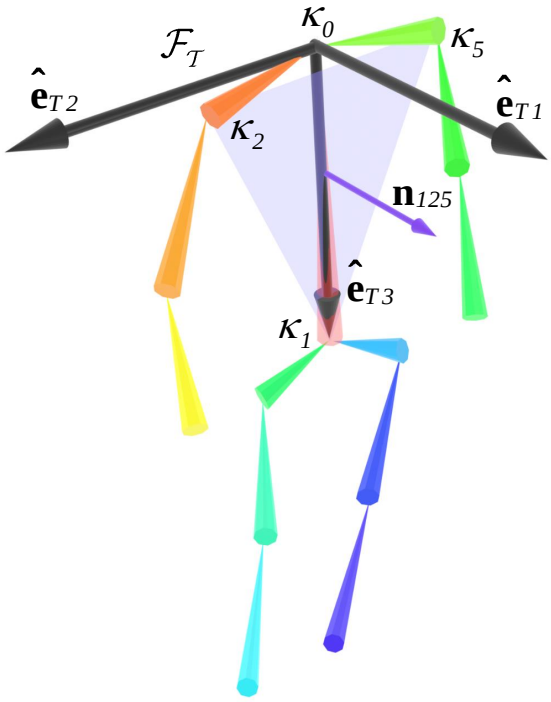


Figure 2.19: The target reference frame composed of orthogonal unit vectors $\hat{\mathbf{e}}_{T1}, \hat{\mathbf{e}}_{T2}, \hat{\mathbf{e}}_{T3}$, where $\hat{\mathbf{e}}_{T1}$ is defined normal to the plane of the three joints $\kappa_1, \kappa_2, \kappa_5$, $\hat{\mathbf{e}}_{T3}$ is in the direction of the torso (κ_0, κ_1) projected into the same plane, and $\hat{\mathbf{e}}_{T2}$ completes the right hand rule.

A novel contribution of this dissertation is that the pose vector $\mathbf{p}_T$ is, by definition, viewpoint invariant and human-scale invariant since all of the Euler angles in $\mathbf{p}_T$ are defined relative to the *target*-fixed reference frame, and the use of joint angles, rather than joint positions, is implicitly scale invariant.

Although the Euler angles φ_{ij}, ϑ_{ij} are viewpoint invariant and therefore extremely valuable for action recognition, these angles are not directly measurable from the camera after perspective projection onto the image plane. However, the in-plane rotation of each link is measurable, and one of the contributions of this dissertation is the derivation of the geometric constraints used to determine the out of plane rotation of each link.

For a link $(\kappa_i,\ \kappa_j) \in \mathcal{S}$ projected onto the image plane $\mathcal{I}$ as shown in Fig.

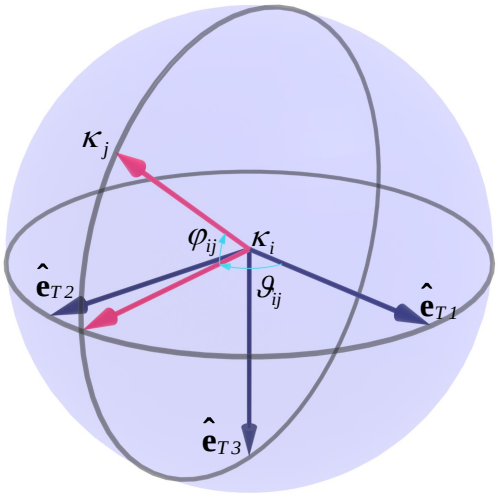


Figure 2.20: The azimuth elevation Euler angles defining the pose between two connected joints in the target reference frame.

2.21, the projected positions $\mathbf{p}_i,\ \mathbf{p}_j$ are estimated using a state of the art 2D HPE systems, e.g., [16, 51]. The image-plane positions $\mathbf{p}_i,\ \mathbf{p}_j$ are used to compute the projected distance d_{ij} between the two joints and the rotation angle θ_{ij} in the image plane made by the link connecting the two joints, given by,

$$d_{ij} = \|\mathbf{p}_j - \mathbf{p}_i\|, \quad \theta_{ij} = \arctan\left(\frac{\eta_j - \eta_i}{\xi_j - \xi_i}\right) \tag{2.41}$$

The complete target human pose can be described relative to the camera reference frame $\mathcal{F}_{\mathcal{S}}$, again using two Euler angles $\theta_{ij},\ \phi_{ij}$ to describe the pose of each pair of connected joints $\kappa_i,\ \kappa_j$, as shown in Fig. 2.21. Note that the out of plane angle ϕ_{ij} is measured in the plane of the three points $\kappa_i,\ \kappa_j,\ \mathcal{O}_{\mathcal{S}}$, where $\mathcal{O}_{\mathcal{S}}$ is the origin of the camera reference frame. The camera reference frame $\mathcal{F}_{\mathcal{S}}$ is composed of three unit vectors, namely $\hat{\mathbf{e}}_{S1},\ \hat{\mathbf{e}}_{S2},\ \hat{\mathbf{e}}_{S3}$, where $\hat{\mathbf{e}}_{S1},\ \hat{\mathbf{e}}_{S2}$ are aligned with the horizontal (right-positive) and negative vertical (down-positive) directions of the image plane $\mathcal{I}$, respectively, and $\hat{\mathbf{e}}_{S3}$ is aligned with the camera's optical axis. The first rotation used to describe the pose of $(\kappa_i,\ \kappa_j)$ is a rotation of θ_{ij} about the $\hat{\mathbf{e}}_{S1}$ direction, which is directly measurable from the image using Eq. 2.41. This first

rotation matrix $\mathbf{R}_{\theta_{ij}} \in SO(3)$ is defined as

$$\mathbf{R}_{\theta_{ij}} \triangleq \begin{bmatrix} \cos\theta_{ij} & \sin\theta_{ij} & 0 \\ -\sin\theta_{ij} & \cos\theta_{ij} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.42}$$

The second rotation is by a value ψ_{ij} about the intermediate $\hat{\mathbf{e}}_{S1}$ direction, where $\psi_{ij} = \arctan\left(\frac{\eta_i'}{\lambda}\right)$, and η_i' is the second component of the rotated pixel positions $\mathbf{R}_\theta \mathbf{p}_i = [\xi_i' \ \eta_i' \ \lambda]^T$, $\mathbf{R}_\theta \mathbf{p}_j = [\xi_j' \ \eta_j' \ \lambda]^T$, and $\eta_i' = \eta_j'$. This second rotation matrix $\mathbf{R}_{\psi_{ij}} \in SO(3)$ is defined as,

$$\mathbf{R}_{\psi_{ij}} \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi_{ij} & \sin\psi_{ij} \\ 0 & -\sin\psi_{ij} & \cos\psi_{ij} \end{bmatrix} \tag{2.43}$$

The reason this second rotation is necessary will become clear in the following section, as it drastically simplifies analytic computations used to determine the out-of-plane rotation angle. The final link rotation has value ϕ_{ij} about the intermediate $\hat{\mathbf{e}}_{S2}$ direction as shown in Fig. 2.21. Unlike the first and second rotations θ_{ij}, the third rotation ϕ_{ij} is not directly measurable from the image, but it will be shown in Sec. **??** that a finite set of possible angles can be computed analytically using the projected lengths d_{ij} and the known link ratios in Eq. 2.37. The rotation matrix $\mathbf{R}_{\phi_{ij}} \in SO(3)$ is defined as

$$\mathbf{R}_{\phi_{ij}} \triangleq \begin{bmatrix} \cos\phi_{ij} & 0 & -\sin\phi_{ij} \\ 0 & 1 & 0 \\ \sin\phi_{ij} & 0 & \cos\phi_{ij} \end{bmatrix} \tag{2.44}$$

A viewpoint invariant definition of the target human pose $\mathbf{p}_T$ has already been defined in Eq. 2.40, however a second description of the target pose, expressed

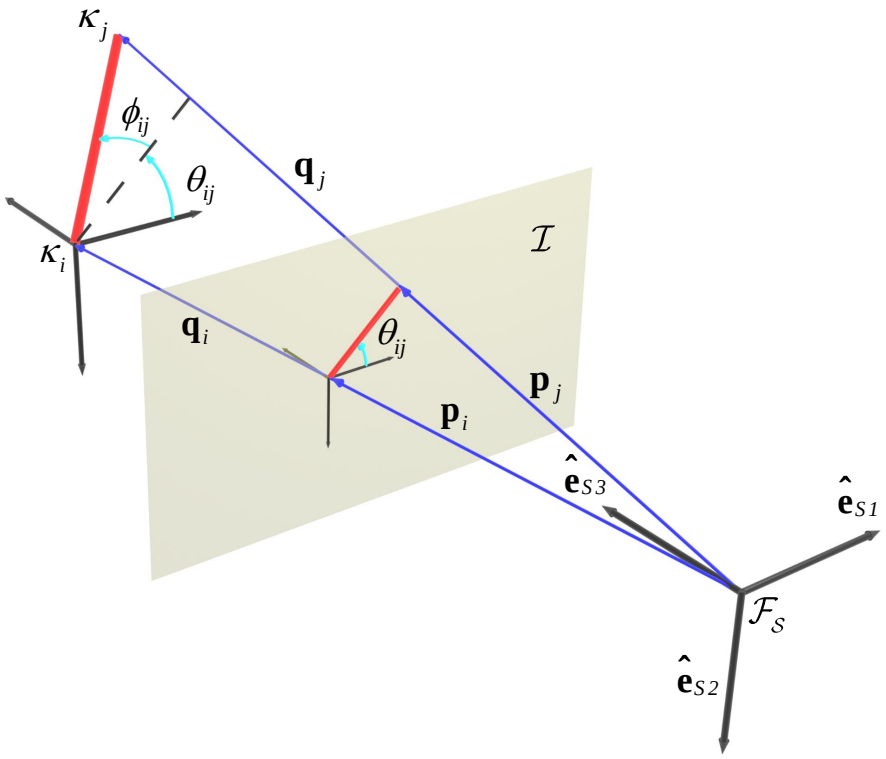


Figure 2.21: The angles $\theta_{ij},\ \phi_{ij}$ used to describe the pose of connected joints $\kappa_i,\ \kappa_j$ with respect to the camera-fixed frame $\mathcal{F}_{\mathcal{S}}$.

using rotations about the camera-fixed reference frame, is necessary for the estimation method presented in the following subsection. Thus, the target human pose with respect to the camera-fixed reference frame is defined as,

$$\begin{aligned}\mathbf{p}_S \triangleq [&\theta_{01}\ \phi_{01}\ \theta_{02}\ \phi_{02}\ \theta_{23}\ \phi_{23}\ \theta_{34}\ \phi_{34}\ldots \\ &\theta_{05}\ \phi_{05}\ \theta_{56}\ \phi_{56}\ \theta_{67}\ \phi_{67}\ldots \\ &\theta_{18}\ \phi_{18}\ \theta_{89}\ \phi_{89}\ \theta_{9,10}\ \phi_{9,10}\ldots \\ &\theta_{1,11}\ \phi_{1,11}\ \theta_{11,12}\ \phi_{11,12}\ \theta_{12,13}\ \phi_{12,13}]^T\end{aligned} \tag{2.45}$$

Although this representation of the target human pose is expressed in terms of angles about the camera-fixed reference frame $\mathcal{F}_{\mathcal{S}}$, the angles that are rotated out of the image plane, i.e., ϕ_{ij}, are not directly measurable from the image plane coordinates $\mathbf{p}_{ij}$. In fact, the only useful measurements that are directly available

from the image plane measurements are the in-plane rotation θ_{ij} and projected link lengths d_{ij}, as expressed in Eq. 2.41. Then, the measurement vector $\mathbf{z}_T \in \mathbb{R}^{28}$, composed of the projected root joint position ξ_0, η_0, the projected torso length d_0, the in-plane link rotations θ_{ij}, and the projected link lengths d_{ij}, that are directly available from the image plane measurements is defined as,

$$\mathbf{z}_T \triangleq [\xi_0 \ \eta_0 \ d_0 \ \theta_{01} \cdots d_{02} \cdots]^T \tag{2.46}$$

In order to make the estimation of the human pose $\mathbf{p}_S$ geometrically possible, a reasonable assumption used in this work is that the human target's torso is parallel to the image plane, i.e., $\phi_{01} = 0$. Ideally, the torso is always vertical to avoid errors but the proposed method is capable of handling non-vertical torso pose, as long as the torso is parallel to the image plane. This assumption leads to the following relationship,

$$d_{01} = l\frac{\lambda}{z_0} \tag{2.47}$$

Until this point, all quantities have been described for a fixed point in time with no motion. However, this dissertation assumes the camera is onboard a mobile quadrotor which may or may not be moving, and the human target may or may not be moving as well and therefore the target human pose may be changing in time. The camera is assumed fixed with respect to the quadrotor, and therefore the position vector $\mathbf{r}_S$ describes the position of the quadrotor and the camera focal point with respect to the fixed reference frame $\mathcal{F}_\mathcal{W}$. The camera begins taking measurements at initial time t_0 and a new measurement vector becomes available at a fixed sampling interval $\Delta t > 0$. Then, at each time step a measurement $\mathbf{z}(t)$ is obtained and used to estimate a viewpoint invariant target human pose $\mathbf{p}_T(t)$. A viewpoint invariant pose history is then defined as the collection of estimated

poses,

$$\mathcal{P}_{t_0:t}^{T} \triangleq \{\mathbf{p}_T(t_0),\ \mathbf{p}_T(t_0+\Delta t), ..., \mathbf{p}_T(t)\} \tag{2.48}$$

In the following section a novel and systematic approach is developed that develops novel feature vector $\mathbf{v}\left(\mathcal{P}_{t_0:t}^{T}\right)$ that is a function of the target human pose history and is, by definition, viewpoint invariant. That is, the same human action viewed from multiple camera viewpoints will yield the same feature vector, and thus a consistent action classification. This property will inevitably improve action classification results over previous methods that compute features relative to a camera-fixed reference frame.

In this dissertation, the set of action classes considered for the task of action recognition are given as,

$$\mathcal{Y} \triangleq \{\text{idle},\ \text{walking},\ \text{running}\} \tag{2.49}$$

where *idle* refers to any stationary action performed by the human, such as sitting, standing, etc. The simple set of actions considered here are chosen since each action would require a controller with slightly different gains in order to minimize the tracking error.

The tracking error used in this dissertation is designed such that the human target remains in the camera FOV at a reliable scale, as projected on the image plane, in order to maintain reliable perception over long time horizons. These specifications are captured by defining the desired set point of the projected root joint, $\xi_0^*,\ \eta_0^*$ and projected torso length, d_{01}^*. Then, the tracking error at time $t \in [t_0,\ t_f)$ is defined as,

$$\mathbf{e}_T(t) \triangleq \begin{bmatrix} \xi_0^* - \xi_0(t) \\ \eta_0^* - \eta_0(t) \\ d_{01}^* - d_{01}(t) \end{bmatrix} \tag{2.50}$$

The control problem addressed in this dissertation can now be expressed as a sequence of control inputs $\mathbf{u}(t) \in \mathcal{U}$ which minimizes the tracking error $e_T(t)$ over the time interval $[t_0,\ t_f)$.

Methodology

The proposed method computes a finite set of possible out of plane link angles ϕ_{ij} for each link $(\kappa_i,\ \kappa_j)$, given a measurement $\mathbf{z}$. Then, an UKF is used to estimate the kinematics of the human to determine the correct out of plane pose from the finite analytical solutions. The resulting pose $\mathbf{p}_S$ that is output from the UKF is expressed in terms of angles relative to $\mathcal{F}_\mathcal{S}$, which is then transformed to the viewpoint invariant $\mathbf{p}_T$. A time history of viewpoint invariant human poses $\mathcal{P}^T_{t0:t}$, from initial time t_0 to the current time t, is then used to define a feature vector that captures temporal information about the human pose history that characterizes actions. Finally, the proposed feature vector is used in a SVM to recognize the current human action. A switched quadcopter controller is then designed to maintain the human in the camera's FOV by adaptively adjusting the control parameters based on the human's actions.

The human skeleton shown in Fig. 2.18 can be represented as a collection of four sets of serially connected links $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4\}$, each extending from the root node κ_0, defined as the ordered sets,

$$\mathcal{L}_1 \triangleq \{\kappa_0, \kappa_2, \kappa_3, \kappa_4\} \tag{2.51}$$

$$\mathcal{L}_2 \triangleq \{\kappa_0, \kappa_5, \kappa_6, \kappa_7\} \tag{2.52}$$

$$\mathcal{L}_3 \triangleq \{\kappa_0, \kappa_1, \kappa_8, \kappa_9, \kappa_{10}\} \tag{2.53}$$

$$\mathcal{L}_4 \triangleq \{\kappa_0, \kappa_1, \kappa_{11}, \kappa_{12}, \kappa_{13}\} \tag{2.54}$$

Using this convention, the position of the k-th joint in any of the ordered sets can be expressed as,

$$\mathbf{q}_k = \mathbf{q}_0 + \sum_{m=1}^{k} (\mathbf{q}_m - \mathbf{q}_{m-1})$$
$$\mathbf{q}_k = \mathbf{q}_0 + l \sum_{m=1}^{k} \gamma_m (\mathbf{R}_{\phi_m} \mathbf{R}_{\psi_m} \mathbf{R}_{\theta_m})^T [1\ 0\ 0]^T \tag{2.55}$$

where $\mathbf{q}_k$ has coordinates expressed with respect to $\mathcal{F}_\mathcal{S}$. Now, the out of plane pose angles ϕ_k can be expressed analytically as shown in the following theorem.

Theorem 2.7.1 *Given a measurement vector* $\mathbf{z}$*, the out of plane pose angle* ϕ_k *of the* k*-th serially connected link, originating at the root joint* κ_0*, has at most* 2^k *real solutions.*

In order to determine ϕ_k for every $k > 0$, consider the position vector of $\mathbf{q}_{k-1}$ expressed in the camera frame and rotated to be in the intermediate frame rotated by θ_k and ψ_k, defined as

$$\begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \end{bmatrix} \triangleq \mathbf{R}_{\psi_k} \mathbf{R}_{\theta_k} \mathbf{q}_{k-1} \tag{2.56}$$

In this reference frame $y_{k-1} = 0$, and the projected length d_k can be expressed as the simple scalar equation,

$$d_k = \lambda' \frac{x_k}{z_k} - \lambda' \frac{x_{k-1}}{z_{k-1}} \tag{2.57}$$
$$d_k = \lambda' \frac{x_{k-1} + l\gamma_k \cos\phi_k}{z_{k-1} - l\gamma_k \sin\phi_k} - \lambda' \frac{x_{k-1}}{z_{k-1}} \tag{2.58}$$

where $\lambda' \triangleq \frac{\lambda}{\cos\psi_k}$. Then, define the following length ratios,

$$\bar{x}_{k-1} \triangleq \frac{x_{k-1}}{l}, \quad \bar{z}_{k-1} \triangleq \frac{z_{k-1}}{l}, \quad \bar{d}_k \triangleq \frac{d_k}{\lambda'} \tag{2.59}$$

which leads to the expression

$$\bar{d}_k = \frac{\bar{x}_{k-1} + \gamma_k \cos\phi_k}{\bar{z}_{k-1} - \gamma_k \sin\phi_k} - \frac{\bar{x}_{k-1}}{\bar{z}_{k-1}} \tag{2.60}$$

Solving for ϕ_k results in the analytical solution for the out of plane pose angle ϕ_k in terms of measurement variables, for arbitrary $k > 0$,

$$\phi_k = \tan^{-1}\left(\frac{\bar{x}_{k-1}}{\bar{z}_{k-1}} + \bar{d}_k\right) \pm \cos^{-1}\left(\frac{\bar{d}_k \frac{\bar{z}_{k-1}}{\gamma_k}}{\sqrt{\left(\frac{\bar{x}_{k-1}}{\bar{z}_{k-1}} + \bar{d}_k\right)^2 + 1}}\right) \tag{2.61}$$

It is clear from Eq. 2.61 that each analytical solution for ϕ_k has at most two real solutions given all previous link pose angles. Therefore, the k-th link has at most 2^k possible out-of-plane pose angles. It is also clear that 2^k solutions exist, however some of these may be complex and are thus not of interest.

Fig. 2.22 shows an example of the analytical calculation for $k = 2$ consecutive links. The first link displays two real solutions and the second link displays two real and two partially complex solutions. The ground truth out of plane angles are also shown in the figure and one of the solutions for both links perfectly aligns with the true solution, validating the analytical solution.

This dissertation proposes the use of an UKF to determine which of the possible out-of-plane pose angle solutions is the correct, by taking into account the kinematics of the human skeleton. The most likely analytical solution of Eq. 2.61 is used to initialize the UKF by taking into account typical human joint limits and constraints.

Unscented Kalman Filter

The UKF proposed in this subsection requires a process model $\mathbf{f}_T(\cdot)$, and a measurement model $\mathbf{h}(\cdot)$, in order to estimate the unique out-of-plane angle for each

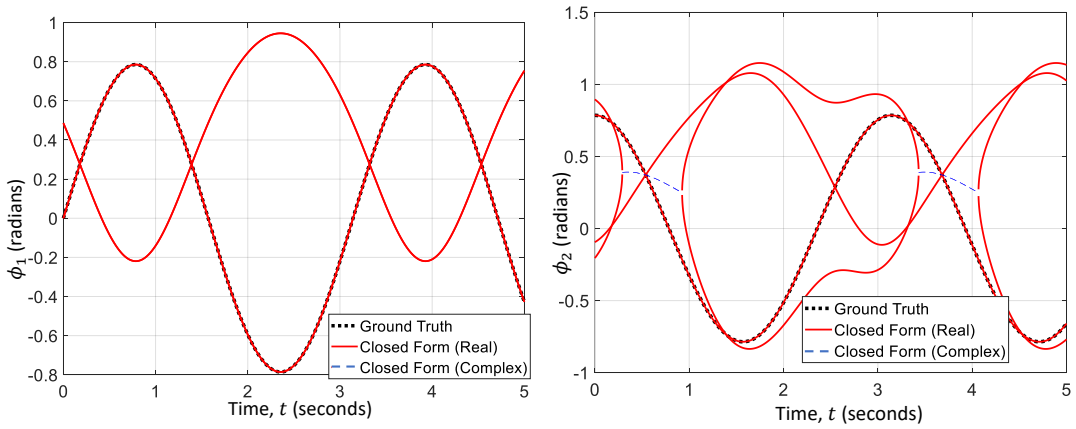


Figure 2.22: $k = 2$ link example of the analytical solutions for the out of plane link pose angles $\phi_1(t), \phi_2(t)$. The first link (left) contains two analytical solutions and the second link (right) contains four analytical solutions. One of the analytical solutions for each link perfectly aligns with the true solution.

link, ϕ_k. The target state $\mathbf{x}_T \in \mathbb{R}^{31}$ is composed of the root joint position and velocity over the torso length and the in-plane and out of plane link rotations, defined as,

$$\mathbf{x}_T = \begin{bmatrix} \frac{1}{l}\mathbf{q}_0^T & \frac{1}{l}\dot{\mathbf{q}}_0^T & \theta_0 \cdots & \dot{\theta}_0 \cdots & \phi_1 \cdots & \dot{\phi}_1 \cdots \end{bmatrix}^T \tag{2.62}$$

The target measurement vector $\mathbf{z}$ is expressed in Eq. 2.46. The measurement model $\mathbf{h}(\cdot)$ maps the target state to a target measurement with additive Gaussian noise, or,

$$\mathbf{z} = \mathbf{h}(\mathbf{x}_T) + \mathbf{w} \tag{2.63}$$

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, and $\mathbf{R} \in \mathbb{R}^{28\times 28}$ is the measurement noise covariance matrix. The measurement model is defined using the pinhole camera model. In particular, the projected root joint position is computed in terms of state variables as

$$\xi_0 = \lambda\frac{\bar{x}_0}{\bar{z}_0}, \quad \eta_0 = \lambda\frac{\bar{y}_0}{\bar{z}_0}, \quad d_0 = \lambda\frac{1}{\bar{z}_0} \tag{2.64}$$

where the components of the root joint position vector expressed in the camera-fixed reference frame are given as $\mathbf{q}_0 = l[\bar{x}_0 \ \bar{y}_0 \ \bar{z}_0]^T$. The in-plane link rotations

θ_{ij} are directly measurable since they are both members of the target state and measurement vectors. The projected link lengths are computed by making use of the ordered sets in Eq. 2.51 as

$$d_k = \|\mathbf{p}_k - \mathbf{p}_{k-1}\| \tag{2.65}$$

$$d_k = \left\| \frac{\lambda}{z_k}\mathbf{q}_k - \frac{\lambda}{z_{k-1}}\mathbf{q}_{k-1} \right\| \tag{2.66}$$

which can be expressed in terms of state variables using Eq. 2.55.

