Towards Active Object-Based Navigation

by

S. Violet Killy

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author . . . .
Department of Mechanical Engineering
Aug 16, 2022

Certified by . . . .
John J. Leonard
Samuel C. Collins Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by . . . .
Nicholas Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

Towards Active Object-Based Navigation

by

S. Violet Killy

Submitted to the Department of Mechanical Engineering
on Aug 16, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This thesis investigates the design and implementation of an active object-based navigation system. Localization refers to the capability of a mobile robot to determine its position relative to a prior map of the environment. Localization based on fiducial markers, such as AprilTags, has been a popular technique used widely in many laboratories for over a decade; however, it is desirable to enable a mobile robot to navigate based on the objects that occur naturally in a given environment. The goal of using real objects as "geometric beacons" for robot navigation was identified many years ago. In practice, however, it has been challenging to develop metrically accurate robot localization systems that use semantic, object-based maps. Recent years have seen tremendous progress in the use of machine learning techniques to recognize objects from camera images, including 6 degree-of-freedom (DOF) object pose estimation. Using these machine learning capabilities for robot navigation is an important application, yet research in creating real-time object-based localization systems has been limited. One of the difficulties is that to obtain accurate 6DOF object pose estimates, typically the object needs to be centered in the field of view of the camera, and viewed from a favorable viewing distance. To help meet this need, this thesis has developed a pan-tilt actively controlled camera mount, and deployed it on a small mobile robot, to explore capabilities for improved object-based navigation. By running a detector in real-time as the robot navigates through the world, the active pan-tilt head can actively track an object of interest; 6DOF pose estimates from the object are used to estimate the robot's pose. The performance of the system is analyzed using ground truth provided by an OptiTrack motion capture system. Future work will extend the system to track multiple objects and to perform active SLAM.

Thesis Supervisor: John J. Leonard
Title: Samuel C. Collins Professor of Mechanical and Ocean Engineering

Acknowledgments

I would first like to thank my advisor, Professor John Leonard, for his guidance and endless support throughout this process. I sincerely appreciate his flexibility throughout the COVID-19 pandemic, and his willingness to always find the time to check in and give much needed advice no matter how much he is juggling. His passion for teaching, contagious excitement for his field, and clear care and willingness to go to bat for all of his students goes above and beyond - for that I am deeply grateful.

I also would like to thank Steve Banzaert for his invaluable technical support, and for always helping me find the path to the finish line. His years of mentorship and encouragement have been one of the highlights of my time at MIT, and have deeply influenced my approach to problem solving and my passion for engineering.

A huge thank you as well to my fellow students and labmates Kevin Doherty and Kurran Singh for their endless pep talks and altruism throughout my Master's. I would also like to thank Ziqi Lu for his technical input and willingness to consult throughout this project, and Pablo Hu for his eager and valuable assistance with mechanical assembly and data collection.

Lastly, I would like to thank my parents, Rebecca and Peter Killy, and my sister Sophie for championing me throughout my degree. I would not have gotten to where I am without their love and endless support, and I am eternally grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Outline

This thesis focuses on the design and implementation of an active object-based navigation platform. This chapter introduces the motivation behind the system and its potential usefulness, and discusses relevant work. Chapter 2 goes through the design of the system in technical detail, discussing each of its critical sub-components, its control scheme, and its network architecture. Chapter 3 presents the results of initial experiments, comparing the performance of various object detection algorithms. Chapter 4 presents a summary and discusses possibilities for future development.

1.2 Background and Motivation

Simultaneous Localization and Mapping (SLAM) is the process of mapping the geometric features of an unknown environment, and concurrently estimating the location of a mobile agent within that map [17]. Existing approaches to the SLAM problem are highly dependent on available sensor information and computational resources. One approach that has become increasingly popular is to leverage the rich visual information from camera sensors. Though evaluating camera data is computationally expensive, recent advances in CPU and GPU technology have made real-time implementation of computer vision much more feasible. Incorporating vision in SLAM algorithms can drastically improve the accuracy and reliability of the localization estimates.

Accurate localization is vital for the safe navigation of potentially hazardous terrain. For visual-based localization methods, that accuracy is dependent on the amount of texture available in a scene. Visual SLAM is therefore much more difficult in smooth-terrain or low-light environments. For robots tasked with navigating the sandy surface of Mars [25], poorly-illuminated stretches of space [22], or sparse bodies of water [12], maximizing the amount of actionable visual information boils down to a deceptively simple question: how do you determine where to look next? The problem then becomes one of *active vision*, a subset of active perception [