Finally, the process model $\mathbf{f}_T(\cdot)$ maps target states from one time step t to the next time step $t + \Delta t$, or,

$$\mathbf{x_T}(t + \Delta t) = \mathbf{f}_T(\mathbf{x}_T(t)) + \mathbf{v}(t), \tag{2.67}$$

where $\mathbf{v}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}(t))$, and $\mathbf{Q}(t) \in \mathbb{R}^{31\times 31}$ is the process model noise covariance matrix. Because human motion is highly unpredictable, a simple kinematic model for the human will not suffice for a reliable process model. Therefore, this work proposes to model the human motion as a Gaussian Process (GP), such that $\mathbf{f}_T(\cdot)$ and $\mathbf{Q}(t)$ are learned from data such that $\mathbf{f}_T$ is the GP mean function and $\mathbf{Q}(t)$ is the GP covariance. However, the proposed method does not require 3D pose training data. Instead the training data is taken from 2D pose measurements and the analytical function determined in the previous section is used to construct training data directly. Fig. 2.23 shows the learned function from the GP for a two-link example with the predicted out-of-plane pose angles as well as the resulting UKF prediction that accurately tracks the ground truth out of plane angle. The uncertainty in the prediction does not grow over time, which may be counter intuitive, but this is because the time is not the feature used for prediction, and only the previous state is needed.

The UKF enables the collection of the target state history $\mathbf{x}_T(t)$ for $t \in [t_0, t)$

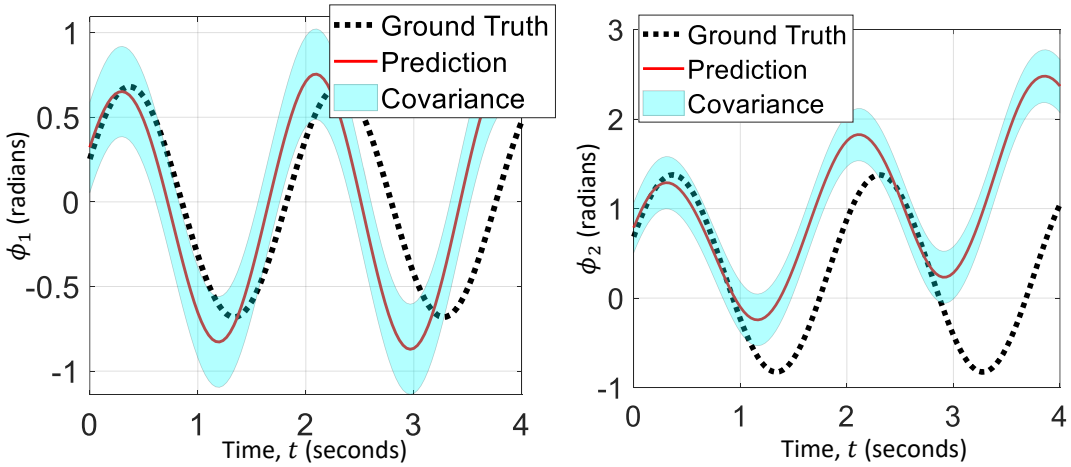


Figure 2.23: $k = 2$ link example of the GP prediction for the out of plane link pose angles $\phi_1(t), \phi_2(t)$. The shaded region around the predicted mean function represents 2 standard deviations from the mean function.

and therefore the human target pose history $\mathbf{p}_S(t)$ in the camera-fixed reference frame. The following subsection explains the methodology for transforming the target pose to a viewpoint invariant representation which is then used to construct a viewpoint invariant feature vector for action recognition.

Action Recognition and Controller Design

In order to develop a viewpoint invariant feature vector, a viewpoint invariant representation of the human pose vector $\mathbf{p}_T(t)$ is computed from the estimated human pose $\mathbf{p}_S(t)$ in the camera-fixed reference frame output of the UKF. The unit vector $\hat{\mathbf{e}}_{ij}$ along the axis of the link from κ_i to κ_j can be expressed in terms of the camera-frame pose angles as,

$$\hat{\mathbf{e}}_{ij} = (\mathbf{R}_{\phi_{ij}}\mathbf{R}_{\psi_{ij}}\mathbf{R}_{\theta_{ij}})^T \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \tag{2.68}$$

Then, the unit vectors composing the target-fixed reference frame, $(\hat{\mathbf{e}}_{T1}, \hat{\mathbf{e}}_{T2}, \hat{\mathbf{e}}_{T3})$ can be written in terms of the link unit vectors by making use of Eqs. 2.38 and 2.39. By taking the vector dot product of each of the target-frame unit vectors

with each link-axis unit vector, the link-axis unit vectors can be expressed in the target-fixed reference frame. Let the components of the link-axis unit vector be denoted by $\hat{\mathbf{e}}_{ij} = [x_{ij}\ y_{ij}\ z_{ij}]^T$. The two rotation matrices defining the pose of any link in the target-fixed reference frame are given by

$$\mathbf{R}_{\varphi_{ij}} = \begin{bmatrix} \cos\varphi_{ij} & 0 & -\sin\varphi_{ij} \\ 0 & 1 & 0 \\ \sin\varphi_{ij} & 0 & \cos\varphi_{ij} \end{bmatrix} \tag{2.69}$$

$$\mathbf{R}_{\vartheta_{ij}} = \begin{bmatrix} \cos\vartheta_{ij} & \sin\vartheta_{ij} & 0 \\ -\sin\vartheta_{ij} & \cos\vartheta_{ij} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.70}$$

Therefore, the unit vector along the axis of any link expressed in the target-fixed reference frame can be expressed as,

$$\hat{\mathbf{e}}_{ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{bmatrix} = (\mathbf{R}_{\varphi_{ij}}\mathbf{R}_{\vartheta_{ij}})^T \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\vartheta_{ij}\cos\varphi_{ij} \\ \sin\vartheta_{ij}\cos\varphi_{ij} \\ -\sin\varphi_{ij} \end{bmatrix} \tag{2.71}$$

Therefore, the viewpoint invariant pose angles can be solved for, leading to the expression,

$$\vartheta_{ij} = \arctan\left(\frac{y_{ij}}{x_{ij}}\right), \quad \varphi_{ij} = \arcsin(-z_{ij}) \tag{2.72}$$

From the viewpoint invariant pose angles that are computed at each time step for each link in the human skeleton, the target human pose history $\mathcal{P}_{t_0:t}^T$ is determined. The feature vector used in this dissertation for the task of action recognition is computed using the viewpoint invariant human pose history and is defined as,

$$\mathbf{v}(t) = \sum_{i=0}^{\tau} \mathbf{w}_i^T \mathbf{p}_T(t - i\Delta t) \tag{2.73}$$

where $\tau > 0$ is the number of frames used in the sliding temporal window such that the temporal action information is captured in the feature representation.

The controller design proposed in this work uses a simple proportional-integral-derivative (PID) control scheme with adaptive gains that depend on the human target's action. The control input $\mathbf{u}(t)$ described here is a high-level velocity command which is ultimately fed to a low-level stabilizing controller. The control input is expressed as $\mathbf{u}(t) = [u\ v\ w\ r]^T$, where u, v, w are the forward, right, and vertical velocities and r is the yaw rate about the vertical axis of the quadrotor. The control input is expressed in terms of the error vector $\mathbf{e}_T(t)$ as,

$$\mathbf{u}(t) = \mathbf{K}_p\mathbf{e}(t) + \mathbf{K}_i \int_{t_0}^{t} \mathbf{e}(\tau)d\tau + \mathbf{K}_d \frac{d\mathbf{e}(t)}{dt} \tag{2.74}$$

where the gain matrices $\mathbf{K}_p, \mathbf{K}_i, \mathbf{K}_d \in \mathbb{R}^{4\times 3}$ are dependent on the action determined from the perception algorithm presented in this dissertation. That is, the set of gain matrices $\{\mathbf{K}_p, \mathbf{K}_i, \mathbf{K}_d\}$ is selected from an *a priori* collection of gain matrices that depend on the action being performed by the human. In this dissertation, the set of actions the human may be performing are $\mathcal{Y} = \{\text{idle}, \text{walking}, \text{running}\}$ as discussed in a previous section. Then, if the action recognition results show that the action is *running*, then a set of predefined gains $\{\mathbf{K}_p, \mathbf{K}_i, \mathbf{K}_d\}$ are used in Eq. 2.74 such that the controller accurately tracks a running human at a safe distance, and similarly for the other action classes.

Physical Experiments and Demonstrations

Extensive experiments are conducted in this dissertation in both simulation and on real autonomous robotic systems that prove the validity of the theoretical contributions of this work. The quantitative results presented in the tables of this

Figure 2.24: Physical experiments of MAV event-based collision avoidance. The MAV onboard sensor (a) navigates an environment shown as a grayscale image (b) and navigates by avoiding collisions using the event-based depth estimation algorithm, visualized in (c).

dissertation have been prepared using visually and physically accurate simulation environments in order to enable repeatability and perfect control of environmental and sensor parameters. The validation of the proposed methods was done using physical experiments with multiple different quadrotors including a very small micro-aerial vehicle and a somewhat larger quadrotor capable of higher speeds.

The outdoor physical experiments were performed on Campus at Cornell University in a complex garden area with small gorges that would be impassable for humans. The garden area is also a high-traffic pedestrian area and many pedestrians are in the sensor FOV at a given time. The quadrotor used in the outdoor experiments was a DJI Mavic Pro 2.0 and the algorithms were implemented using the DJI mobile software development kit. The DJI is equipped with a 1-inch CMOS sensor that is used for computer vision and simulated event-camera data and algorithms. The onboard camera's FOV angle is approximately 77 degrees and captures video at 1080p spatial resolution at 60 fps. The indoor experiments are performed on a Crazyflie 2.0 quadrotor equipped with a small camera capable of 720p video at 30 fps.

This section demonstrates the proposed algorithms on a physical quadrotor

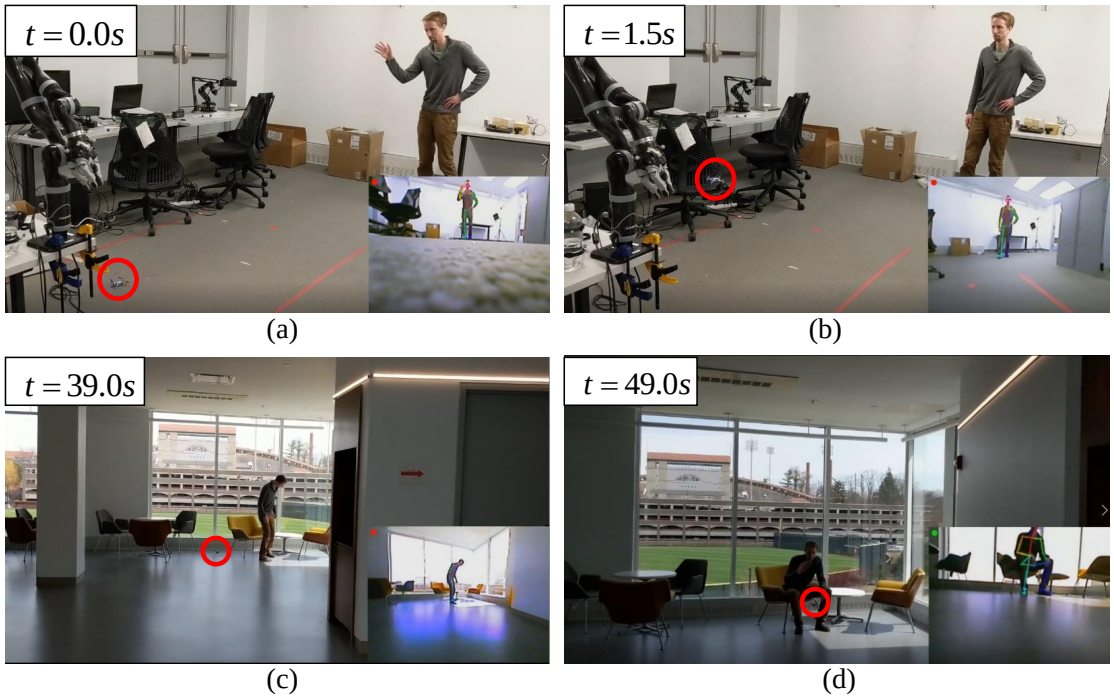


Figure 2.25: Demonstration of the MAV performing high-level autonomous ative perception using event-based algorithms. The MAV is initially stationary on the ground, having recognized the human (a), then immediately after the human waves in the sensor FOV, the quadrotor takes off and tracks the human of interest through an indoor environment (b-d).

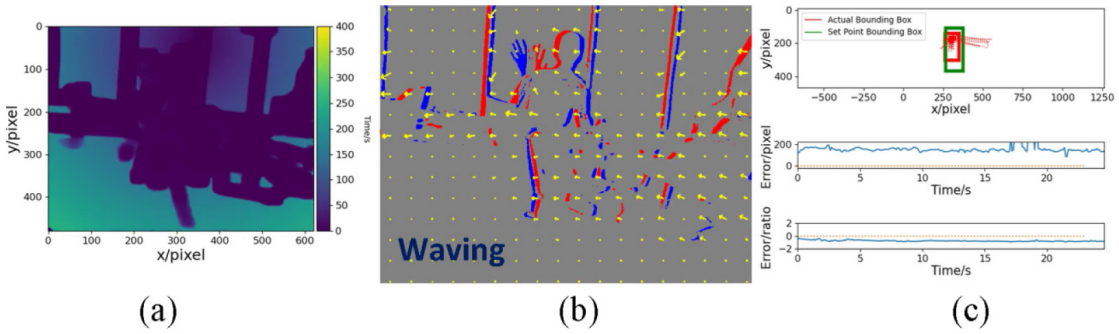


Figure 2.26: Perception in action MAV controller. While navigating an unknown environment (a), the MAV recognizes the human's waving action (b) and the on-board controller switches from following the human to landing.

interacting with one or more humans. The first set of experiments are performed on a MAV, shown in Figure 2.24, where the quadrotor autonomously navigates an environment while avoiding collisions with objects using the proposed depth estimation algorithm for event sensors. In addition to simple collision avoidance and navigation, the event-based algorithms are shown to perform as expected in the active perception scenarios involving autonomous high-level behavior around humans. For instance, the same MAV shown previously reacts to a human waving and follows the human of interest in an indoor environment, as shown in Figure 2.25 and 2.26. In addition, a similar high-level autonomous active perception demonstration is shown earlier on a larger drone capable of higher speeds. In addition to the physical experiments that validate the feasibility of the proposed event-based algorithms, a large number of simulation experiments are conducted in order to quantitatively compare the event-based algorithms with the analogous RGB camera algorithms. The simulations are conducted in a photo-realistic and physically accurate simulator.

3D Human Pose Estimation Results

The results in this section are performed using a photo-realistic rendering engine, Unreal Engine^{TM}, with Microsoft AirSim [88] to simulate a quadrotor with the onboard camera in real-time. The 2D pose measurements are obtained using OpenPose [16] and the algorithms presented in this dissertation are implemented using Python 3.6. The simulation uses a realistic city environment with variable lighting and shadows, and the quadrotor has initial position and orientation chosen such that the human is initially in the camera FOV.

The target human is programmed to walk following an arbitrary spline and the

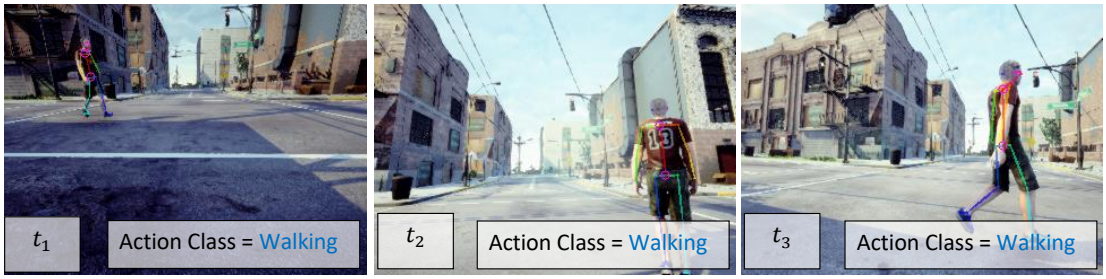


Figure 2.27: Resulting camera images at three times during an experiment where the target human was walking. The algorithm consistently correctly recognized the action and used the correct control settings to maintain the target in the camera FOV.

quadrotor successfully maintains the target in the camera FOV for the duration of the simulation. Additionally, the system developed in this dissertation correctly recognizes the human action for nearly all of the frames when a complete sliding temporal window is available. Fig. 2.27 shows a subset of representative images where the target is maintained in the FOV and the action is correctly classified.

2.8 Perception-driven Controller Design and Experiments

To demonstrate a combined implementation of the active control methods developed in this dissertation, a logical control methodology is developed in this section that combines the visual perception capabilities to inform the autonomous capabilities of an autonomous mobile quadrotor. A switching control strategy is designed to control the event-camera to execute different commands based on the perception inputs from the human target. In particular, the discrete set of high-level quadrotor commands are given as,

$$\mathcal{C} = \{\text{landing/take-off, object following, move in prescribed direction}\} \quad (2.75)$$

Other control laws can be similarly considered and integrated with the perception feedback. The perception feedback is represented by a set of semantic labels extracted by the active perception algorithms after the object of interest is identified by the object recognition algorithm. by this approach, the vehicle reacts to the behavior of one human, recognized by the onboard camera, evein in the presence of other moving objects and humans in its environment. The object recognition algorithm first detects all humans present in the camera FOV and, subsequently, assigns the 'object-of-interest' label to the human 'waving'. Other methods of identification tested in the experiments include appearance, e.g., wearing a pair of colored glasses.

The actions of the person-of-interest are then classified by the action recognition algorithm by selecting one semantic label in the perceived action set:

$$\mathcal{Z} = \{'walking', 'biking', 'pointing', 'waving'\} \tag{2.76}$$

The one-to-one mapping from the perception feedback label in $\mathcal{Z}$ to the corresponding control law in $\mathcal{C}$ at any time t is defined *a priori*, as follows:

$$f(\mathcal{Z}_i(t)) \triangleq \begin{cases} \mathcal{C}_1 & \text{if } \mathcal{Z}_i(t) = \text{'walking' or 'biking'} \\ \mathcal{C}_2 & \text{if } \mathcal{Z}_i(t) = \text{'waving'} \\ \mathcal{C}_3 & \text{if } \mathcal{Z}_i(t) = \text{'pointing'} \end{cases} \tag{2.77}$$

where $\mathcal{Z}_i(t) \in \mathcal{Z}$. Other actions, such as biking and sitting, have also been successfully tested and can be easily included by augmenting $\mathcal{Z}$ accordingly. Each vehicle action is implemented by a control law by the same index that takes as inputs all of the perception results as well as the vehicle state vector measure by onboard sensors able to provide the robot attitude and angular velocities:

$$\mathbf{u}_i(t) = \mathbf{u}_i(t, \mathbf{x}(t), \mathcal{Z}_i(t)), \text{ for } f(\mathcal{Z}_i(t)) = \mathcal{C}_i \tag{2.78}$$

The first control law, providing $\mathbf{u}_1(t) \triangleq [u_{11}(t)\ u_{12}(t)]^T$, is designed to allow the vehicle to maintain a desired distance from the person-of-interest while, simultaneously, keeping the human centered in the camera FOV. Two proportional-integral-derivative (PID) controllers are used to achieve these objectives,

$$u_{11}(t)k_{p1}(d(t)-\delta) + k_{i1}\int_{t_0}^{t}(d(\tau)-\delta)d\tau + k_{d1}\frac{d}{dt}(d(t)-\delta) \tag{2.79}$$

$$u_{12}(t)k_{p2}(j(t)-w/2) + k_{i2}\int_{t_0}^{t}(j(\tau)-w/2)d\tau + k_{d2}\frac{d}{dt}(j(t)-w/2) \tag{2.80}$$

where $d(t)$ is the relative distance between the target and camera at time t, and δ is the desired distance. Also, $j(t)$ represents the coordinates in the width direction of the centroid of the target in the image plane, w is the width of the image.

The second landing/take-off control law is designed as follows:

$$u_2(t) = \begin{cases} L(\cdot) & \text{if } d_y(t) > 0 \\ T(\cdot) & (if) d_y(t) = 0 \end{cases} \tag{2.81}$$

where $L(\cdot)$ denotes the landing function and $T(\cdot)$ denotes the launching function, both of which are embedded vehicle functions. $d_y(t)$ is the relative vertical position between the event-camera and the ground measured by onboard vehicle sensors.

The third control law is designed to allow the vehicle to follow a direction specified by the human pointing, extracted by the active perception algorithm and provided in the form of the desired Euler angle vector $\zeta_T(t) = [\phi_T(t)\ \theta_T(t)\ \psi_T(t)]^T$. Let the attitude of the vehicle-based camera obtained by the onboard sensors be represented by the Euler angle vector $\zeta(t) = [\phi(t)\ \theta(t)\ \psi(t)]^T$, where both vectors are referenced with respect to an inertial frame. Then, the third control law is obtained as follows:

$$u_3(t) = k_p\|\zeta(t)-\zeta_T(t)\| + k_i\int_{t_0}^{t}\|\zeta(t)-\zeta_T(t)\|dt + k_d\|\dot{\zeta}(t)-\dot{\zeta}_T(t)\| \tag{2.82}$$

where the gains $k_p, k_i, k_d \geq 0$ control the weight of each term in the PID controller.

2.9 Algorithm Accuracy comparison

The accuracy of the event-camera- and RGB camera-based algorithms are compared for the tasks of depth estimation, object recognition, object tracking, and action recognition, in this section. For each task, a large dataset is collected from a photo-realistic and physically accurate simulation environment in order to obtain perfect ground truth information enabling a quantitative results analysis. The simulation environment also allows for consistent and repeatable experiments in complex and widely varying scenarios, such as a forest environment or a structured city environment, while simultaneously allowing for experiments in which camera parameters, such as spatial and temporal resolution are varied. Analogous physical experiments would otherwise require many different sensors and prove infeasible. For each task a quantitative accuracy measurement is defined and the results are summarized in the set of tables in this section.

All event-based algorithms are implemented in Python 3 and optimized using Pycharm Professional profiler. The Pycharm profiler is also used to record and report computational efficiency results. The RGB camera and event-camera algorithms have been implemented using Numpy and OpenCV libraries. The quadrotors in physical experiments communicate all information in real-time to a local PC for analyzing performance using the DJI Mobile SDK for the outdoor experiments, and using the Crazyflie development platform for the indoor experiments. The simulation results are performed using Unreal Engine and Airsim for visual and physical accuracy, respectively. The assets used in the Unreal Engine environments have been obtained from the Unreal Engine marketplace and all of the algorithms are implemented in Python using Airsim's python API for communication with Unreal Engine.

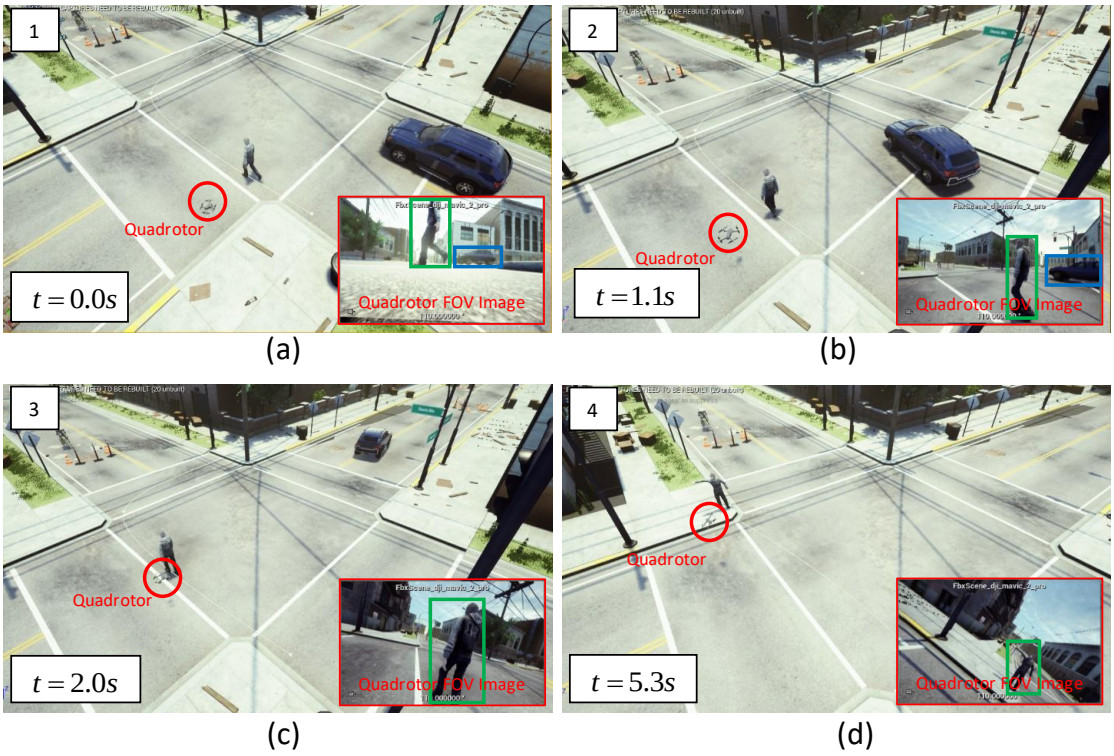


Figure 2.28: Demonstration of the event-camera active perception algorithms in a high-fidelity simulation environment. The quadrotor is initially on the ground and detects two objects that are classified as a person and a car using the event-based HOF algorithm (a). The robot begins tracking and following the human of interest as the car moves in a different direction (b). The quadrotor continues to track the human while maintaining a desired distance using the depth perception and object tracking algorithms (c). The quadrotor recognizes the human pointing and flies in the direction specified by the human (d).

2.9.1 Depth Estimation Accuracy

The accuracy of an algorithm estimating the scene depth for each pixel in the sensor image plane over a time interval is best represented using the structural similarity index that is a common performance metric for depth estimation [97]. In this article, the structural similarity index measures the similarity between the ground truth depth matrix $\mathbf{D}(t)$ and the estimated depth matrix $\hat{\mathbf{D}}(t)$ at time t, in terms of the multiplicative luminance, contrast, and structural similarity terms,

respectively.

$$s(\mathbf{D}(t), \hat{\mathbf{D}}(t)) = \left(\frac{2\mu(t)\hat{\mu}(t) + c}{\mu^2(t) + \hat{\mu}^2(t) + c}\right)\left(\frac{2\sigma(t)\hat{\sigma}(t) + c}{\sigma^2(t) + \hat{\sigma}^2(t) + c}\right)\left(\frac{\tilde{\sigma}(t) + c}{\sigma(t)\hat{\sigma}(t)}\right) \quad (2.83)$$

where depth estimate means, standard deviations, and cross covariance are defined as

$$\mu(t) = \frac{1}{hw}\sum_{d\in\mathbf{D}(t)} d, \quad \hat{\mu}(t) = \frac{1}{hw}\sum_{d\in\hat{\mathbf{D}}(t)} d \quad (2.84)$$

$$\sigma(t) = \left(\frac{1}{hw}\sum_{d\in\mathbf{D}(t)} (d - \mu(t))^2\right)^{1/2}, \quad \hat{\sigma}(t) = \left(\frac{1}{hw}\sum_{d\in\hat{\mathbf{D}}(t)} (d - \hat{\mu}(t))^2\right)^{1/2} \quad (2.85)$$

$$\tilde{\sigma}(t) = \frac{1}{hw}\sum_{d\in\mathbf{D}(t), \hat{d}\in\mathbf{D}(t)} (d - \mu(t))(\hat{d} - \hat{\mu}(t)) \quad (2.86)$$

The data used to quantify the accuracy of the depth estimation algorithms for both types of cameras is composed of five video sequences captured from a front-facing camera fixed onboard a mobile quadrotor. The difference between the various sequences is the position and orientation history of the quadrotor is varied such that some of the sequences the rotation or translational part of the total motion dominates the net motion and thus the net optical flow in the sequence. Because the depth estimation in this dissertation is primarily based on optical flow, the accuracy of the optical flow is reported as well, in order to quantify what proportion of the error in the depth estimation is caused by the optical flow algorithm rather than the depth estimation algorithm that operates on the optical flow. The optical flow accuracy is defined using the separate magnitude error and orientation error in the optical flow vector associated with each pixel over the time interval. The error in the optical flow orientation is computed using the structural similarity index. The error in the optical flow magnitude is computed as the average L2 norm of the magnitude difference between the estimated and ground truth flow magnitude

Table 2.1: Optical flow and depth estimation accuracy for event and RGB camera algorithms.

Sensor Type	Depth Estimation Accuracy	Optical Flow Accuracy	Optical Flow Magnitude Accuracy	Optical Flow Orientation Accuracy
Event-Camera	56.7%	52.8%	50.6%	55.0%
RGB Camera	65.%	70.3%	67.1%	73.5%

over all pixels in the image plane,

$$\frac{1}{hw}\sum_{\mathbf{p}\in\mathcal{I}}\|\hat{\dot{\mathbf{p}}}(t)-\dot{\mathbf{p}}(t)\| \tag{2.87}$$

Table 1 shows the accuracy of the optical flow and depth estimation algorithms computed using the aforementioned performance metrics. The measures of accuracy that are derived from the structural similarity index are linearly transformed to be in the range from zero to one, to be shown as a percentage.

2.9.2 Object Recognition Accuracy

The object recognition algorithm is analyzed using a large and diverse simulated dataset to enable perfect environmental and sensor control and allow for perfect experiment repeatability. The object recognition task considered in this dissertation uses three classes of moving objects,

$$\mathcal{Y}=\{\text{person, car, bicycle}\} \tag{2.88}$$

where each class is represented as a string of characters in the algorithm's implementation. To evaluate the object recognition accuracy, this dissertation uses the well-known classification accuracy metric known as the F1 score. To understand the F1 score, one must first understand precision and recall classification metrics.

Table 2.2: Object recognition accuracy for event and RGB camera algorithms. The classification accuracy for each moving object class is reported from a dataset composed of 18 videos composed of 4459 images.

Sensor Type	Object Recognition Accuracy	Person Recognition Accuracy	Car Recognition Accuracy	Bicycle Recognition Accuracy
Event-Camera	87.0%	91.0%	88.0%	66.0%
RGB Camera	85.0%	89.0%	80.0%	81.0%

A classifiers precision is the ratio of the number of true positives over the sum of true positives and false positives, or,

$$P = \frac{T_P}{T_P + F_P} \tag{2.89}$$

where T_P is the number of true positives classified by the model on the test data set, F_P is the number of false positives classified by the model on the test data, and P is the model's precision. Precision can be thought of as the ability of a classifier to not incorrectly label a particular class. Recall, on the other hand is the ratio of true positives over the sum of true positives and false negatives, or

$$R = \frac{T_P}{T_P + F_n} \tag{2.90}$$

where F_n is the number of false negatives classified by the model on the test data set, and R is the recall. Then, the F1 score, referred to as the object recognition accuracy in this dissertation, is the harmonic mean of the precision and recall,

$$F_1 = 2\frac{PR}{P + R} \tag{2.91}$$

Table 2 shows the results for the object recognition algorithm and the F1 score for each class and the overall accuracy. In addition, the confusion matrices for the event and RGB cameras are shown in Figure 11.

2.9.3 Object Tracking Accuracy

Object tracking consists of combined object detection and data association. The object detection method presented in this dissertation is used for both object tracking and object classification in the previous section. At time t the collection of detections $\mathcal{D}(t)$ becomes available through the detection algorithm and at time $t+\tau$, where $\tau > 0$, the set of detections $\mathcal{D}(t+\tau)$ becomes available. The detection accuracy at time t is computed using the intersection-over-union (IoU) metric that is commonly used in semantic segmentation algorithm evaluation. Denote the estimated detection of an object as the set of pixels $\hat{D}_i(t) \in \mathcal{D}(t)$ and denote the ground truth detection of an object as the set of pixels $D_i(t) \in \mathcal{D}(t)$, then the IoU performance metric is cardinality of the intersection of the estimated and ground truth detection over the cardinality of the union of the estimated and ground truth detection,

$$IoU(t) = \frac{|\hat{D}_i(t) \cap D_i(t)|}{|\hat{D}_i(t) \cup D_i(t)|} \tag{2.92}$$

Then, the IoU averaged over each true object in the FOV at every time instant when new detections become available determines the average object detection accuracy reported in Table 3. The average data association accuracy is simply the ratio of the number of times an object's identity was incorrectly associate over the total number of objects in the sensor FOV for each time instant when new detections become available. The overall tracking accuracy is then the average of the object detection and data association accuracies. Table 3 summarizes the object tracking accuracy for the event and RGB cameras.

Table 2.3: Object tracking accuracy for event and RGB camera algorithms. The detection accuracy is measured using the intersection over union metric. The data association measures the number of times the correct ID label is associated between consecutive time instants.

Sensor Type	Overall Object Tracking Accuracy	Average Data Association Accuracy	Average Object Detection Accuracy
Event-Camera	77.1%	99.0%	55.2%
RGB Camera	70.4%	97.0%	43.8%

2.9.4 Action Recognition Accuracy

The action recognition algorithms presented in this dissertation are evaluated in a similar fashion to the object classification method using the F1 score. The set of action classes investigated in this dissertation are,

$$\mathcal{Z} = \{\text{pointing, waving, walking, running}\} \tag{2.93}$$

where each action is represented as a string in the algorithm implementation. The tests are conducted using simulated data in order to obtain perfect ground truth while simultaneously having perfect control over all environmental and sensor parameters and resolutions. The action recognition model is trained using 10 training videos for each action containing the simulated human subject at variable scales and viewpoint angles. The trained model is then tested on 5 different testing videos for each action. The action recognition accuracy represented as the F1 score for each class are averaged to determine the overall action recognition accuracy is shown in Table 4.

Table 2.4: Action recognition accuracy for event and RGB cameras.

Sensor Type	Action Recognition Accuracy	Pointing Recognition Accuracy	Waving Recognition Accuracy	Walking Recognition Accuracy	Running Recognition Accuracy
Event-Camera	62.5%	80.0%	60.0%	30.0%	80.0%
RGB Camera	72.5%	50.0%	90.0%	50.0%	100.0%

2.10 Algorithm computational cost comparison

One of the main hypotheses for event-camera value to active perception systems is that they will provide significant computational savings, given their biologically inspired design and processing techniques. This section quantitatively analyzes the computational cost associated with each active perception algorithm in a rigorous manner to enable realization of the true benefits and limitations of event-cameras over RGB cameras for active perception tasks.

2.10.1 Depth Estimation Computational Cost

The depth estimation computational cost is evaluated on multiple video sequences with a stationary environment and a moving sensor onboard a mobile quadrotor. The event-camera sensor measurement is simulated using a video collected at 300 fps resulting in very high temporal resolution of the event sensor. The RGB camera temporal resolution is 30 fps by sampling every 10th image of the 300 fps data. In addition to the temporal resolution, the spatial resolution of each sensor type is varied in this task, which would not be feasible for a physical experiment. For depth estimation, a low spatial resolution may meet the requirements for active perception purposes, such as navigation and collision avoidance, although the high

Table 2.5: Depth estimation computational efficiency for event and RGB cameras.

Sensor Type	Mean Run Time (ms)	STD Run Time (ms)	Mean Memory Usage (Mb)	STD Memory Usage (Mb)
Event-Camera	171.6	1.6	5.3	0.1
RGB Camera	3050.1	26.0	398.1	0.0

resolution may be beneficial and provide some benefits. The computational tradeoff of the different resolutions is also investigated for this purpose and the results of both high and low spatial resolution for both sensors is shown in Table 5.

2.10.2 Object Recognition Computational Cost

The object recognition computational cost is evaluated on the same dataset used for the accuracy performance comparison of 18 videos and over 4000 images sampled at 300 fps. The RGB camera is simulated at 30 fps by only sampling every 10th image in a sequence, while the event-camera uses the full 300 fps resolution video to simulate high temporal frequency events with a sampling resolution that is actually more resolved than 300 fps. Although the event-camera has a high temporal frequency, the algorithm requires a significant amount of time to generate a quality optical flow estimate to construct an object representation using HOF features. Although, a significant amount of time is required for the event-camera, there is actually less memory usage since the event data structure is generally smaller than a RGB camera data structure, since the memory requirement is dependent on the number of events, or how active the visual scene is. Table 6 summarizes the run time and memory requirements for both sensor type algorithms.

Table 2.6: Object recognition computational efficiency for event and RGB cameras.

Sensor Type	Mean Run Time (ms)	STD Run Time (ms)	Mean Memory Usage (Mb)	STD Memory Usage (Mb)
Event-Camera	139.5	11.9	1.3	0.7
RGB Camera	104.6	12.3	16.6	0.0

Table 2.7: Object tracking computational efficiency for event and RGB cameras.

Sensor Type	Mean Run Time (ms)	STD Run Time (ms)	Mean Memory Usage (Mb)	STD Memory Usage (Mb)
Event-Camera	150.4	60.1	4.5	0.8
RGB Camera	845.1	29.0	398.1	0.0

2.10.3 Object Tracking Computational Cost

The computational cost for the object tracking task is evaluated on a set of scenarios where a quadrotor is tasked with following a person of interest. The computation time and memory required for each sensor type to correctly detect and associate the detection accurately is shown in Table 7. The event-camera receives events at a much higher frequency than the RGB camera samples images and can therefore achieve similar performance to the RGB camera in significantly less time and memory. This advantage is caused by the high temporal resolution of the event-camera.

2.10.4 Action Recognition Computational Cost

The computational cost for the action recognition task is evaluated on a video clip that is 3.2 seconds in length and the RGB camera frame rate is 30 fps. The

Table 2.8: Action recognition computational efficiency for event and RGB cameras.

Sensor Type	Mean Run Time (ms)	STD Run Time (ms)	Mean Memory Usage (Mb)	STD Memory Usage (Mb)
Event-Camera	81.1	27.1	16.7	4.2
RGB Camera	234.6	1.7	134.1	60.3

event-camera uses simulated events from a high-speed version of the same video sequence collected at 300 fps, although the simulated events can have finer temporal resolution than 300 fps, which leads to similar performance to real event-camera hardware. The RGB camera image resolution is 1080p, which is typical for modern cameras onboard mobile robots, while the event-camera's image resolution is much lower at 328x512, which is a common value among current event-cameras. Table 8 shows the computational performance of the two algorithms over the same time horizon required to accurately determine the action in the sequence with action recognition accuracy of very similar performance between the two sensors.

2.11 Discussion

Overall, the accuracy of the event-camera and RGB camera are comparable, with only small differences in the overall performance of each task. However, we acknowledge that more sophisticated RGB camera algorithms exist can increase the RGB camera performance significantly at the cost of adding computational expense. The RGB camera algorithms chosen for comparison in this dissertation are designed to be computationally efficient and, even so, the event-camera algorithms are often more computationally efficient. The cases in which the event-camera algorithm performs slower than the RGB camera algorithm is often due to the lack

of availability of sophisticated and optimized implementations of various low-level operations. We acknowledge also that RGB cameras are capable of significantly higher spatial resolution and can therefore infer more information at small scales, such as objects that are far away from the sensor and small motions in the FOV. In the end, the main tradeoff between RGB and event-cameras is dependent on the task at hand. That is, if the event would benefit from high spatial resolution, such as object recognition or action recognition, a RGB camera may be preferable over an event-camera, and for tasks that would benefit from high temporal resolution, such as depth estimation and object tracking, an event-camera may be preferable. In addition, for extremely resource constrained platforms an event-camera will likely be beneficial, especially with the increase of interest in development of event-camera software and optimized algorithms become widely available as is currently the case for RGB cameras.

This article has presented an active perception framework for event-cameras and compared the accuracy and computational efficiency performance quantitatively and qualitatively between the event-camera sensor with RGB camera sensor methods. A main difference between the RGB camera and event-camera methods is the temporal resolution of the event-camera is significantly higher while using significantly less computational resources. The RGB camera algorithms executed in this article are tested at a 30 fps sampling rate and the effective frame rate used to simulate the event-camera algorithms is 300 fps. Despite this, the event-camera uses significantly less computational resources than the RGB camera. If a RGB camera were to increase it's sampling frequency to 300 fps, the amount of runtime and memory required would be prohibitive, while the event-camera can maintain performance around that of the RGB camera at the high sampling rates. However, the event-camera does not have the spatial resolution of a RGB camera and

therefore cannot perform well for objects that cannot be resolved in the low spatial resolution of the event-cameras. In addition, the lack of readily available intensity information from the event-camera results in poor performance on some recognition tasks when compared to the RGB camera. In the end, the event-camera tends to outperform the RGB camera at tasks in which a high temporal resolution is valuable such as object tracking and depth estimation.

CHAPTER 3

OCCLUSION AVOIDANCE

3.1 Background and Problem Formulation

This dissertation presents a novel and systematic approach to planning the minimum-length path for a mobile robot equipped with a directional sensor tasked with viewing multiple targets in an obstacle-populated workspace. Previous sensor path planning methods developed for information-driven sensor strategy planning make use of the sensor configuration space by developing C-targets [14, 62, 63, 75, 92, 100, 101, 103]. These methods have been proven to be incredibly effective for applications in which the sensor measurement process is the primary concern of the path planning method. Therefore, this work extends these works to incorporate the bounded directional sensor FOV and LOS visibility constraint in the presence of opaque obstacles. This dissertation extends the notion of the C-target to include the LOS visibility constraint as well as multiple target visibility from a single configuration. The new subsets of the sensor configuration that are derived analytically are coined *visibility regions*. The visibility regions are used to construct a connectivity graph that approximates the connectivity of the sensor's free configuration space. The connectivity graph contains nodes that are systematically chosen using the visibility regions to avoid an unnecessary large number of nodes required to determine a high quality path. In addition, a pruning approach is developed to further reduce the computational complexity of the method while maintaining the high quality node selection and ultimately a high quality path to be produced.

The directional sensor planning method developed in the Methodology sec-

tion is demonstrated through myriad simulations as well as physical experiments. The proposed method and the computationally improved version of the proposed method are compared to a cell decomposition approach [14, 62, 103], an ad-hoc TSP approach [2], and a coverage approach [22]. The results show the proposed method is capable of generating shorter paths than all of the comparison methods while always viewing at least all of the targets viewed by the comparison methods. In some cases a target is impossible to view in which all cases fail, however, in other cases the comparison methods may miss some targets, while the proposed method will always view all of the targets that are possible to view from the given initial configuration and workspace geometry.

3.2 Problem Formulation and Assumptions

This dissertation considers the problem of planning an efficient and effective path for a mobile robotic sensor deployed to classify multiple targets in an obstacle-populated environment. The sensor workspace is a closed and compact subset of a two-dimensional Euclidean space, $\mathcal{W} \subset \mathbb{R}^2$, populated with M known and stationary convex polygonal obstacles $\mathcal{B}_j \subset \mathcal{W}$, indexed by $j \in \mathcal{J}$, where $\mathcal{J} = \{1, ..., M\}$ is the index set used to enumerate the obstacles. The sensor is tasked with observing N targets at known positions, $\mathbf{x}_i \in \mathcal{W}$, assumed stationary and modeled by a point, where $i \in \mathcal{I}$, and $\mathcal{I} = \{1, ..., N\}$ is the index set used to enumerate the targets. Each target position may represent the centroid or similar representative point in the target geometry.

In this dissertation, the sensor is assumed to be installed on a mobile robotic platform, such as a ground, aerial, or underwater robot. As in robot path planning

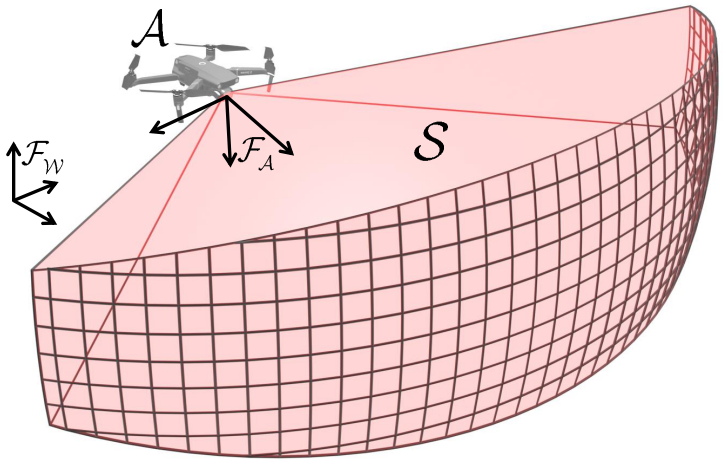


Figure 3.1: The robot geometry $\mathcal{A}$ and sensor FOV $\mathcal{S}$ in a 3-dimensional workspace with body reference frame $\mathcal{F}_{\mathcal{A}}$ that is embedded in $\mathcal{A}$ and whose origin is at the apex of $\mathcal{S}$.

methods, this dissertation assumes the robot is a rigid object with geometry $\mathcal{A}$ that is a compact subset of the workspace $\mathcal{W}$, i.e., $\mathcal{A} \subset \mathcal{W}$. Then, the position and relative orientation of every point in $\mathcal{A}$ can be represented by the position and orientation of a Cartesian body-frame $\mathcal{F}_{\mathcal{A}}$ that is embedded in $\mathcal{A}$ with origin $\mathcal{O}_{\mathcal{A}}$, relative to a fixed Cartesian frame $\mathcal{F}_{\mathcal{W}}$ that is embedded in $\mathcal{W}$ with origin $\mathcal{O}_{\mathcal{W}}$. Fig. 3.1 shows the robot geometry $\mathcal{A}$ with embedded reference frame $\mathcal{F}_{\mathcal{A}}$ relative to the inertial frame $\mathcal{F}_{\mathcal{W}}$.

For simplicity, the sensor is assumed fixed with respect to the robot such that the sensor position $\mathbf{s} \in \mathcal{W}$ and orientation $\theta \in \mathbb{S}^1$ are represented by the robot body-frame $\mathcal{F}_{\mathcal{A}}$. Throughout the remainder of this dissertation, the robot configuration $\mathbf{q}$ is defined as the augmented vector of the sensor position $\mathbf{s}$ and orientation θ, i.e., $\mathbf{q} \triangleq [\mathbf{s}^T \quad \theta]^T$. The sensor is directional and, therefore it has a preferred sensing direction and is characterized by line-of-sight visibility (LOS). Examples of robots that use directional sensors are ground or aerial robots equipped with monocular or stereo cameras, or an underwater robot equipped with an onboard synthetic aperture sonar.

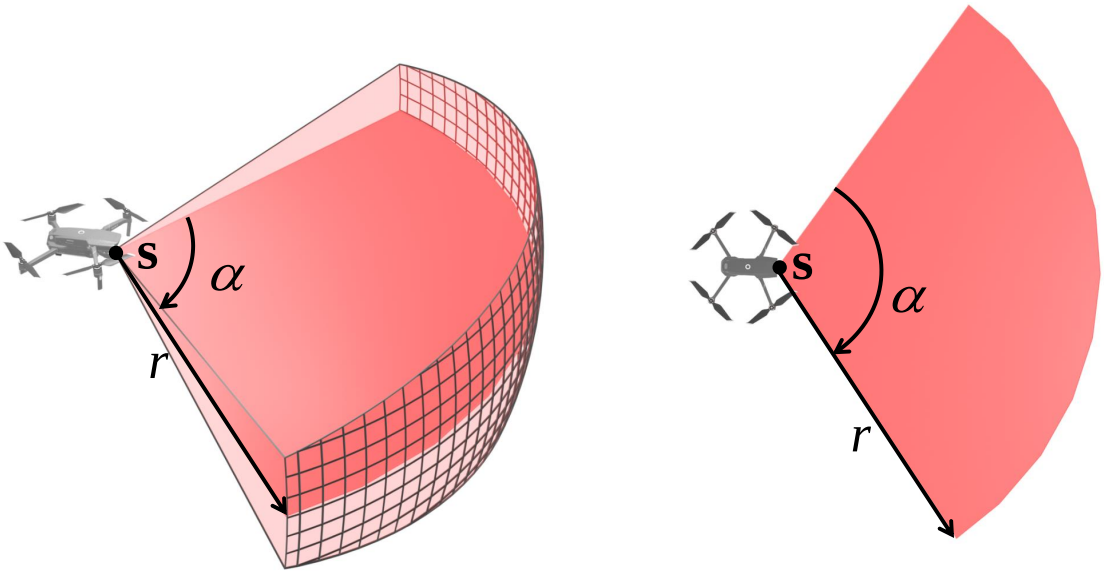


Figure 3.2: 2-dimensional parameterization of the sensor FOV $\mathcal{S}$, with apex $\mathbf{s} \in \mathcal{W}$, characterized by the opening angle $\alpha \in \mathbb{S}^1$ and maximum sensing range $r > 0$.

Definition 3.2.1 (Field of View) *For a sensor characterized by configuration,* $\mathbf{q} = [\mathbf{s}^T \quad \theta]^T$*, the sensor FOV is a rigid object that can be described by a closed and bounded subset* $\mathcal{S}(\mathbf{q}) \subset \mathcal{W}$ *in which the sensor can obtain a measurement of any target,* $\mathbf{x}$*, such that* $\mathbf{x} \in \mathcal{S}(\mathbf{q})$*.*

In this dissertation the sensor FOV $\mathcal{S}$ is modeled by a sector with opening angle $\alpha \in [0, 2\pi)$ and radius $r > 0$ which correspond to the sensor's aperture angle and maximum sensing range, respectively, and the sector apex coincides with the sensor position, $\mathbf{s} \in \mathcal{W}$, as shown in Fig. 3.2.

Target visibility requires LOS visibility, in addition to the target being within the sensor FOV, to prevent occlusions induced by opaque obstacles $\mathcal{B}_j$. A target, $\mathbf{x}_i$ is in a directional sensor's LOS if the straight line segment between $\mathbf{s}$ and $\mathbf{x}_i$ does not intersect any obstacle regions. Fig. 3.3 illustrates an example of target that is occluded by an obstacle.

Definition 3.2.2 (Line of Sight) *Given obstacles* $\mathcal{B}_j \subset \mathcal{W}, \quad j = 1, ..., M$, *a target at* $\mathbf{x} \in \mathcal{W}$ *is in the line-of-sight of a sensor at* $\mathbf{s} \in \mathcal{W}$ *if and only if*

$$L(\mathbf{s}, \mathbf{x}) \cap \mathcal{B}_j = \emptyset, \quad \forall j \in \mathcal{J} \tag{3.1}$$

where $L(\mathbf{s}, \mathbf{x}) \triangleq \{(1-\gamma)\mathbf{s} + \gamma\mathbf{x} \mid \gamma \in [0, 1]\}$ *is a line segment from the sensor position* $\mathbf{s}$ *to the target position* $\mathbf{x}$.

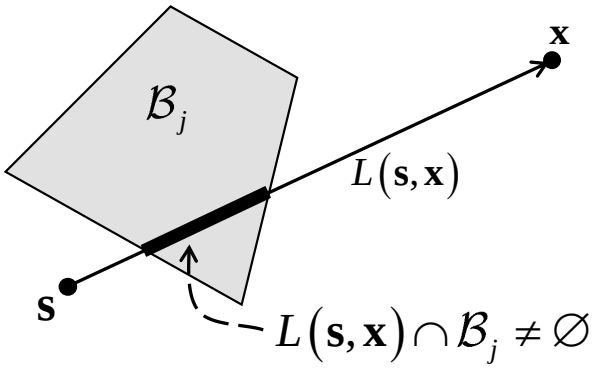


Figure 3.3: Example of an obstacle $\mathcal{B}_j$ occluding a directional sensor's LOS to a target at position $\mathbf{x}_i$ from the sensor's position $\mathbf{s}$.

A target $\mathbf{x}$ is said to be visible if and only if $\mathbf{x} \in \mathcal{S}$ and $L(\mathbf{s}, \mathbf{x}) \cap \mathcal{B}_j = \emptyset$ for all $j \in \mathcal{J}$. That is, the target must be in the sensor FOV and the sensor LOS, simultaneously. Fig. 3.4 shows a target that is visible by satisfying both the FOV and LOS conditions.

The configuration space for the robot $\mathcal{A}$ and sensor $\mathcal{S}$ is the space $\mathcal{C} \cong SE(2)$ of all configurations $\mathbf{q} \in \mathcal{C}$ of $\mathcal{A}$ and $\mathcal{S}$ with respect to $\mathcal{F}_\mathcal{W}$. As a result, the subsets of $\mathcal{W}$ occupied by $\mathcal{A}$ and $\mathcal{S}$ are both functions of the configuration vector $\mathbf{q}$, and are denoted by $\mathcal{A}(\mathbf{q})$ and $\mathcal{S}(\mathbf{q})$, respectively. This representation enables path planning for simultaneous robot collision avoidance and obtaining sensor measurements. The sensor's configuration space $\mathcal{C}$ is homeomorphic to the Special Euclidean group, $SE(2)$, meaning the configuration space can be represented as any space with

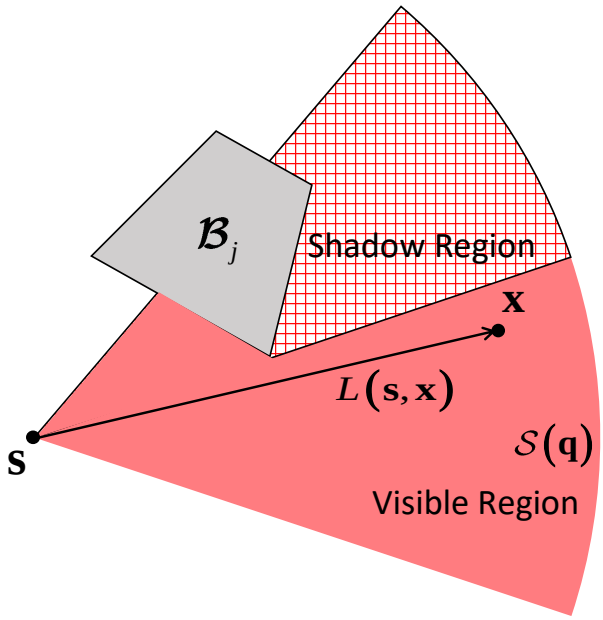


Figure 3.4: Example of a visible target, $\mathbf{x}$, in the presence of an obstacle $\mathcal{B}_j$ that occludes a portion of the sensor FOV $\mathcal{S}$.

the same topological properties as $SE(2)$ [55]. Thus, the configuration space is equivalently represented as $\mathcal{C} \cong \mathcal{W} \times \mathbb{S}^1$, and because $\mathbb{S}^1$ is 2π-periodic and multiply connected, the configuration space is also 2π-periodic and multiply connected in the direction of the sensor rotation angle.

The set of all configurations that induce a collision between the robot $\mathcal{A}$ and an obstacle $\mathcal{B}_j$ is referred to as a C-obstacle, denoted by $\mathcal{CB}_j$, [54], [55]. Obstacles in the workspace, $\mathcal{B}_j \subset \mathcal{W}$, are mapped to C-obstacles in the robot's configuration space $\mathcal{CB}_j \subset \mathcal{C}$, defined as $\mathcal{CB}_j \triangleq \{\mathbf{q} \in \mathcal{C} \mid \mathcal{B}_j \cap \mathcal{A}(\mathbf{q}) \neq \emptyset\}$. Then, the union of all C-obstacles is used to define the free configuration space, $\mathcal{C}_{free} \triangleq \mathcal{C} \setminus \bigcup_{j=1}^{M} \mathcal{CB}_j$, which represents the subset of $\mathcal{C}$ comprised of collision-free configurations. The dual representation of the C-obstacle is the C-target which represents the set of all configurations where a measurement of target $\mathbf{x}_i$ is available, denoted by $\mathcal{CT}_i$ [14] [92]. A target in the workspace $\mathbf{x}_i \in \mathcal{W}$ is mapped to a C-target in the sensor's configuration space, $\mathcal{CT}_i \subset \mathcal{C}$, defined as $\mathcal{CT}_i = \{\mathbf{q} \in \mathcal{C} \mid \mathbf{x}_i \in \mathcal{S}(\mathbf{q})\}$.

The topology of $\mathcal{C}$ can be induced by the distance metric

$$D(\mathbf{q}, \mathbf{q}') \triangleq \left((\mathbf{q} - \mathbf{q}')^T \mathbf{W} (\mathbf{q} - \mathbf{q}')\right)^{1/2} \tag{3.2}$$

where

$$\mathbf{W} \triangleq \begin{bmatrix} w_t & 0 & 0 \\ 0 & w_t & 0 \\ 0 & 0 & w_r \end{bmatrix}$$

and $w_t > 0$, $w_r \geq 0$ is a translational and rotational weight, respectively. Note that when $w_t = w_r = 1$, the distance metric in (3.2) reduces to the Euclidian metric in $\mathbb{R}^3$. Having defined the topology of $\mathcal{C}$ the collision-free sensor path from an initial to a final configuration, denoted by $\mathbf{q}_0$ and $\mathbf{q}_f$, respectively, is obtained from the classic definition of a path in a topological space [11], as follows:

Definition 3.2.3 (Path) *A path from* $\mathbf{q}_0$ *to* $\mathbf{q}_f$ *is a continuous map,*

$$\tau : [0, 1] \to \mathcal{C}_{free}$$

with $\tau(0) = \mathbf{q}_0$ *and* $\tau(1) = \mathbf{q}_f$.

In general, the sensor motion is governed by an ordinary differential equation which models the robot kinematics as,

$$\dot{\mathbf{q}}(t) = \mathbf{f}[\mathbf{q}(t), \mathbf{u}(t), t], \quad \mathbf{q}(t_0) = \mathbf{q}_0, \tag{3.3}$$

where $\mathbf{u}(t) \in \mathcal{U}$ is the control vector, and $\mathcal{U} \subset \mathbb{R}^m$ is the m-dimensional space of admissible control inputs. The vector function $\mathbf{f}(\cdot)$ is an accurate model of the robotic platform kinematics over time interval $t \in [t_0, t_f]$, where $\mathbf{q}_0$ is the initial configuration at time t_0, and $\mathbf{q}_f = \mathbf{q}(t_f)$ is the final configuration at time t_f. For simplicity, the kinematic model is chosen to be equal to the control vector, i.e., $\mathbf{f}[\mathbf{q}(t), \mathbf{u}(t), t] = \mathbf{u}(t)$.

In this dissertation, all paths produced by the proposed and comparison algorithms can be described as a continuous piecewise-linear path, also known as a polygonal chain, with vertices $(\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_n)$, such that $\mathbf{q}_n = \mathbf{q}_f$. The length of a path τ that can be described as a polygonal chain is computed using the distance metric (3.2) as:

$$J(\tau) \triangleq \sum_{k=0}^{n-1} D(\mathbf{q}_k, \mathbf{q}_{k+1}) \tag{3.4}$$

Therefore, the problem of determining minimum distance path that obtains measurements of every target at at least one point along the path while simultaneously avoiding collisions between the robot and obstacles can be written as,

$$\begin{aligned} \min_{\tau} \quad & J(\tau) = \sum_{k=0}^{n-1} D(\mathbf{q}_n, \mathbf{q}_{n+1}) \\ \text{subject to} \quad & \forall i \in \mathcal{I} \quad \exists \gamma \in [0,1] \\ & \text{such that } \mathbf{x}_i \in \mathcal{S}(\tau(\gamma)) \text{ and } L(\tau(\gamma), \mathbf{x}_i) \cap \mathcal{B}_j = \emptyset, \ \forall j \in \mathcal{J} \\ & \forall \gamma \in [0,1] \quad \tau(\gamma) \in \mathcal{C}_{free} \end{aligned} \tag{3.5}$$

The following section proposes a systematic approach to determine a sensor path τ that solves the optimization problem (3.5). This dissertation extends the existing robot-sensor path planning methods by taking into account occlusions induced by obstacles and developing new subsets of the sensor's configuration space, called visibility regions. Moreover, this work quantifies the sets of all robot-sensor configurations where individual, and sets of multiple targets are visible, leading to high-quality paths that are guaranteed to avoid occlusions in target measurements.

3.3 Directional visibility region theory

This dissertation presents a novel and systematic approach to mobile sensor path planning, where the sensor is subject to LOS visibility constraints and the robot in which the sensor is embedded must simultaneously avoid collisions with obstacles in the workspace. The first contribution of this dissertation is the closed-form derivation of the novel visibility regions, which are the sets of all configurations where a target, or collection of targets, is visible according to the sensor FOV geometry and LOS visibility. In particular, two types of visibility regions are presented: first, the target visibility regions, which are the set of all sensor configurations where a target is visible; and second, the set visibility regions, which are the sets of all configurations in which a particular set of multiple targets are visible. Additionally, this dissertation uses the visibility regions to develop an approximate solution to the sensor path planning problem in Eq. (3.5) by determining a graphical representation of the sensor configuration space that is feasible to search using a state of the art graph search algorithm.

Visibility regions are sets of sensor configurations that enable target observation by avoiding occlusions. As a first step, target visibility regions, which account for LOS visibility for a particular target, are determined by removing configurations from the C-target where the target is occluded by one or more obstacles. Then, the set visibility regions, or subsets of the configuration space from which multiple targets are visible, are computed by intersecting appropriate target visibility regions.

Definition 3.3.1 (Target Visibility Region) *Given a target at* $\mathbf{x}_i \in \mathcal{W}$*, a sensor at* $\mathbf{s} \in \mathcal{W}$ *with FOV geometry* $\mathcal{S}$*, and occluding obstacles* $\mathcal{B}_j,\ j = 1, ..., M,$

and define the line segment $L(\mathbf{s}, \mathbf{x}) \triangleq \{(1-\gamma)\mathbf{s} + \gamma\mathbf{x} \mid \gamma \in [0,1]\}$*, then, the target visibility region,* $\mathcal{PV}_i$*, is the subset of* $\mathcal{C}_{free}$ *that simultaneously satisfies the FOV and LOS target visibility conditions for one target* $\mathbf{x}_i$*,*

$$\begin{aligned} \mathcal{PV}_i \triangleq &\{\mathbf{q} \in \mathcal{C}_{free} \mid \mathbf{x}_i \in \mathcal{S}(\mathbf{q}), \\ &L(\mathbf{s}, \mathbf{x}_i) \cap \mathcal{B}_j = \emptyset, \ \forall j \in \mathcal{J}\} \end{aligned} \tag{3.6}$$

The collection of target visibility regions may not be mutually disjoint, and there may be regions in $\mathcal{C}_{free}$ that enable observations from multiple targets. Let the index set $P \subseteq \mathcal{I}$ denote the indices of one or more targets. Set visibility regions are used to describe configurations associated with a set of targets, as follows:

Definition 3.3.2 (Set Visibility Region) *Given a set of targets at* $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, ..., \mathbf{x}_{i_n}$*, where* $i_1, ..., i_n \in P$ *and* $P \subseteq \mathcal{I}$*, along with a sensor at* $\mathbf{s} \in \mathcal{W}$ *with FOV geometry* $\mathcal{S}$*, and occluding obstacles* $\mathcal{B}_j$*,* $j = 1, ..., M$*, and define the line segment* $L(\mathbf{s}, \mathbf{x}) \triangleq \{(1-\gamma)\mathbf{s} + \gamma\mathbf{x} \mid \gamma \in [0,1]\}$*, then, the set visibility region* $\mathcal{UV}_P$ *is the set of all configurations in* $\mathcal{C}_{free}$ *where each target with index in* P *is visible, and any target with index not in* P *is not visible, defined as:*

$$\begin{aligned} \mathcal{UV}_P \triangleq &\{\mathbf{q} \in \mathcal{C}_{free} \mid \mathbf{x}_i \in \mathcal{S}(\mathbf{q}), \\ &L(\mathbf{s}, \mathbf{x}_i) \cap \mathcal{B}_j = \emptyset, \ \forall i \in P, \ P \in \mathcal{P}, \ \forall j \in \mathcal{J}\} \end{aligned} \tag{3.7}$$

The closed form expressions for the target visibility and set visibility regions are obtained in terms of the convex hull and coverage cone operations. A general coverage cone is defined with respect to a point $\mathbf{a} \in \mathcal{W} \backslash \mathcal{B}$ and an arbitrary obstacle $\mathcal{B}_j \subset \mathcal{B}$,

$$\mathcal{K}(\mathbf{a}, \mathcal{B}) \triangleq \{\alpha\mathbf{x} + (1-\alpha)\mathbf{a} | \alpha \in \mathbb{R}^+, \mathbf{x} \in \mathcal{B}_j\} \tag{3.8}$$

In this dissertation, all obstacles are assumed to be convex polyhedron, which can be represented as a finite set of vertices making up the obstacle geometry. Then, for a finite set of points in $\mathcal{W}$, denoted by $A = \{\mathbf{z}_1, ..., \mathbf{z}_K\}$, the convex hull and coverage cones are defined as,

$$\text{conv}(A) \triangleq \left\{ \sum_{k=1}^{K} a_k \mathbf{z}_k \;\middle|\; \mathbf{z}_k \in A,\ a_k \geq 0,\ \sum_{k=1}^{K} a_k = 1 \right\} \tag{3.9}$$

$$\text{cone}(A, \mathbf{a}) \triangleq \left\{ \mathbf{a} + \sum_{k=1}^{K} a_k (\mathbf{z}_k - \mathbf{a}) \middle| \mathbf{z}_k \in A, a_k \geq 0 \right\} \tag{3.10}$$

respectively, where $\mathbf{a} \in \mathcal{W}$ is the cone vertex.

As a first step, the shadow region $\mathcal{D}_i \subset \mathcal{W}$ associated with a target $\mathbf{x}_i \in \mathcal{W}$ is constructed, which is the set of all positions in $\mathcal{W}$ where the target is occluded according to the LOS visibility definition (3.2.2), defined as $\mathcal{D}_i \triangleq \left\{ \mathbf{s} \in \mathcal{W} \mid L(\mathbf{s}, \mathbf{x}_i) \cap \bigcup_{j=1}^{M} \mathcal{B}_j \neq \emptyset \right\}$, where the line segment $L(\mathbf{s}, \mathbf{x}_i) \triangleq \{(1-\gamma)\mathbf{s} + \gamma \mathbf{x}_i \mid \gamma \in [0,1]\}$. The shadow region is derived in closed form using the operations (3.9) and (3.10) as:

$$\mathcal{D}_i = \bigcup_{j \in \mathcal{J}} \left(\text{cone}(\mathcal{B}_j, \mathbf{x}_i) \setminus \text{conv}(\mathcal{B}_j \cup \{\mathbf{x}_i\}) \right) \tag{3.11}$$

Fig. 3.5 shows the shadow region computed relative to the sensor position leading to the visible portion of the sensor FOV.

The shadow region, $\mathcal{D}_i \subset \mathcal{W}$, is independent of the sensor rotation, and can thus be mapped from $\mathcal{W}$ to $\mathcal{C}$ simply as $\hat{\mathcal{D}}_i \triangleq \{\mathbf{q} = [\mathbf{s}^T \quad \theta]^T \in \mathcal{C} \mid \mathbf{s} \in \mathcal{D}_i,\ \theta \in \mathbb{S}^1\}$. Then, the target visibility region with respect to $\mathbf{x}_i$ can be written in terms of the C-target $\mathcal{CT}_i$ and shadow region $\hat{\mathcal{D}}_i$ as,

$$\mathcal{PV}_i = \mathcal{CT}_i \setminus \hat{\mathcal{D}}_i \tag{3.12}$$

In practice, the rotational component of the configuration space $\theta \in [0, 2\pi)$

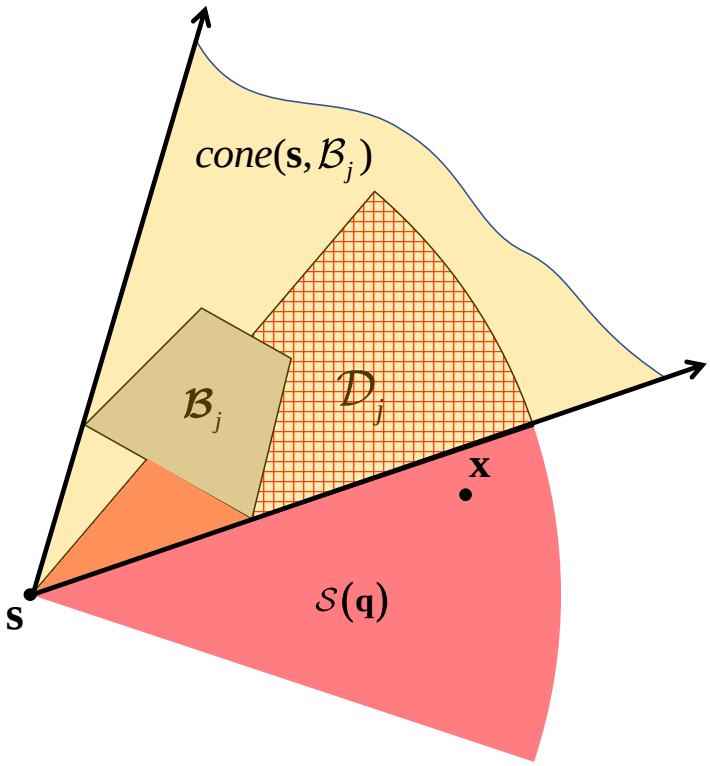


Figure 3.5: Sets used to define the shadow region and visible region of the sensor FOV with respect a single obstacle.

is discretized into $\kappa > 0$ linearly spaced intervals and the C-target and shadow region are computed as planar geometries. The target visibility regions are shown in Fig. 3.6 with a highlighted plane at a fixed rotation $\theta = \hat{\theta}$ along with a top-down view of the 2-D manifold at $\hat{\theta}$. By discretizing the rotational component of the configuration space into a collection of κ 2-D manifolds in practice, many efficient computational geometry tools can be taken advantage of, such as [70].

The set visibility region for a subset of target indices $P \subseteq \mathcal{I}$ is expressed in terms of the target visibility regions, as

$$\mathcal{UV}_P = \left(\bigcap_{i \in P} \mathcal{PV}_i \right) \setminus \left(\bigcup_{i \notin P} \mathcal{PV}_i \right) \tag{3.13}$$

Because the set visibility regions are typically concave and possibly discon-

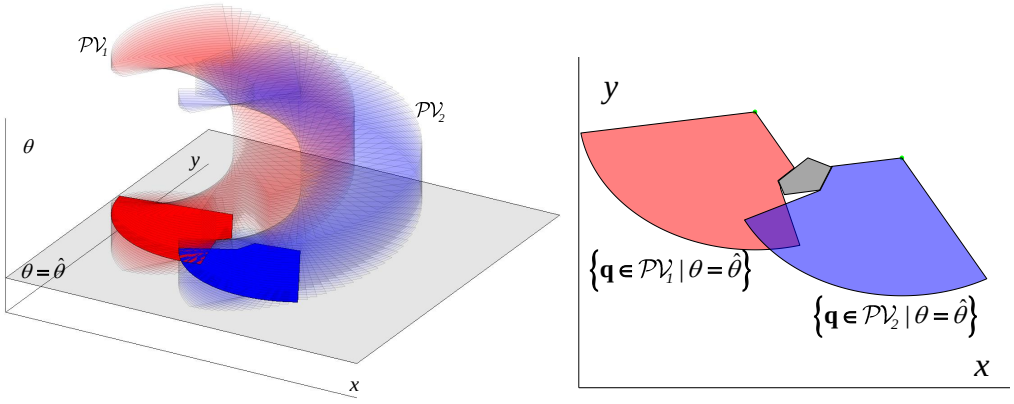


Figure 3.6: Target visibility regions $\mathcal{PV}_1,\ \mathcal{PV}_2 \subset \mathcal{C}$ and the 2-dimensional manifolds of the target visibility regions with a fixed rotation $\theta = \hat{\theta}$.

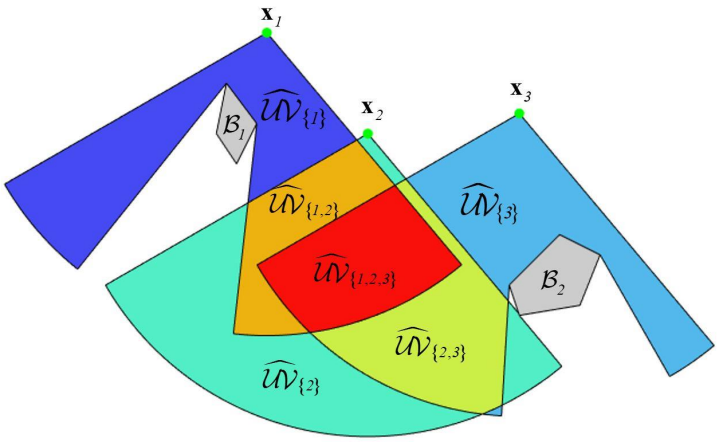


Figure 3.7: Example of a 2D manifold $\hat{\mathcal{UV}}_P$ of $\mathcal{UV}_P$ defined as $\hat{\mathcal{UV}}_P \triangleq \left\{\mathbf{q} \in \mathcal{UV}_P \mid \theta = \hat{\theta}\right\}$ for $P \subset \{1,2,3\}$.

nected, existing sensor planning methods cannot be directly applied. The visibility regions derived in this section are used in the following section to produce a graphical representation of the configuration space, referred to as a connectivity graph, that captures the connectivity of $\mathcal{C}_{free}$ as well as target visibility. The connectivity graph is then searched to for a minimum distance sensor path that avoids obstacles and views all targets by guaranteeing occlusion avoidance using the visibility regions. Fig. 3.7 shows the 2-dimensional manifold at $\theta = \hat{\theta}$ of the set visibility regions for three targets.

3.4 Directional Visibility Graph

The connectivity graph seeks to capture the free space connectivity between visibility regions in the form of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, in which the set of nodes, $\mathcal{N}$, represents a set of configuration-label pairs, $N = (\mathbf{q},\ P) \in \mathcal{N}$, where $P \subseteq \mathcal{I}$ is the index set of all targets visible at configuration $\mathbf{q} \in \mathcal{C}$. The nodes are connected by a set of undirected edges or arcs $\mathcal{E}$ where the arc $(N_l, N_p) \in \mathcal{E}$ connects nodes $N_l,\ N_p \in \mathcal{N}$. The set of nodes $\mathcal{N}$ is computed by representing each set visibility region $\mathcal{UV}_P$ with a collection of $n_P > 0$ representative configurations, $\mathcal{Q}_P \triangleq \{\mathbf{q}_i^* \in \mathcal{UV}_P\}_{i=1}^{n_P}$.

Because a visibility region may be concave, the region's centroid may not belong to the visibility region, and therefore the Chebyshev center(s) [10] are chosen to represent each set visibility region. Typically, the Chebyshev center is computed by solving a linear program for convex shapes yielding a unique solution, but the Chebyshev center for concave visibility region may have more than one local maxima. Therefore, the set of node configurations $\mathcal{Q}_P$ that represent $\mathcal{UV}_P \subset \mathcal{C}$ are the set of all local maxima of the boundary-distance function $z(\mathbf{q}) \triangleq \min_{\boldsymbol{\xi} \in \partial\mathcal{UV}_P} \|\boldsymbol{\xi} - \mathbf{q}\|$, expressed as

$$\mathcal{Q}_P = \{\mathbf{q}^* \in \mathcal{UV}_P \mid z(\mathbf{q}^*) = \sup_{\mathbf{q} \in \mathcal{UV}_P} \{z(\mathbf{q}) \mid \|\mathbf{q} - \mathbf{q}^*\| \leq R, R > 0\}\Big\} \tag{3.14}$$

For every $P \subseteq \mathcal{I}$ and every Chebyshev center in $\mathcal{Q}_P$, a node $N = (\mathbf{q}, P)$, with $\mathbf{q} \in \mathcal{Q}_P$, is added to the collection of nodes, $\mathcal{N}$. Additionally, the sensor's initial configuration $\mathbf{q}_0 \in \mathcal{C}$ and the index set P_0 of visible targets at $\mathbf{q}_0$, is used to create a node N_0 which is added to the collection of nodes, $\mathcal{N}$. An example of a concave set visibility region with multiple Chebyshev centers is illustrated in Fig. 3.8.

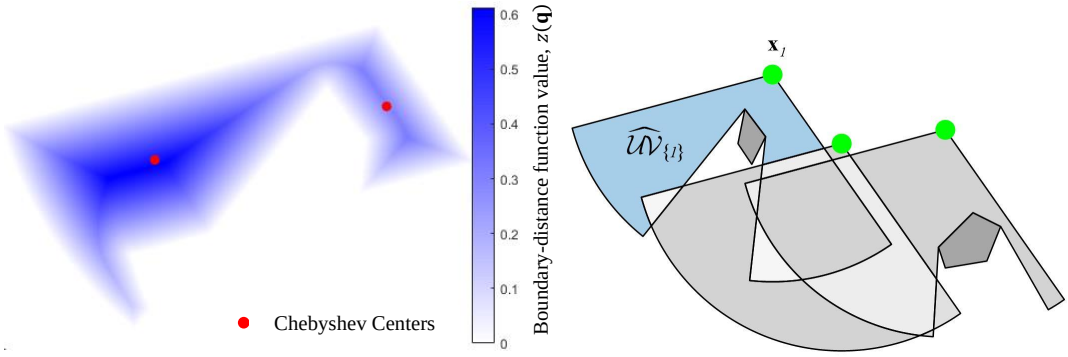


Figure 3.8: The boundary-distance function, $z(\mathbf{q}) : \mathcal{UV}_P \rightarrow \mathbb{R}$, is illustrated for the two-dimensional manifold $\hat{\mathcal{UV}}_1$ of $\mathcal{UV}_1$, along with the Chebyshev centers that occur at the local maxima of the boundary distance function.

The edge $(N_l,\ N_p) \in \mathcal{E}$ that connects nodes N_l and N_p exists if the straight-line path between each node's configuration $\mathbf{q}_l,\ \mathbf{q}_p \in \mathcal{C}$ is collision-free, that is

$$L(\mathbf{q}_l, \mathbf{q}_p) \cap \bigcup_{j=1}^{M} \mathcal{B}_j = \emptyset \Rightarrow (N_l, N_p) \in \mathcal{E} \tag{3.15}$$

where the line segment is defined as $L(\mathbf{q}_l,\ \mathbf{q}_p) \triangleq \{(1-\alpha)\mathbf{q}_l + \alpha\mathbf{q}_p \mid \alpha \in [0,1]\}$. However, Eq. 3.15 alone for connectivity will often lead to a graph that is not connected. That is, a path may not exist that connects any node to any other node in the graph. Therefore, a probabilistic roadmap method (PRM) is used to connect each node to at least some large fraction $\delta \in (0,1]$ of other nodes. This connectivity methodology guarantees that every edge in the graph is collision-free, and therefore any path produced from $\mathcal{G}$ must also be collision-free.

A piecewise continuous path τ can be characterized using the connectivity graph $\mathcal{G}$, as a sequence of adjacent nodes, referred to as a channel

$$\begin{aligned} C =& N_0, N_1, ..., N_n, \\ & \text{where } (N_l, N_{l+1}) \in \mathcal{E} \ \ \forall N_l, N_{l+1} \in C \end{aligned} \tag{3.16}$$

that starts at the initial node N_0. Therefore, the cost associated with a channel $C = N_0, N_1, ..., N_n$ is the length of the associated piecewise linear path in $\mathcal{C}$ that

is computed using (3.4), denoted by

$$J(C) \triangleq \sum_{k=0}^{n-1} D(\mathbf{q}_k, \mathbf{q}_{k+1}) \tag{3.17}$$

where the distance metric D is defined in (3.2), and $N_k = (\mathbf{q}_k, P_k)\ \forall N_k \in C$. The set of targets that are visible in a channel C is computed as the union of the node labels, $\bigcup_{k=0}^{n} P_k$, where $P_k \subseteq \mathcal{I}$. Then, the optimization problem in (3.5) is reduced to finding the optimal channel C^* in the connectivity graph $\mathcal{G}$ by solving the following problem

$$\begin{aligned} \min_{C \in \mathcal{G}} \quad & J(C) = \sum_{k=0}^{n-1} D(\mathbf{q}_n, \mathbf{q}_{n+1}) \\ \text{subject to} \quad & \bigcup_{k=0}^{n} P_k = \mathcal{I} \end{aligned} \tag{3.18}$$

In this dissertation, C^* is determined using a meta-heuristic optimization approach that incorporates a tradeoff of local search and stochastic search in order to search the most promising regions of $\mathcal{G}$ at a global scale [12]. The MCTS framework presented here is applicable to any graph optimization and thus it is used with fixed parameters for all comparison analyses in the following section in which a graph is optimized.

The complete path planning algorithm is shown in Alg. 1. The inputs to the algorithm are the obstacle geometries $\mathcal{B}_j,\ j \in \mathcal{J}$, the target positions $\mathbf{x}_i,\ i \in \mathcal{I}$, and the sensor initial configuration $\mathbf{q}_0$. The parameters of the proposed algorithm are the number of rotational discretizations $\kappa > 0$, the number of samples $\eta > 0$ used in the PRM approach for connecting graph nodes, and the number of Monte Carlo simulations $\mu > 0$ used in the MCTS optimization. In the following section, the complexity of the proposed algorithm is investigated.

Algorithm 1 Exact Visibility Planner

Require: $\mathcal{B}_j \subset \mathcal{W}\ \forall j \in \mathcal{J},\ \mathbf{x}_i\ \forall i \in \mathcal{I},\ \mathbf{q}_0$
Parameters: $\kappa > 0,\ \eta > 0,\ \mu > 0$
Initialize: $\mathcal{N} \leftarrow \emptyset,\ \mathcal{E} \leftarrow \emptyset$
for $i \in \mathcal{I}$ **do**
  $\mathcal{PV}_i \leftarrow \text{getTargetVisibility}(\cup_j \mathcal{B}_j,\ \mathbf{x_i};\ \kappa)$
end for
for $P \subseteq \mathcal{I}$ **do**
  $\mathcal{UV}_P \leftarrow \text{getSetVisibility}(\{\mathcal{PV}_i\}_{i\in\mathcal{I}},\ P)$
  $\mathcal{Q}_P \leftarrow \text{getChebyshevCenters}(\mathcal{UV}_P)$
  for $\mathbf{q} \in \mathcal{Q}_P$ **do**
    $\mathcal{N} \leftarrow \mathcal{N} \cup (\mathbf{q},\ P)$
  end for
end for
$\mathcal{E} \leftarrow \text{connectNodes}(\mathcal{N},\ \cup_j \mathcal{B}_j;\ \eta)$
$\mathcal{G} = (\mathcal{N},\ \mathcal{E})$
$C^* \leftarrow \text{MonteCarloTreeSearch}(\mathcal{G};\ \mu)$
return C^*

3.5 Monte-Carlo tree search graph optimization

The meta-heuristic optimization approach used in this work is based on the Monte-Carlo Tree Search (MCTS) [78]. In the MCTS approach, the connectivity graph $\mathcal{G}$ is adaptively searched starting from the root node $N_0 = (\mathbf{q}_0, P_0)$ which coincides with the sensor initial configuration. Then, for every node N_p that is connected to N_0, such that $(N_0, N_p) \in \mathcal{E}$, a number of Monte Carlo simulations $\mu > 0$ are performed. Each simulation consists of iteratively and randomly selecting a connected node until all of the targets have been visited or there are no unvisited connected nodes. After each simulation, the total cost of the path taken during the simulation is back-propagated back to the root node. Therefore, after μ simulations for each N_p where $(N_0, N_p) \in \mathcal{E}$, the node N_l that has the minimum length successful path is selected, then N_l is taken to be the root node and all unvisited connected nodes N_p where $(N_0, N_p) \in \mathcal{E}$ are simulated again. This process continues until the path has visited all targets. The value function $Q(N_p)$ used in this dissertation to

select node values during Monte Carlo simulation is given as

$$Q(N_p) \triangleq \frac{1}{\bar{J}} + \gamma\sqrt{\frac{\log n_l}{n_p}} \tag{3.19}$$

where, n_l and n_p are the number of times the node N_l and it's child N_p have been visited during all simulation, respectively, and γ is a pre-defined parameter that controls the exploration-exploitation tradeoff. $\bar{J}$ represents the minimum-length successful path that has been observed over all simulations that have traversed the edge $(N_l, N_p) \in \mathcal{E}$.

3.6 Comparison algorithms and performance metrics

The visibility-based path planning problem addressed in this dissertation does not exist in the literature. However, several methods in the literature address problems that vary slightly from the problem presented here and can be modified slightly to address this problem. In addition, The closed-form visibility regions presented in this dissertation are highly valuable for directional sensor planning and the previously discussed Exact Visibility algorithm is just one possibly algorithm that takes advantage of the visibility regions. For these reasons, this section presents alternative algorithms that address the visibility planning problem presented in this dissertation.

3.7 Comparison Algorithms

Prior to this work, the visibility path planning problem addressed in this dissertation has not been solved, although several algorithms address problems that share

some characteristics with the visibility problem presented here, which are implemented with slight modifications for comparison in this dissertation. In addition, two alternative approximation methods are presented that make use of the closed-form visibility regions in this section that will be compared to the exact visibility algorithm developed in the previous section.

Pruned Visibility Algorithm The first comparison algorithm is referred to as the pruned visibility algorithm. This method makes a number of simplifications to the exact visibility method in order to reduce the configuration space dimensionality and reduce the size of the connectivity graph in order to reduce computational cost. First, the target visibility region $\mathcal{PV}_i \subset \mathcal{C}_{free}$ is computed for each target index $i \in \mathcal{I}$, which are then used to compute reduced-dimensionality translational target visibility regions,

$$\mathcal{PV}_i^T \triangleq \bigcup_{\theta \in \mathbb{S}^1} \left\{ \mathbf{x} \in \mathcal{W} \mid [\mathbf{x} \quad \theta]^T \in \mathcal{PV}_i \right\} \subseteq \mathcal{W} \tag{3.20}$$

The translational target visibility regions characterize the set of positions from which a particular target is visible from at least one sensor orientation. Similarly, the translational set visibility regions $\mathcal{UV}_P^T$ are computed using Eq. 3.13, where $P \subseteq \mathcal{I}$. Then, redundant translational set visibility regions with a low index set cardinality are pruned. That is, if the set visibility regions $\mathcal{UV}_{P_1}^T$ and $\mathcal{UV}_{P_2}^T$ are such that $P_1 \subset P_2$ and $\mathcal{UV}_{P_2} \neq \emptyset$, then $\mathcal{UV}_{P_1}^T$ is pruned since all of the targets are visible from a different position. The remaining collection of pruned translational set visibility regions are used to construct connectivity graph $\mathcal{G}_{Pruned} = (\mathcal{N}_{Pruned}, \mathcal{E}_{Pruned})$, such that $\mathcal{N}_{Pruned} = \{\mathbf{x}_0, \mathbf{c}_1, \mathbf{c}_2, ...\}$, where $\mathbf{x}_0 \in \mathcal{W}$ is the sensor's initial position and $\mathbf{c}_k \in \mathcal{W}$ is a representative point of the pruned translational set visibility regions. $\mathbf{c}_k$ is taken to be the geometric center of the k-th translational set visibility region if $\mathbf{c}_k$ is in the region, otherwise $\mathbf{c}_k$ is sampled uniformly from the region.

The nodes are then connected following the same procedure as that for the exact visibility algorithm, in which a local planner is used to fully connect the graph with collision-free paths. This pruning technique dramatically reduces the total number of set visibility regions compared to the exact method, in turn also reducing the resulting connectivity graph size.

Cell Decomposition Algorithm The cell decomposition approach is a commonly used technique for robot and sensor path planning. This method discretizes the sensor rotation space into $\kappa > 0$ orientations, $\theta_1, ..., \theta_\kappa$. For each θ_u, the method computes the rectangloid approximations $\mathcal{RT}_i^u$ for the target visibility region $\mathcal{PV}_i^u$, such that $\mathcal{RT}_i^u \subset \mathcal{PV}_i^u$, $i \in \mathcal{I}$ and $\mathcal{PV}_i^u = \{\mathbf{q} \in \mathcal{PV}_i | \theta = \theta_u\}$. Then the method computes rectangloid approximation $\mathcal{RB}_j$ for the obstacle $\mathcal{B}_j$, which satisfies $\mathcal{B}_j \subset \mathcal{RB}_j$, $j \in \mathcal{J}$. Then for each θ_u the method computes the rectanguloid approximation $\mathcal{K}_{void}^u$ for the void configuration space $\mathcal{C}_{void}^u = C_{free}^u \setminus \{\bigcup_{j\in\mathcal{J}} \mathcal{B}_j \cup \bigcup_{i\in\mathcal{I}} \mathcal{PV}_i\}$. Then, for each rectangloid $\mathcal{RT}_k^u$, the centroid of each rectangle $\mathbf{c}_k$ represents a node in a connectivity graph, $[\mathbf{c}_k^T, \theta^u]^T \in \mathcal{N}_{Cell}$, such that $\mathcal{G}_{Cell} = (\mathcal{N}_{Cell}, \mathcal{E}_{Cell})$. Two nodes that share the same θ_u are connected if and only if their corresponding rectangloid cells share an edge. Two nodes N_1, N_2 that do not have the same θ_u are connected if and only if the corresponding rectangloids have a nonzero intersection and the corresponding orientation angles satisfy $|\theta_{u_1} - \theta_{u_2}| \leq \epsilon$, where $\epsilon > 0$ is a user-defined threshold.

Ad-hoc Traveling Salesman Problem Algorithm The third method presented here, referred to as the Ad-hoc TSP, directly adapts the standard TSP optimization problem [2] to the visibility problem presented in this dissertation. The standard TSP problem does not account for obstacles in the workspace. For

this reason, the Ad-hoc TSP solution adapts a visibility diagram along with a PRM methodology to ensure the TSP graph is fully connected leading to a methodology that most similar to a standard TSP solution while ensuring the algorithm is capable of solving the more difficult visibility method presented in this dissertation. In particular, the initial sensor configuration $\mathbf{q}_0$ and each target position $\mathbf{x}_i$ for all $i = 1, ..., N$ is treated as a node in the TSP graph, denoted by $\mathcal{G}_{TSP} = (\mathcal{N}_{TSP}, \mathcal{E}_{TSP})$, such that $\mathcal{N}_{TSP} = \{\mathbf{q}_0, \mathbf{x}_1, ..., \mathbf{x}_N\}$. The nodes are connected by first constructing a visibility diagram, such that any two nodes that satisfy line of sight with respect to C-obstacles are connected by a straight edge. Then, the remaining nodes are connected using a PRM local planner that samples a uniform distribution over the workspace in order to avoid collisions with obstacles. Therefore, $\mathcal{G}_{TSP}$ is a fully connected graph that can be optimized according the same MCTS method presented in the appendix.

Coverage Algorithm The final algorithm used as a benchmark for comparison is referred to as the Coverage algorithm, and is similar to the WRP [43]. The coverage method employs a lawnmower path shape, such that the sensor moves in a back-and-forth path across the workspace in an attempt to cover the entire workspace with the sensor FOV geometry. Traditional WRP solutions do not account for a bounded field of view geometry and thus perfect coverage is not guaranteed by this method. The Coverage algorithm also uses a PRM local planner to avoid collisions with obstacles along the lawnmower path.

Path Performance Metrics The following sections aim to compare the exact visibility algorithm, which is the main contribution of this dissertation, with the several methods presented in this section, including the pruned visibility and cell

decomposition algorithms, which are also novel algorithms developed in this dissertation for the purpose of comparison with the exact visibility method. In order to compare the overall quality of the path produced by the visibility-based planning algorithms, as well as the travelling salesman and coverage algorithms, a set of performance metrics are given here.

The first performance metric is the path length $J(\tau) \geq 0$, as defined in the optimization problem statement in Eq. 3.4. Clearly, because the problem aims to minimize the path length, a smaller value for the path length produced by an algorithm is indicative of a higher quality path. In addition to the path length $J(\tau)$ the goal of the algorithm as stated in the problem formulation, is to observe each of the N target positions $\mathbf{x}_i$ for all $i = 1, ..., N$ from at least one configuration along the path τ produced by the algorithm. The path performance metric used to measure this objective is given by the number of targets observed along a path, denoted by $n_T(\tau) \in [0, N]$. Because several workspaces are investigated in the following sections containing vastly different numbers of targets, the performance metric $n_T(\tau)$ is normalized by the number of targets in the workspace, or $\frac{n_T(\tau)}{N} \in [0, 1]$. A value of $\frac{n_T(\tau)}{N} = 1$ indicates that all of the targets are observed at least once along the path τ, and any value less than 1 indicates that some targets were missed. The final metric used to quantify the quality of a path is referred to as the *path performance* which is the ratio of the previously mentioned metrics. The path performance is given as $\frac{n_T(\tau)}{J(\tau)} \geq 0$, such that larger values of path performance indicate a higher quality path.

3.8 Algorithm complexity analysis

The various algorithms presented above are all designed to address the visibility path planning problem presented in Sec. **??**. In this section, the computational complexity is investigated for all of the algorithms that take advantage of the closed-form visibility regions, which are the main contribution of this dissertation. That is, the computational complexity analysis is performed for the (a) Exact Visibility, (b) Pruned Visibility, and (c) Cell Decomposition algorithms.

Exact Visibility Computational Complexity The computational complexity of computing the target visibility regions is $O(\kappa NMm \log m + \kappa NMm)$ where κ is the number of rotations used to discretize the rotational part of the sensor's configuration space, N is the number of targets, M is the number of obstacles, and $m \triangleq \frac{1}{M}\sum_{j=1}^{M} m_j$ is the average number of vertices in a polygonal obstacle, and m_j is the number of vertices in obstacle $\mathcal{B}_j$. The first term is induced by the convex hull operation which has complexity $O(m \log m)$ [52] and is computed κNM times, and the second term is from the coverage cone operation which has complexity $O(m)$ and is computed κNM times. The set visibility region requires the power set of the targets, which has computational complexity $O(2^N)$ and is computed for each κ. Thus, the set visibility region has complexity $O(\kappa 2^N)$ which is computationally prohibitive for large N. To ensure the graph is connected, a probabilistic roadmap is computed and solved using Dijkstra's algorithm [55], which has complexity $O(\eta^2)$, where η is the number of samples used to construct the roadmap. Therefore, the complexity for connecting the graph is $O(\kappa N \eta^2)$. The MCTS algorithm that is used to find the best channel through $\mathcal{G}$ has complexity $O(\mu bd)$ where μ is the number of Monte Carlo simulations, b is the maximum breadth of the tree,

and h is the maximum depth of the tree. Since the tree should never be searched wider or deeper than the number of targets, the computational complexity of the MCTS algorithm is $O(\mu N^2)$. Because each of these operations happens in series, the total computational complexity of the exact visibility algorithm is

$$O(\kappa NMm \log m + \kappa NMm + \kappa 2^N + \kappa N\eta^2 + \mu N^2)$$

For even reasonably sized N, the $\kappa 2^N$ term dominates and is the bottleneck of the algorithm. This problem can be addressed by realizing that the set visibility region does not exist for targets that are more than twice the maximum sensing radius from each other, that is

$$\|\mathbf{x}_i - \mathbf{x}_j\| > 2r \Rightarrow \mathcal{PV}_i \cap \mathcal{PV}_j = \emptyset$$

where $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{W}$ are target positions and $\mathcal{PV}_i, \mathcal{PV}_j \subset \mathcal{C}$ are the associated target visibility regions, respectively. Therefore, it is necessary that for a set visibility region to exist for a combination of targets, all positions are within a distance of $2r$ or less of each other. Therefore, the computational complexity is only bounded from above by the exponential term and is typically significantly more efficient. More precisely, the set visibility region computational complexity goes like $O(2^{N^*})$, where N^* is the maximum number of targets that are within a ball of radius $2r$, and r is the maximum sensing range of the sensor. Then, if the targets are uniformly distributed in a workspace that has area $A(\mathcal{W})$,

$$N^* \approx \frac{\pi r^2}{A(\mathcal{W})} N \tag{3.21}$$

Therefore, as long as the square of the sensor range is much less than the area of the workspace, as is typically the case, the problem will remain computationally tractable. Fig. 3.9 demonstrates how small values of a, which are typically much less than 0.2, lead to computationally tractable set visibility region algorithm complexity although the upper bound on the complexity is intractable.

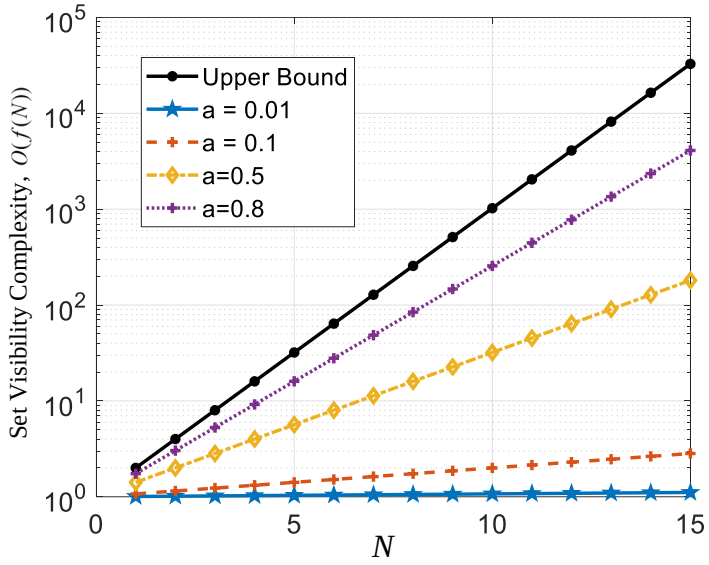


Figure 3.9: Set visibility region complexity for the exact visibility algorithm as the ratio of $a = N^*/N$ is varied.

Pruned Visibility Region Algorithm Complexity Because the pruned visibility algorithm uses the target visibility regions, which has complexity $O(\kappa NMm \log m + \kappa NMm)$, still applies with the same complexity as described in the previous paragraph. However, the pruned visibility algorithm does not require that all set visibility regions be computed, and instead only requires the set reduced-dimension visibility regions with large cardinality. More specifically, if a set visibility region $\mathcal{UV}_{P_1}$ is non-empty and the target index $i \in P_1$, then all set visibility regions $\mathcal{UV}_P$, such that $i \in P$ and $|P| < |P_1|$, are not computed and only $\mathcal{UV}_{P_1}$ is used to construct the directional visibility graph. By this approach, the largest possible number of set visibility regions that need to be computed is N. In addition, the pruned visibility region algorithm ignores the orientation of the robot and thus, the set visibility construction does not depend on a rotational discretization, κ. Therefore, the complexity of this step is linear with the number of targets, $O(N)$, and the remaining steps of the algorithm are the same as for the exact visibility algorithm. Therefore, the computational complexity of the pruned

visibility algorithm is,

$$O(\kappa NMm \log m + \kappa NMm + N + N\eta^2 + \mu N^2)$$

Clearly, the pruned visibility algorithm provides significant savings as N increases when compared to the exact visibility method by reducing the complexity from exponential to linear. Furthermore, the resulting graph size for this algorithm will inevitably be much smaller than that of the exact visibility method and will therefore require less time to find a reasonable channel through the graph, as will be shown in the numerical results in the following sections.

Cell Decomposition Algorithm Complexity The first step in the cell decomposition algorithm has complexity that is similar to the rectangle decomposition algorithm presented in [14]. However, the main difference being that the complexity depends on the number of edges in the obstacles, denoted by b, and the number of edges in the target visibility regions denoted by v. In addition, the discretization in the rotational direction κ used in this dissertation is incorporated linearly, such that the rectangle decomposition step has complexity $O(\kappa(b+v)^2)$. Then, the graph construction step has complexity $O(\kappa(b+v))$, from [14], and the optimization routine complexity is the same as for the previous methods, except the cell decomposition Monte Carlo Tree Search can lead to much wider and deeper trees that are approximated as $\kappa(b+v)$ for the breadth and the width, leading to $O(\mu\kappa^2(b+v)^2)$ for the optimization algorithm complexity. Then, the complexity of the cell decomposition algorithm is given as,

$$O(\kappa(b+v)^2 + \kappa(b+v) + \mu\kappa^2(b+v)^2)$$

Because the number of edges that make up the visibility regions can be very large, the quadratic complexity of the cell decomposition algorithm can be pro-

hibitive for large workspaces that contain many targets and/or many obstacles. This quadratic complexity exists in both the graph construction and the optimization part of the algorithm.

In summary, the exact visibility method scales exponentially in the number of targets, the pruned visibility algorithm scales quadratically in the number of targets, and the cell decomposition scales quadratically in the total number of edges of obstacles and visibility regions. If computational efficiency were not an issue, the exact visibility algorithm is expected to outperform the other two methods. However, the pruned visibility algorithm provides a tradeoff to the computational efficiency by throwing out the low cardinality set visibility regions and ignoring the orientation of the robot in order to reduce the problem dimensionality. The cell decomposition method provides a significantly different approach that scales in terms of the obstacle geometries and the visibility region geometries. The following results sections provide numerical and physical experiments that both validate the computational complexity analyses presented here as well as the path performance between the algorithms.

3.9 algorithm path performance comparison

This section presents the simulation and experimental results of the visibility based planning algorithms developed in this dissertation. First, the simulation results are presented for the exact visibility algorithm, the main contribution of this dissertation, for the task of classifying multiple targets in a complex workspace, in order to demonstrate the direct application of this algorithm in a realistic use case. Then, the path performance of the exact visibility algorithm is quantitatively com-

pared to multiple algorithms presented in the previous sections. In addition to the path performance comparison, the algorithm computational efficiency is compared numerically, validating the analytical results of Sec. **??**.

This subsection presents simulation results for the exact visibility method presented in this dissertation. The exact visibility algorithm and all comparison algorithms are implemented in MATLAB© 2019b on a Dell™Precision Tower 7910 equipped with two Intel™2.40 GHz Xeon CPU processors.

The functionality of the proposed exact visibility path planning algorithm is employed for a multi-target classification task, which is implemented in a complex 3D and photo-realistic workspace, simulated using a rendering engine known as Unreal Engine™ [88]. The target classification is performed using an off-the-shelf state of the art person detection algorithm [81], and the maximum sensing range of the sensor FOV geometry is determined by estimating the scale of humans that the algorithm is capable of recognizing consistently, given the camera resolution. For this simulation, a high-resolution 1080p image resolution is simulated and the maximum sensing range is determined to be 25 meters.

Fig. 3.10 shows the realistic and complex 3D workspace that is simulated in Unreal Engine™. This complex 3D demonstration validates the assumptions made throughout this dissertation in order to realize high-quality paths in complex environments using reduced dimensionality representations of complex geometries of the sensor FOV and polygonal obstacles. In addition, the simulations in Unreal Engine allow for variable luminosity and fog conditions, for example that can be controlled perfectly. The three-dimensional robot and sensor FOV used in the simulations is shown in Fig. 3.11.

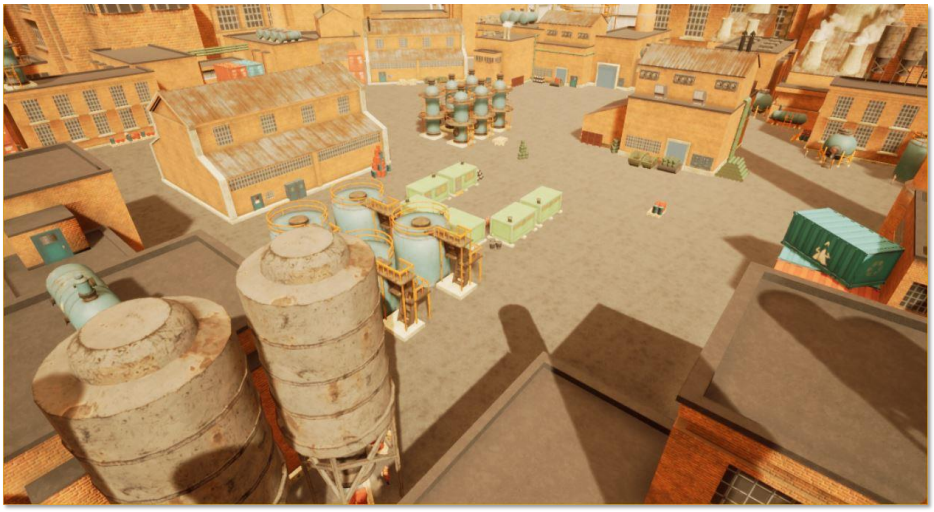


Figure 3.10: Unreal Engine simulation environment.

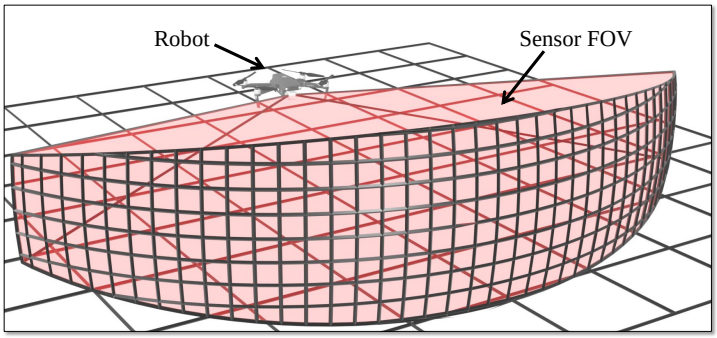


Figure 3.11: Robot and sensor FOV used in Unreal Engine simulation environment.

The workspace $\mathcal{W}$ in this simulation consists of $M = 59$ obstacles with several narrow passages and $N = 13$ targets that lead to overlapping visibility regions. The initial robot configuration is chosen such that no obvious path exists and no targets are visible from the initial configuration. The polygonal representation of the obstacles is obtained from the Unreal Engine mesh data for each obstacle and taking the convex hull of the obstacle vertices and extruding the geometries to be uniform in the vertical direction. Fig. 3.12 shows the simplified 3D representation of the complex workspace from Fig. 3.10. The geometrically approximated workspace in Fig. 3.12 is used as an input to the exact visibility path planning

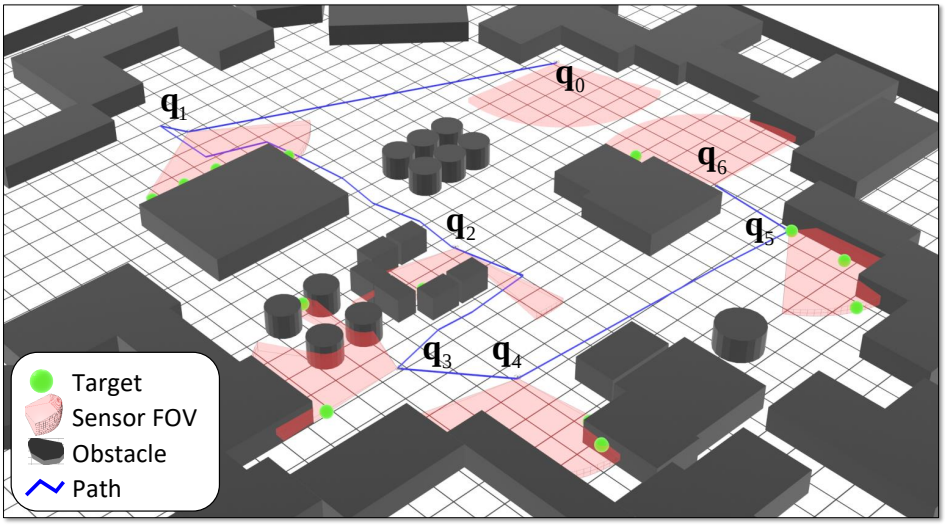


Figure 3.12: The geometrically simplified simulation workspace used for path planning with the exact visibility algorithm, along with the generated path produced by the algorithm that successfully visits all target positions.

algorithm and the resulting path produced is shown in blue on the figure.

The exact visibility path planning algorithm successfully observes all of the targets along the proposed path, as shown in Fig. 3.12. In addition, the object recognition algorithm successfully recognizes all of the targets in the images from the proposed waypoints of the algorithm, and the resulting images with bounding boxes are shown in Fig. 3.13.

An interesting observation of the path produced by the exact visibility algorithm is shown in Fig. 3.14. In particular, the algorithm finds a configuration that is capable of viewing two targets simultaneously due to their combined set visibility region that has been used in the planner. One of the humans is only visible through a narrow passage, but because the visibility regions have been described using closed-form geometry, the exact visibility method exploits this sensor configuration and likely saves several meters of robot travel. Fig. 3.14 shows the complex environment, and the geometrically simplified environment, as well as the

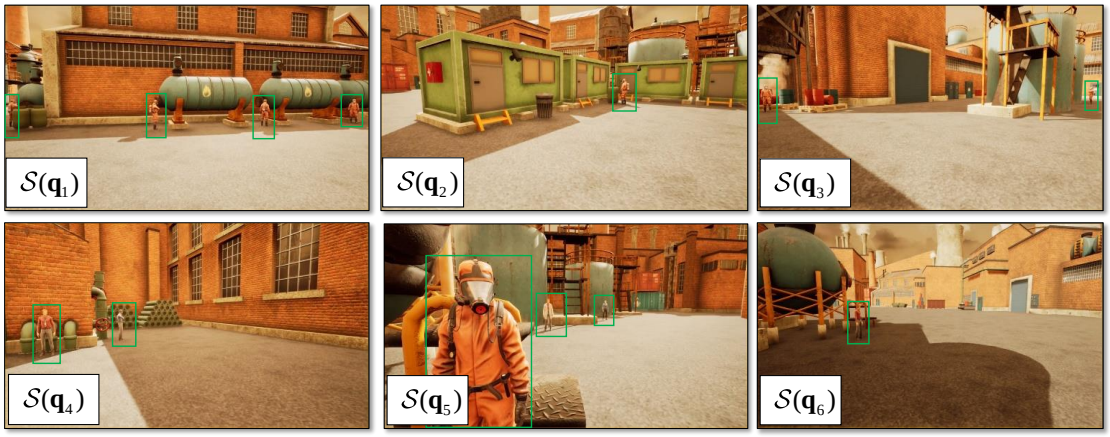


Figure 3.13: Resulting sensor FOV images taken from the proposed waypoints generated by the Exact Visibility algorithm, shown with all target humans being successfully recognized in green bounding boxes.

sensor FOV with correctly recognized targets.

In addition to the simulated target recognition results, further results are produced that demonstrate the capability of the exact visibility algorithm. Fig 3.15 shows the exact visibility method easily finds a high quality configuration capable of viewing multiple targets in the presence of occlusions while simultaneously avoiding collisions with the obstacles. This example highlights the benefits of the exact visibility method and the closed-form visibility regions that are developed in this dissertation. Alternative planning approaches, such as purely sampling based methods would likely not find such a valuable configuration without requiring hundreds of samples, rather than the very small number of nodes required using the exact visibility algorithm.

Algorithm Performance Comparison

This subsection presents a rigorous and quantitative comparison of the multiple algorithms described in Sec. 3.7. The first comparison is between the Exact

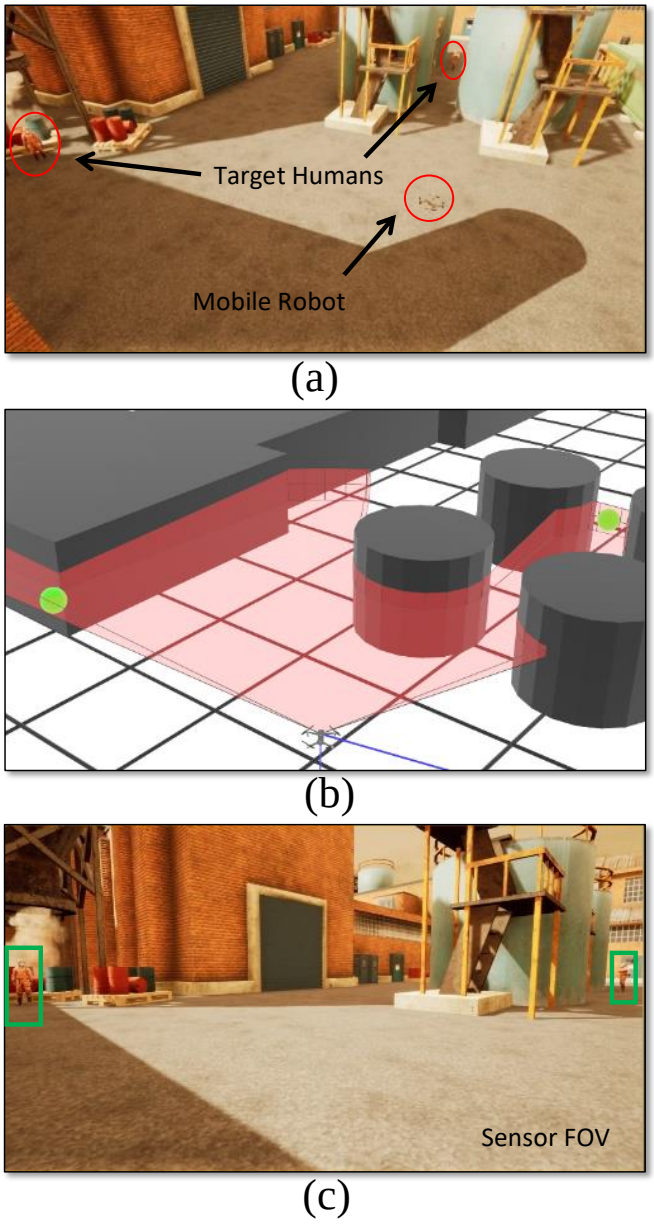


Figure 3.14: A configuration determined using the exact visibility algorithm where two targets are visible simultaneously in the complex 3D environment (a), the geometrically simplified environment (b), and the associated sensor FOV image and successful human target recognition shown as bounding boxes (c).

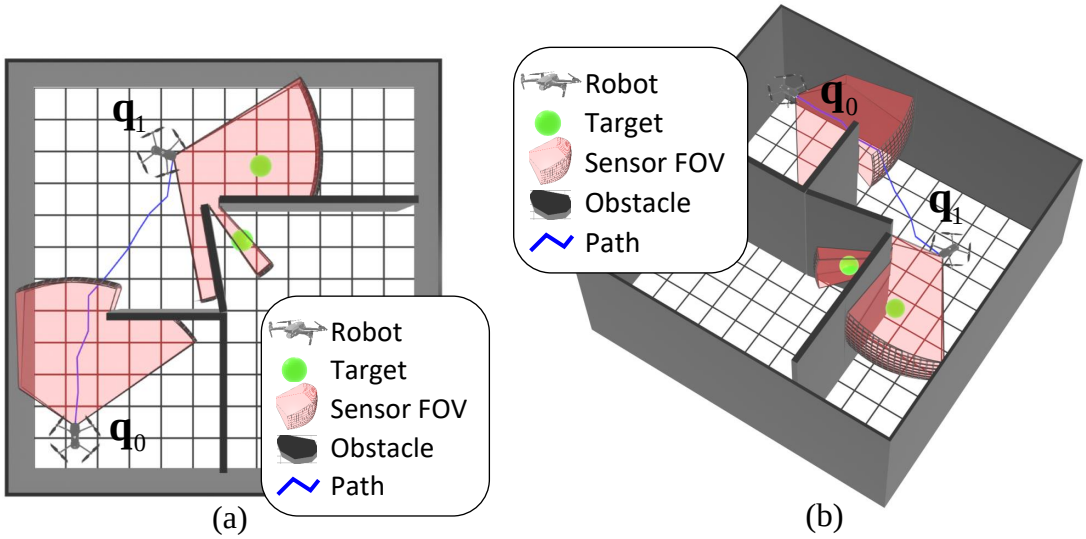


Figure 3.15: The exact visibility algorithm finds a high quality configuration at $\mathbf{q}_1$ capable of viewing two targets in the presence of occlusions while simultaneously avoiding collisions to reach the desired configuration.

visibility and Pruned Visibility methods, which are very similar approaches with the pruned visibility method making several simplifying assumptions in order to save on efficiency. Fig. 3.16 demonstrates the loss in performance of the pruned visibility method due to the dimensionality reduction by ignoring the rotational component of the configuration space. In this simulation, three equally spaced targets are placed just far enough so that there are no overlapping visibility regions, which leads to each target requiring an individual configuration chosen by the algorithm. The exact visibility algorithm is capable of taking advantage of the robot yaw orientation in order to significantly reduce the path length, while the pruned visibility method is required to travel much further due to the rotation angle being ignored.

In order to quantitatively and fairly compare the path performance of the various algorithm, several workspaces with variable numbers of targets and obstacles are randomly generated. More specifically, the number of targets N and number

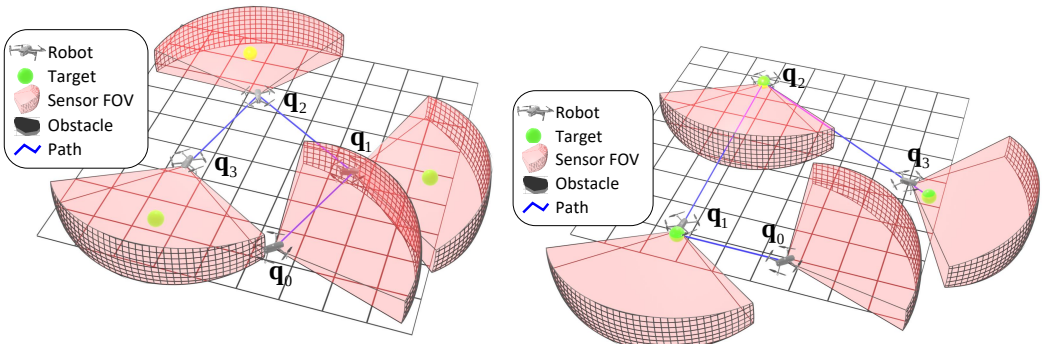


Figure 3.16: Comparison of the Exact Visibility (a) and Pruned Visibility (b), which produce a path which have path lengths of $J(\tau_{EV}) = 15.48$ meters and $J(\tau_{PV}) = 24.33$ meters respectively, demonstrating the exact visibility method outperforming the pruned visibility method.

of obstacles M are chosen by the user and the initial sensor configuration, the position of the targets and obstacles, as well as the obstacle geometries are randomly generated in MATLAB™. The random obstacles are generated by fist randomly sampling a position from the workspace uniformly, then randomly selecting an integer number of vertices to represent the obstacle uniformly between 3 and 8. The position of individual obstacle vertices relative to the randomly sampled obstacle location are chosen randomly between relatively small bounds with respect to the size of the workspace. Then, the target positions are randomly sampled from the free workspace. Fig. 3.17 shows a randomly generated workspace, with $M = 25$ obstacles and $N = 8$ targets, along with a path produced by the exact visibility algorithm.

In the algorithm comparison several problem parameters are varied and the path performance of each method is recorded for each simulation parameter set. The parameters that are varied are the number of targets N, the number of obstacles M, and the maximum sensing range of the sensor FOV r. For each set of parameters, the path length $J(\tau) \geq 0$, normalized target coverage $\frac{n_T(\tau)}{N}$, and path performance $\frac{n_T(\tau)}{J(\tau)}$ are recorded and compared as the parameters are varied.

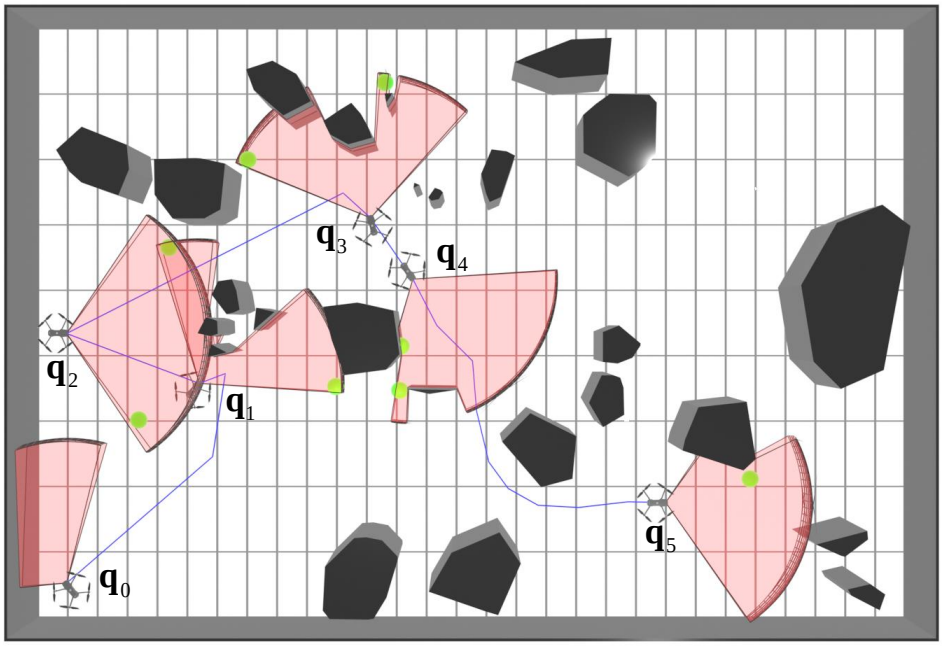


Figure 3.17: Example of the randomly generated workspace, with $M = 25$ obstacles and $N = 8$ targets, used the in the algorithm comparison experiments with a path produced by the exact visibility algorithm.

In addition, the MCTS optimization algorithm parameters are held constant for every simulation in the performance comparison. In particular, it is noted that the number of Monte Carlo simulations μ is not varied between algorithms.

The first experiment in the algorithm comparison demonstrates the algorithm path performance as the number of targets N is increased from $N = 1$ to $N = 15$, and is shown in Fig. 3.18. All of the simulations used in this experiment use $M = 25$ obstacles and a maximum sensing range of $r = 5$ meters. For small N, it can be seen that the path length of the various algorithms is quite similar, and all of the algorithms view every target, except for one. The Ad-hoc TSP algorithm fails to visit many of the targets because the targets are too close to the obstacles for the simple TSP algorithm to visit the targets without causing a collision between the obstacle and the robot. This phenomena further demonstrates the value that is gained by taking advantage of the sensor FOV. Furthermore, as the

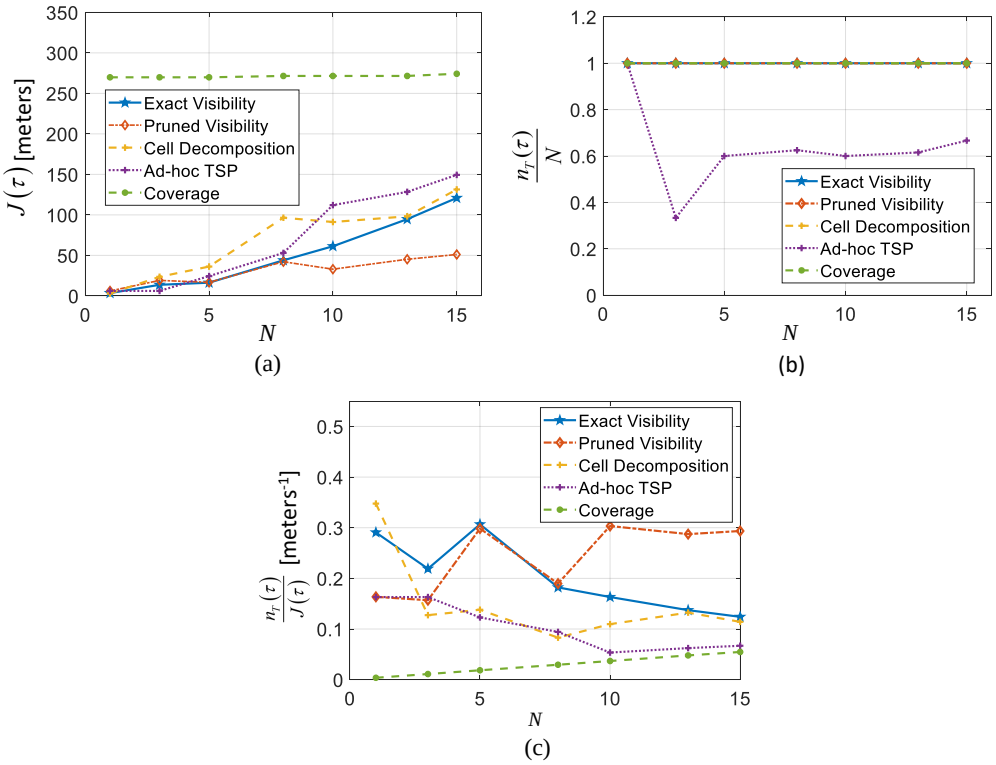


Figure 3.18: Path length $J(\tau)$ (a), normalized target coverage $\frac{n_T(\tau)}{N}$ (b), and path performance $\frac{n_T(\tau)}{J(\tau)}$ (c), as the number of targets N in the workspace increases.

number of targets is increased the pruned visibility algorithm overtakes the exact visibility algorithm. This makes perfect sense, given that the number of Monte Carlo simulations μ in the MCTS graph optimization is held constant even as the number of targets is increased. The pruned visibility method continues to have high quality path performance as N increases since the size of the pruned visibility graph does not grow as fast as the exact visibility algorithm graph. Thus, it is also reasonable to assume that if the number of Monte Carlo iterations were increased as the number of targets were increased, then the Exact Visibility algorithm would continue to outperform the Pruned Visibility algorithm, as is the case for small N. However, even as the number of targets becomes large, the exact visibility method consistently outperforms all of the other comparison algorithms.

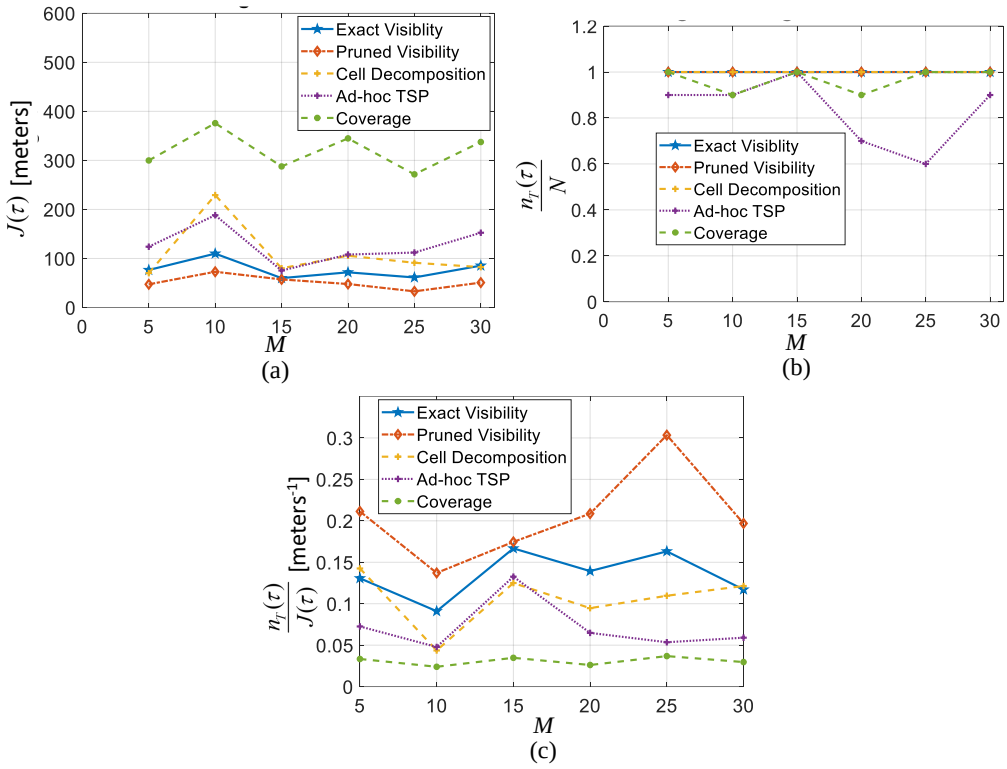


Figure 3.19: Path length $J(\tau)$ (a), normalized target coverage $\frac{n_T(\tau)}{N}$ (b), and path performance $\frac{n_T(\tau)}{J(\tau)}$ (c), as the number of obstacles M in the workspace increases.

The second experiment designed to compare the path performance of the various algorithms investigates the performance as the number of obstacles in the workspace M is increased, as shown in Fig.3.19. The number of targets used in all simulations for this experiment are held constant at $N = 10$ and the maximum sensing range is set to $r = 5$ meters. Because there is a large number of targets the pruned visibility algorithm again slightly outperforms the exact visibility algorithm due to the fixed number of MCTS iterations for these experiments. These experiments show that none of the visibility-based algorithms (i.e., exact visibility, pruned visibility, cell decomposition) miss any targets as the workspace becomes populated with increasingly more and more obstacles, and the path performance remains effectively constant for all of the methods.

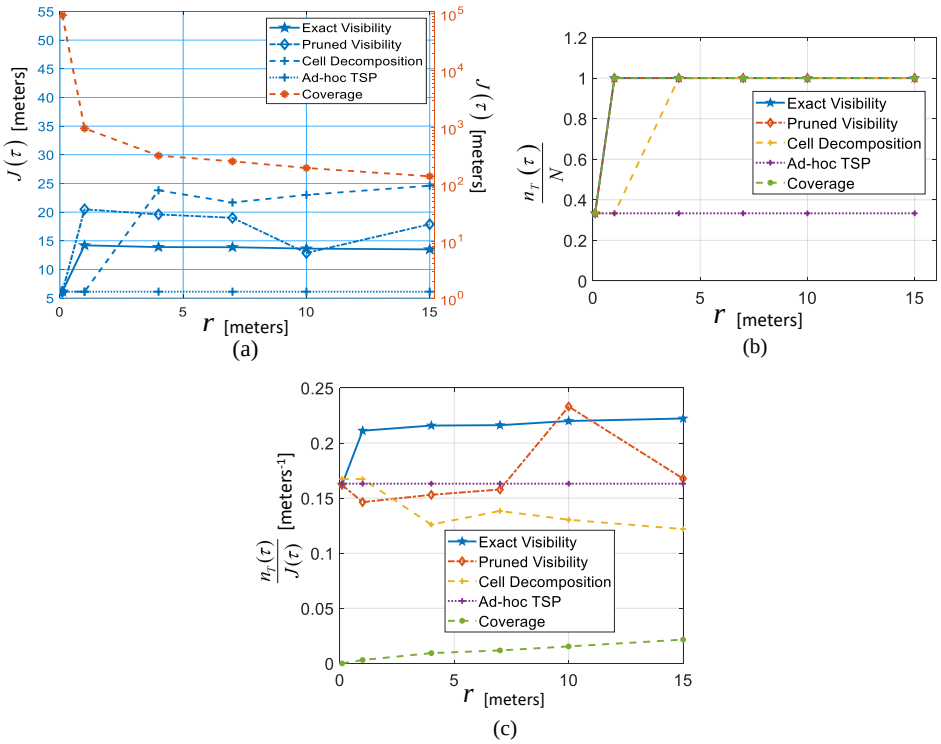


Figure 3.20: Path length $J(\tau)$ (a), normalized target coverage $\frac{n_T(\tau)}{N}$ (b), and path performance $\frac{n_T(\tau)}{J(\tau)}$ (c), as the maximum sensing range r of the directional sensor is increased.

The third experiment is designed to investigate the path performance of the various algorithms as the maximum sensing range r is increased as shown in Fig. 3.20. For these experiments, the number of targets is held constant at $N = 3$ and the number of obstacles is set to be $M = 25$. For these experiments, it is clear that a coverage approach is not an effective method for small sensing radius, which is caused by the fact that as the sensing radius approaches 0, the sensor needs to visit every point in the workspace, and thus $J(\tau)$ for the coverage method goes to infinity. In addition, as the sensing radius approaches zero, all of the visibility-based algorithm performance approaches the performance of the TSP algorithm. However, as the sensing radius increases, the exact visibility method quickly outperforms all other methods, with the occasional close performance of the pruned visibility method.

3.10 Empirical algorithm computational cost comparison

The various algorithms are further analyzed in this subsection with respect to the algorithm computational complexity gathered from numerical experiments. Because in the computational complexity section, Sec. **??**, the number of targets N was shown to be the limiting scaling parameter for the visibility-based algorithms, the results in this section show the run time of various components of the algorithms as N is increased. The first analysis shows the time required for each algorithm to produce the graph that is to be optimized. Alternatively, this analysis can be viewed as the total runtime not counting the time required to optimize the graphs. This analysis is shown in Table 3.1. As expected from the analytical complexity analysis, the exact visibility (EV) method grows exponentially and the pruned visibility (PV) method grows significantly slower. The cell decomposition (CD) algorithm graph construction time complexity does not grow as quickly as the exact visibility algorithm, but the time does remain consistently large due to the fact that this method requires a cell decomposition of the workspace to be performed, regardless of the number of targets. The Ad-hoc TSP (TSP) and Coverage (Cov) algorithms are significantly computationally simpler than the visibility-based methods, although the pruned visibility method complexity is comparable to these methods. Also, the Coverage algorithm does not require a graph optimization, so the time reported for that algorithm is simply the entire time required to compute the path.

Although the main contribution of this dissertation is encompassed in the visibility-based method used to construct a graph for path planning, the size of the graph produced, and the time required to optimize such a graph is of great importance to most any real application of this work. Therefore, Table 3.2 shows the

Table 3.1: Graph construction time (seconds) as N inreases.

Alg.	$N = 1$	3	5	8	10	13	15
EV	1.6	347.3	4173.6	12388	16741	32141	51753
PV	0.4	16.4	1.4	137.6	244.9	410.6	474.5
CD	378.1	606.9	850.9	1085.2	1240.1	1436.1	1506.6
TSP	0.2	31.0	70.9	235.8	389.3	540.2	683.1
Cov	46.1	52.5	52.8	52.6	52.7	44.6	43.3

Table 3.2: Optimization time (seconds) as N inreases.

Alg.	$N = 1$	3	5	8	10	13	15
EV	1.8	9.2	16.1	74.7	173.4	196.8	365.5
PV	0.3	0.8	0.9	5.4	7.1	12.0	15.3
CD	95.7	635.9	4837.2	21589	21101	62054	78421
TSP	0.3	0.3	1.7	4.0	6.1	11.0	15.1
Cov	-	-	-	-	-	-	-

time required to optimize each of the graphs produced by the algorithms. The exact visibility algorithm (EV) graph optimization time complexity scales at a much slower rate than that of the graph construction, as expected since this part of the algorithm scales quadratically rather than exponentially as shown in the analytical complexity analysis on Sec. **??**. In addition, the pruned visibility algorithm requires a negligible amount of time for path planning, even with the maximum of $N = 15$ targets, the pruned visibility method is nearly as fast as the simple TSP algorithm to optimize. The Cell Decomposition algorithm, however scales quickly in this optimization step which is caused by the very large graph size associated with the cell decomposition approach. The coverage algorithm does not require optimization and therefore no time complexity is reported for that algorithm.

The total runtime, or the sum of Tables 3.1 and 3.2, is shown in Fig. 3.3. Overall, the cell decomposition approach is shown to be the most inefficient method as

Table 3.3: Total run time (seconds) as N inreases.

Alg.	$N = 1$	3	5	8	10	13	15
EV	3.4	356.5	4189.7	12463	16914	32338	52119
PV	0.7	17.2	2.3	143.0	252.0	422.6	489.8
CD	473.8	1242.8	5688.1	22674	22341	63490	79928
TSP	0.5	31.3	72.6	239.8	395.4	551.2	698.2
Cov	46.1	52.5	52.8	52.6	52.7	44.6	43.3

N is increased, followed by the exact visibility method. Interestingly, the pruned visibility method is even more efficient than the simple TSP algorithm as the number of targets is increased. This phenomena occurs because the pruned visibility method can require less nodes in the graph than the TSP algorithm by taking advantage of overlapping visibility regions. The coverage algorithm is clearly the most efficient being that it does not scale with the number of targets, although the efficiency comes at the expense of the path performance as shown in the previous subsection.

The graph size is another interesting measure of the computational requirements for the various algorithms. Tables 3.4 and 3.5 show the number of nodes and the number of edges as N increases, respectively. This representation highlights the reason for the high optimization time complexity for the cell decomposition algorithm as it requires nearly 1000 nodes for 15 targets. the exact visibility method also produces a significant amount of nodes and an incredible amount of edges for a nearly fully connected graph. The pruned visibility method is very impressive in the small graph size produces while maintaining competitive path performance. In fact, the number of nodes required for the pruned visibility method is bounded by the number of targets, i.e., $|\mathcal{E}_{PV}| \leq N + 1$, where the $(+1)$ is due to the added node for the initial configuration. This is interesting because it is therefore always

Table 3.4: Number of nodes as N increases

Alg.	$N = 1$	3	5	8	10	13	15
EV	9	34	76	110	127	175	238
PV	2	3	3	7	9	13	14
CD	442	500	560	675	709	854	906
TSP	2	4	6	9	11	14	16
Cov	-	-	-	-	-	-	-

Table 3.5: Number of edges as N increases

Alg.	$N = 1$	3	5	8	10	13	15
EV	36	392	2267	4838	6505	12610	23380
PV	1	5	3	38	66	139	163
CD	2871	3991	5182	7217	7924	10400	11640
TSP	1	2	8	26	38	67	101
Cov	-	-	-	-	-	-	-

at least as efficient, in terms of graph size, as the simple TSP algorithm.

3.11 Physical Experiments and Demonstrations

In addition to the various simulation results performed in the previous subsection, experimental validation is also performed using a DJI™Mavic 2.0 quadrotor drone as shown in Fig. 3.21. The workspaces used for the experiments are mapped and the path is planned using the exact visibility method *a priori*. The robot then starts at a known initial configuration and moves along a constant-altitude path specified by the planning algorithm. The quadrotor is equipped with multiple vision and time-of-flight depth sensors for highly accurate state estimation using the built-in DJI™functionality. In addition, the quadrotor is equipped with a GPS that works in most outdoor environments. The closed-loop trajectory following

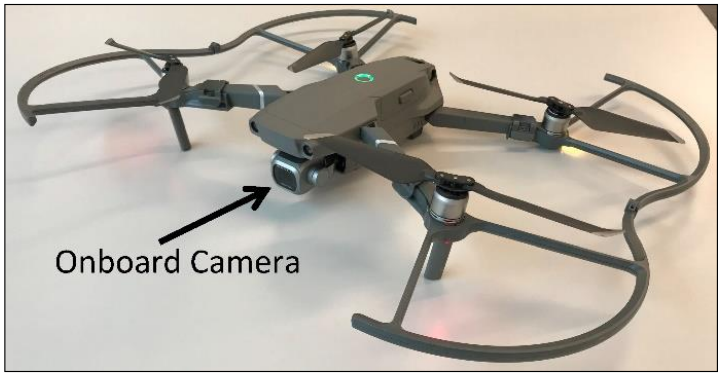


Figure 3.21: DJI Quadrotor used in physical experiments equipped with an onboard camera.

controller is implemented in Java from a Dell Alienware mobile workstation with an Intel™CPU. The state estimate is computed at a high frequency onboard the quadrotor and communicated to the workstation over a Wifi connection, where the corresponding motor control inputs are computed and communicated back to the quadrotor in real-time. Similar to the Unreal Engine™simulations, the quadrotor in this case study is tasked with classifying all of the targets in the workspace correctly using an off-the-shelf object recognition software [81]. Two experimental case studies are shown in which the quadrotor follows the exact visibility algorithm path to visit multiple targets: The first case study is an (a) indoor environment, and the second is a (b) outdoor environment.

Case Study 1: Indoor Workspace The first case study is that of an indoor workspace performed at Cornell University's Upson Hall as shown in Fig. 3.22. This workspace is a GPS denied environment requiring the use of the onboard state estimation for trajectory following by the quadrotor. The workspace involves a hallway, sitting area, and an office space, requiring traversal through a narrow passage to view all of the targets. The workspace is populated with $N = 9$ targets that are made up of various objects such as backpacks and computers. The

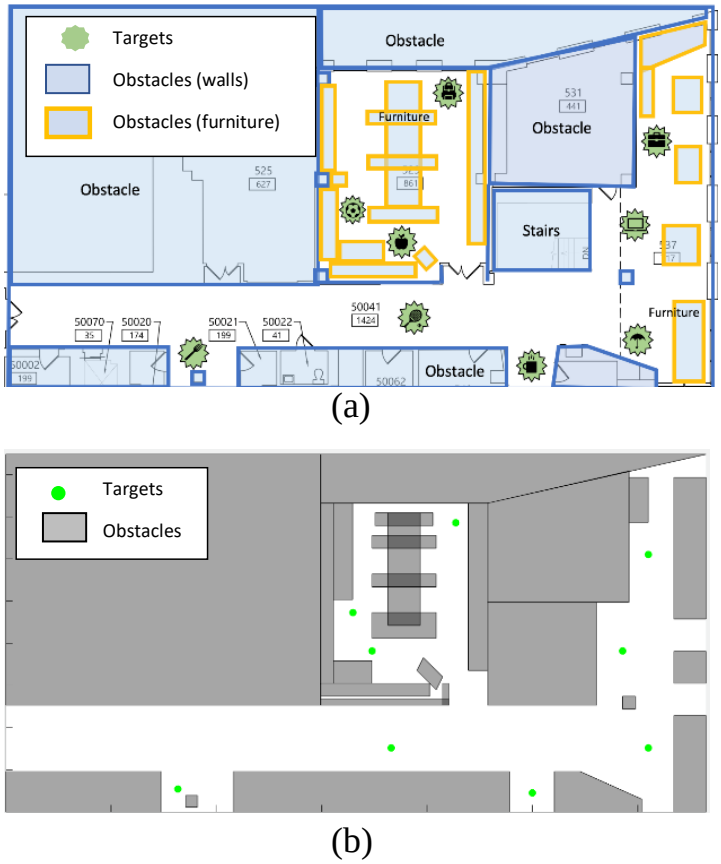


Figure 3.22: Indoor workspace for physical environments computed from schematics of the 5th floor of Upson Hall at Cornell University (a), and the simplified polygonal map used in the path planning algorithm (b).

obstacles in the workspace are modeled using $M = 26$ obstacles.

The exact visibility algorithm is employed and produces the path shown in Fig. 3.23 that successfully observes all of the targets in the workspace. The quadrotor also manages to avoid collision with all obstacles and navigate successfully through the narrow passage of the propped doorway between the office space and the hallway.

Case Study 2: Outdoor Workspace The second case study in the experimental results is performed in an outdoor environment on Cornell University's

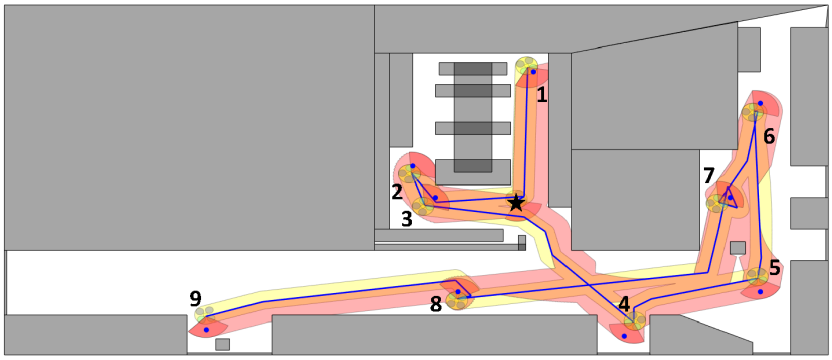


Figure 3.23: Path produced by the exact visibility algorithm on the indoor workspace.

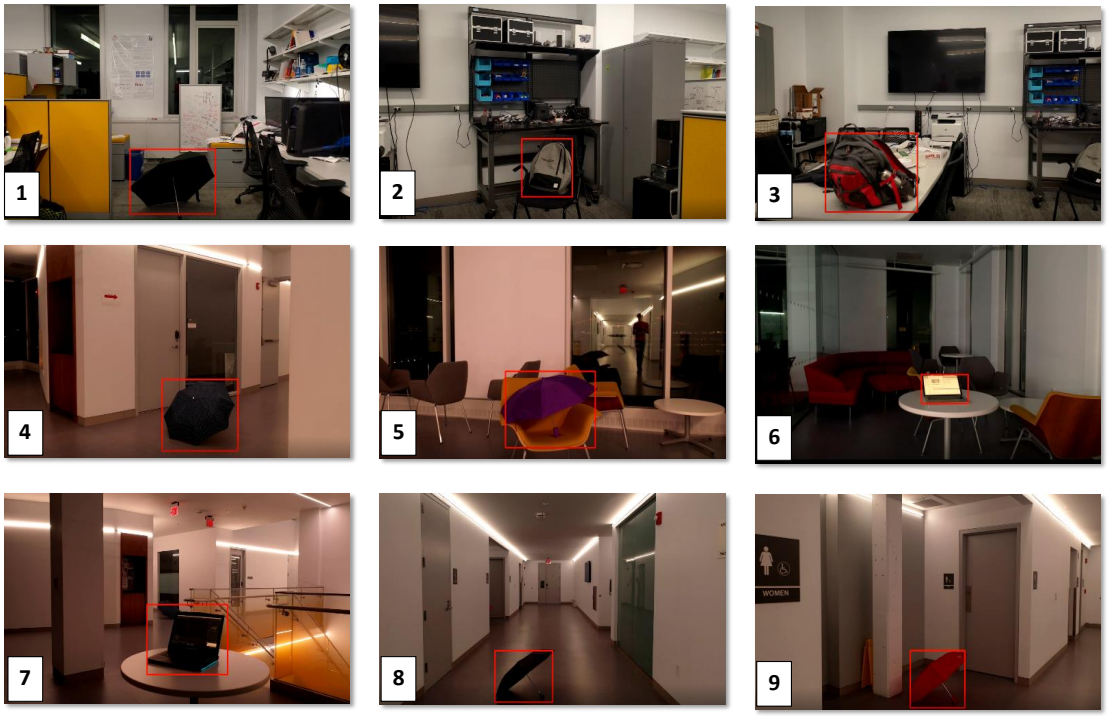
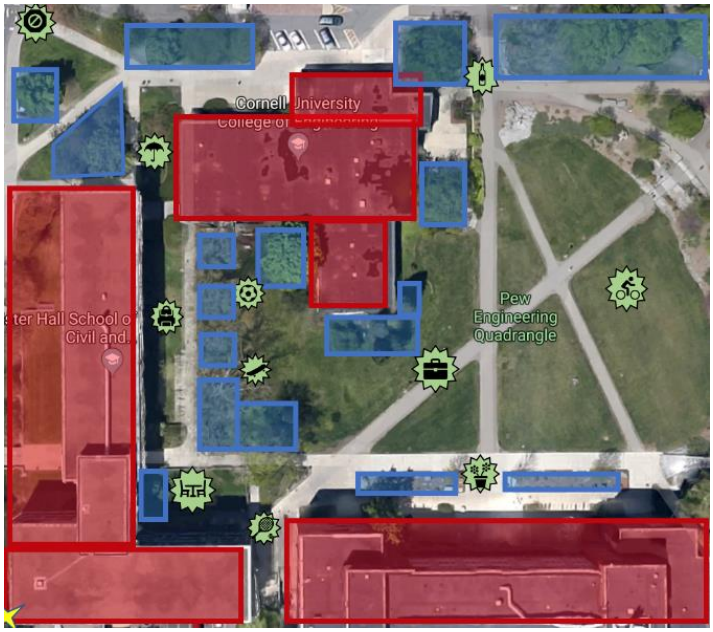
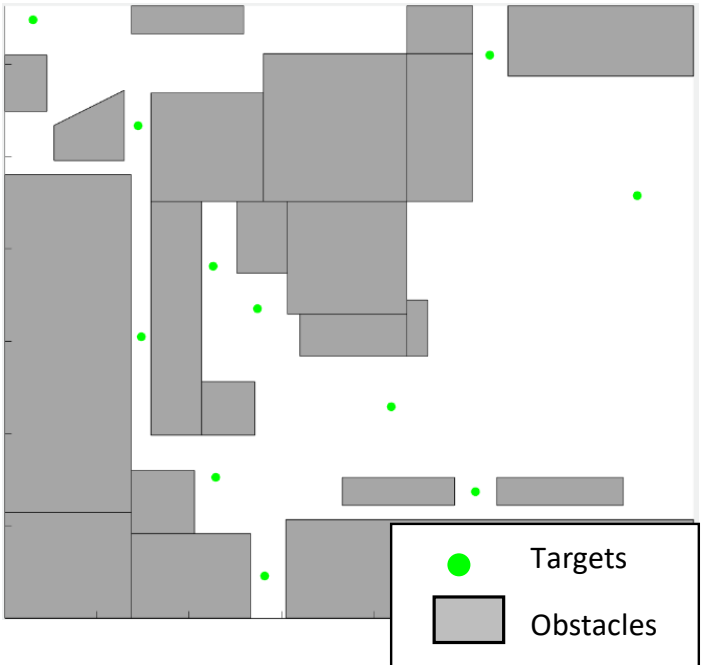


Figure 3.24: Sensor FOV images obtained from the onboard camera in the indoor workspace with correctly recognized target objects in the sensor FOV at each waypoint along the path produced by the exact visibility method shown as red bounding boxes.

Engineering Quadrangle. This workspace is a very large scale where the obstacles are actually large buildings and trees, unlike the previous experiment. The outdoor workspace contains $N = 11$ targets and $M = 21$ obstacles as shown in Fig. 3.25. The exact visibility algorithm successfully produces an adequate path for this workspace as shown in Fig. 3.26. The drone has access to GPS coordinates in this outdoor environment and can therefore successfully and easily navigate along the prescribed trajectory and correctly recognize all targets in the workspace.



(a)



(b)

Figure 3.25: Outdoor workspace used for the physical experiments at Cornell University's Engineering Quadrangle.

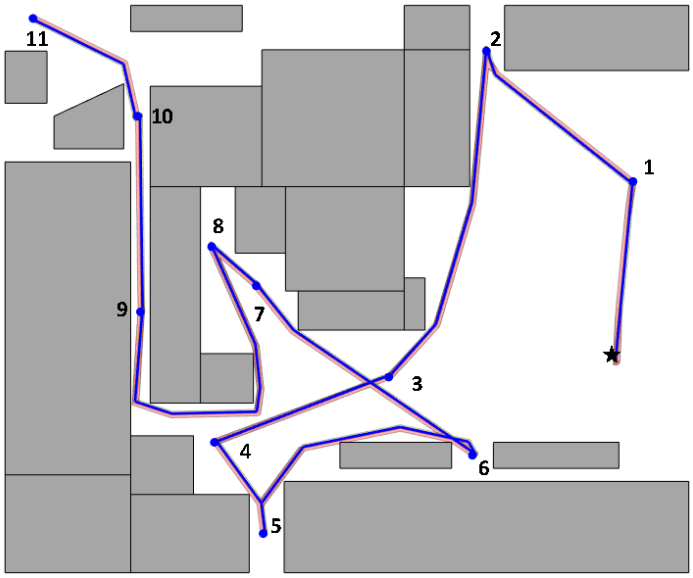


Figure 3.26: Path produced by the exact visibility algorithm on the outdoor workspace.

CHAPTER 4

CONCLUSION

A novel and systematic methodology for planning the trajectory of a mobile robotic sensor deployed to classify multiple fixed targets located in an obstacle-populated workspace is presented in this dissertation. Unlike existing robotic sensor path planning methods that are not directly applicable to robots whose primary objective is to gather sensor measurements using a sensor with a bounded FOV subject to LOS visibility, this dissertation develops a novel path planing method in which the obstacles, sensor FOV, and robotic platform, are represented as closed and bounded subsets of an Euclidean workspace. The visibility regions developed in this dissertation represent a collection of subsets of the directional sensor's configuration space that quantify the visibility of every subset of targets subject to the sensor FOV geometry and LOS visibility. The visibility regions are then used to construct a connectivity graph that represents the visibility region connectivity of the sensor's configuration space. The computational performance of the algorithm is analyzed and an approximation algorithm with significantly improved efficiency is also presented which leverages visibility regions in which multiple targets are visible. The effectiveness of the sensing strategies computed by this method are demonstrated through multiple simulation and physical experiments, in which the method developed in this dissertation is compared to traditional Traveling Salesman Problem approaches, as well as cell decomposition, and coverage methods. The proposed algorithm outperforms the comparison algorithms in both physical and simulated experiments.

This dissertation develops a novel and systematic approach to real-time three dimensional human pose estimation for *viewpoint invariant* action recognition from

a monocular camera onboard a quadrotor tasked with actively monitoring a human of interest. Leveraging recent advancements in two-dimensional human pose estimation, a novel geometric approach developed here computes a finite set of possible three-dimensional human poses given the two-dimensional image plane measured pose. This dissertation defines a three-dimensional human pose in terms of two consecutive Euler angle rotations for each pair of connected human joints, thereby eliminating the human's scale from the human's pose. The rotations used to describe the human pose are then transformed to a human-fixed coordinate reference frame resulting in a viewpoint invariant representation of the human pose. The time history of the estimated three-dimensional pose is used to construct a viewpoint invariant feature vector that captures temporal action information, which is employed for human action recognition using a Support Vector Machine. Finally, a switched controller is developed that enables the quadrotor to maintain the human in the onboard camera's field of view using the perceived action of the human. Results of the three-dimensional human pose estimation are shown to accurately represent the true pose of a human in a photo-realistic simulation environment. Additionally, the action-based switched controller is shown to maintain the human in the camera's field of view while the human is performs various actions that would typically require re-tuning of a controller's gains.

This dissertation presents a method for mobile camera control using its video feedback in real time, in order to detect and pursue a human target. Because video frames are dependent on the camera position and orientation, the interactive and highly realistic game programming environment Unreal Engine™is used to perform virtual experiments in real time. The proposed approach relies on consistent bounding box extraction to control the camera's forward speed and yaw rate to maintain the target within its FoV and at a specified distance for accurate

image processing. The simulation results show the camera tracking the human target, keeping the target within the FoV and at a reasonable distance for reliable image processing. The same control algorithm is successfully implemented on the Clearpath™Jackal robot, which also successfully follows the target and maintains it in the camera FoV. Future work includes tracking a particular human target using multiple cameras with different viewpoints in a crowded environment.

BIBLIOGRAPHY

[1] John D Albertson, Tierney Harvey, Greg Foderaro, Pingping Zhu, Xiaochi Zhou, Silvia Ferrari, M Shahrooz Amin, Mark Modrak, Halley Brantley, and Eben D Thoma. A mobile sensing approach for regional surveillance of fugitive methane emissions in oil and gas production. *Environmental science & technology*, 50(5):2487–2497, 2016.

[2] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[3] Nicholas Ayache. *Artificial vision for mobile robots: stereo vision and multisensory perception*. Mit Press, 1991.

[4] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. *Computer Vision and Pattern Recognition*, 2011.

[5] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[6] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[7] Partha S Bhagavatula, Charles Claudianos, Michael R Ibbotson, and Mandyam V Srinivasan. Optic flow cues guide flight in birds. *Current Biology*, 21(21):1794–1799, 2011.

[8] Aaron Bobick and James Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):257–267, March 2001.

[9] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Theodor Bröcker and Klaus Jänich. *Introduction to differential topology*. Cambridge University Press, 1982.

[12] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[13] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[14] Chenghui Cai and Silvia Ferrari. Information-driven sensor path planning by approximate cell decomposition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(3):672–689, 2009.

[15] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Asynchronous convolutional networks for object detection in neuromorphic cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[16] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *arXiv preprint arXiv:1812.08008*, 2018.

[17] Rizwan Chaudhry, Avinash Ravichandran, Gregory Hager, and René Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1932–1939. IEEE, 2009.

[18] Ching-Hang Chen and Deva Ramanan. 3D human pose estimation= 2D pose estimation+ matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7035–7043, 2017.

[19] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P. How. Socially aware motion planning with deep reinforcement learning. *CoRR*, abs/1703.08862, 2017.

[20] Peng Cheng, Emilio Frazzoli, and Steven LaValle. Improving the performance of sampling-based motion planning with symmetry-based gap reduction. *IEEE Transactions on Robotics*, 24(2):488–494, 2008.

[21] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3218–3226, 2015.

[22] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001.

[23] Gabriele Costante, Christian Forster, Jeffrey Delmerico, Paolo Valigi, and Davide Scaramuzza. Perception-aware path planning. *arXiv preprint arXiv:1605.04151*, 2016.

[24] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[25] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[26] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *European conference on computer vision*, pages 428–441. Springer, 2006.

[27] Jonathan Daudelin and Mark Campbell. An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-d objects. *IEEE Robotics and Automation Letters*, 2(3):1540–1547, 2017.

[28] Rika Sugimoto Dimitrova, Mathias Gehrig, Dario Brescianini, and Davide Scaramuzza. Towards low-latency high-bandwidth control of quadrotors using event cameras. *arXiv preprint arXiv:1911.04553*, 2019.

[29] Khaled Elbassioni, Aleksei V Fishkin, and René Sitters. Approximation algorithms for the euclidean traveling salesman problem with discrete and continuous neighborhoods. *International Journal of Computational Geometry & Applications*, 19(02):173–193, 2009.

[30] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[31] Davide Falanga, Suseong Kim, and Davide Scaramuzza. How fast is too fast? the role of perception latency in high-speed sense and avoid. *IEEE Robotics and Automation Letters*, 4(2):1884–1891, 2019.

[32] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40), 2020.

[33] Silvia Ferrari and Thomas A. Wettergren. *Information-driven Planning and Control*. MIT Press, 2021.

[34] Greg Foderaro, Silvia Ferrari, and Thomas A Wettergren. Distributed optimal control for multi-agent trajectory optimization. *Automatica*, 50(1):149–154, 2014.

[35] Vasiliy E Gai, Igor V Polyakov, and Olga V Andreeva. Depth mapping method based on stereo pairs. In *International Conference on Neuroinformatics*, pages 303–308. Springer, 2019.

[36] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *arXiv preprint arXiv:1904.08405*, 2019.

[37] Jake Gemerek, Taylor Clawson, Rui Liu, and Silvia Ferrari. Real-time 3d human pose estimation for viewpoint invariant action recognition. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[38] Jake Gemerek, Taylor Clawson, Rui Luo, and Silvia Ferrari. Viewpoint invariant 3d human pose estimation and action recognition for active perception. *IEEE Conference on Decision and Control (CDC)*, 2020.

[39] Jake Gemerek, Silvia Ferrari, Brian H. Wang, and Mark E. Campbell. Video-guided camera control for target tracking and following. In *2nd Annual Conference on Cyber-Physical and Human Systems (CPHS)*, 2018.

[40] Jake Gemerek, Bo Fu, Yucheng Chen, Min Zheng, Zeyu Liu, and Silvia Ferrari. Visibility-based directional sensor path planning. *IEEE Transactions on Robotics (TRO)*, 2020 (submitted).

[41] Jake R. Gemerek, Silvia Ferrari, and John D. Albertson. Fugitive gas emission rate estimation using multiple heterogeneous mobile sensors. In *Olfaction and Electronic Nose (ISOEN), 2017 ISOCS/IEEE International Symposium on*, pages 1–3. IEEE, 2017.

[42] A. Goldhoorn, A. Garrell, R. Alquézar, and A. Sanfeliu. Continuous real time pomcp to find-and-follow people by a humanoid service robot. In *2014*

IEEE-RAS International Conference on Humanoid Robots, pages 741–747, Nov 2014.

[43] Rostislav Goroshin, Quyen Huynh, and Hao-Min Zhou. Approximate solutions to several visibility optimization problems. *Communications in Mathematical Sciences*, 9(2):535–550, 2011.

[44] Renshu Gu, Gaoang Wang, and Jenq-Neng Hwang. Efficient multi-person hierarchical 3d pose estimation for autonomous driving. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 163–168. IEEE, 2019.

[45] Ibrahim A Hameed. Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain. *Journal of Intelligent & Robotic Systems*, 74(3-4):965–983, 2014.

[46] Shaoming He, Hyo-Sang Shin, and Antonios Tsourdos. Trajectory optimization for target localization with bearing-only measurement. *IEEE Transactions on Robotics*, 35(3):653–668, 2019.

[47] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[48] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.

[49] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*, 2017.

[50] Giacomo Indiveri and Paul Verschure. Autonomous vehicle guidance using analog vlsi neuromorphic sensors. In *International Conference on Artificial Neural Networks*, pages 811–816. Springer, 1997.

[51] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[52] David G Kirkpatrick and Raimund Seidel. The ultimate planar convex hull algorithm? *SIAM journal on computing*, 15(1):287–299, 1986.

[53] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[54] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[55] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[56] Laura Leal-Taixé, Anton Milan, Ian D. Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *CoRR*, abs/1504.01942, 2015.

[57] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[58] Guoliang Liu, Tiantian Liu, Guohui Tian, and Ze Ji. Distributed human 3d pose estimation and action recognition. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2316–2321. IEEE, 2019.

[59] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European conference on computer vision*, pages 816–833. Springer, 2016.

[60] Shih-Chii Liu, Bodo Rueckauer, Enea Ceolini, Adrian Huber, and Tobi Delbruck. Event-driven sensing for efficient perception: Vision and audition algorithms. *IEEE Signal Processing Magazine*, 36(6):29–37, 2019.

[61] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[62] Zeyu Liu. A cell decomposition approach to autonomous path planning for directional mobile sensors. Master's thesis, Cornell University, 2018.

[63] W. Lu, G. Zhang, and S. Ferrari. An information potential approach to

integrated sensor path planning and control. *IEEE Transactions on Robotics*, 30(4):919–934, Aug 2014.

[64] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[65] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[66] Diogo C Luvizon, Hedi Tabia, and David Picard. Multi-task deep learning for real-time 3d human pose estimation and action recognition. *arXiv preprint arXiv:1912.08077*, 2019.

[67] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015.

[68] David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 204(1156):301–328, 1979.

[69] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.

[70] MATLAB. *version 9.5.0 (R2019b)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[71] N. McLaughlin, J. M. d. Rincon, and P. Miller. Video person re-identification for wide area tracking based on recurrent neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2017.

[72] T. Nageli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges. Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters*, 2(3):1696–1703, July 2017.

[73] Toshiyuki Nakata, Nathan Phillips, Patrício Simões, Ian J Russell, Jorn A Cheney, Simon M Walker, and Richard J Bomphrey. Aerodynamic imaging

by mosquitoes inspires a surface detector for autonomous flying vehicles. *Science*, 368(6491):634–637, 2020.

[74] Alexandre Moreira Nascimento, Lucio Flavio Vismari, Caroline Bianca Santos Tancredi Molina, Paulo Sergio Cugnasca, João Batista Camargo, Jorge Rady de Almeida, Rafia Inam, Elena Fersman, Maria Valeria Marquezini, and Alberto Yukinobu Hata. A systematic literature review about the impact of artificial intelligence on autonomous vehicle safety. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[75] Carole Nissoux, Thierry Siméon, and J-P Laumond. Visibility based probabilistic roadmaps. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, volume 3, pages 1316–1321. IEEE, 1999.

[76] Narcís Palomeras, Natalia Hurtós, Eduard Vidal, and Marc Carreras. Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner. *IEEE Robotics and Automation Letters*, 4(2):1619–1625, 2019.

[77] Huy Hieu Pham, Houssam Salmane, Louahdi Khoudour, Alain Crouzil, Pablo Zegers, and Sergio A Velastin. A unified deep framework for joint 3d pose estimation and action recognition from a single rgb camera. *arXiv preprint arXiv:1907.06968*, 2019.

[78] Edward J Powley, Daniel Whitehouse, and Peter I Cowling. Monte carlo tree search with macro-actions and heuristic route planning for the physical travelling salesman problem. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 234–241. IEEE, 2012.

[79] Weichao Qui and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. *Computing Research Repository (CoRR)*, 2016.

[80] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[81] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Toward real-time object detection with region proposal networks. *arXiv preprint*, 2016.

[82] Elisa Ricci, Jagannadan Varadarajan, Ramanathan Subramanian, Samuel Rota Bulo, Narendra Ahuja, and Oswald Lanz. Uncovering interactions and interactors: Joint estimation of head, body orientation and f-formations from surveillance videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4660–4668, 2015.

[83] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *CoRR*, abs/1701.01909, 2017.

[84] Paolo Salaris, Marco Cognetti, Riccardo Spica, and Paolo Robuffo Giordano. Online optimal perception-aware trajectory generation. *IEEE Transactions on Robotics*, 35(6):1307–1322, 2019.

[85] Oren Salzman and Dan Halperin. Asymptotically near-optimal rrt for fast, high-quality motion planning. *IEEE Transactions on Robotics*, 32(3):473–483, 2016.

[86] GD Schott. The extent of man from vitruvius to marfan. *The Lancet*, 340(8834-8835):1518–1520, 1992.

[87] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. A 128 x 128 1.5 contrast sensitivity 0.9 fpn 3 microsecond latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits*, 48(3):827–838, 2013.

[88] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *CoRR*, abs/1705.05065, 2017.

[89] Mahesh Kr Singh, KS Venkatesh, and Ashish Dutta. A new next best view method for 3d modeling of unknown objects. In *2015 Third International Conference on Image Information Processing (ICIIP)*, pages 516–519. IEEE, 2015.

[90] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[91] Josef Steinbaeck, Christian Steger, Gerald Holweg, and Norbert Druml. Next generation radar sensors in automotive sensor fusion systems. In *2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–6. IEEE, 2017.

[92] A. Swingler and S. Ferrari. On the duality of robot and sensor path planning. In *52nd IEEE Conference on Decision and Control*, pages 984–989, Dec 2013.

[93] Zhijun Tang and Umit Ozguner. Motion planning for multitarget surveillance with mobile sensor agents. *IEEE Transactions on Robotics*, 21(5):898–908, 2005.

[94] H. Tsutsui, J. Miura, and Y. Shirai. Optical flow-based person tracking by multiple cameras. In *Conference Documentation International Conference on Multisensor Fusion and Integration for Intelligent Systems. MFI 2001 (Cat. No.01TH8590)*, pages 91–96, 2001.

[95] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[96] Jorge Urrutia. Art gallery and illumination problems. In *Handbook of computational geometry*, pages 973–1027. Elsevier, 2000.

[97] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[98] H. Wei, W. Lu, P. Zhu, S. Ferrari, R. H. Klein, S. Omidshafiei, and J. P. How. Camera control for learning nonlinear target dynamics via bayesian nonparametric dirichlet-process gaussian-process (dp-gp) models. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 95–102, Sept 2014.

[99] Hongchuan Wei and Silvia Ferrari. A geometric transversals approach to sensor motion planning for tracking maneuvering targets. *IEEE Transactions on Automatic Control*, 60(10):2773–2778, 2015.

[100] Hongchuan Wei, Wenjie Lu, Pingping Zhu, Guoquan Huang, John Leonard, and Silvia Ferrari. Optimized visibility motion planning for target tracking and localization. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 76–82. IEEE, 2014.

[101] Guoxian Zhang, Silvia Ferrari, and Chenghui Cai. A comparison of information functions and search strategies for sensor planning in target classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(1):2–16, 2011.

[102] Yi Zhang, Weichao Qiu, Qi Chen, Xiaolin Hu, and alan L. Yuille. Unreal stereo: A synthetic dataset for analyzing stereo vision. *Computing Research Repository (CoRR)*, 2016.

[103] Min Zheng. A probabilistic approach to autonomous path planning for directional mobile sensors. Master's thesis, Cornell University, 2018.

[104] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018.

[105] Pingping Zhu, Jason Isaacs, Bo Fu, and Silvia Ferrari. Deep learning feature extraction for target recognition and classification in underwater sonar images. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2724–2731. IEEE, 2017.

[106] Hanqi Zhuang, Raghavan Sudhakar, and Jen-yu Shieh. Depth estimation from a sequence of monocular images with known camera motion. *Robotics and Autonomous Systems*, 13(2):87–95, 1994.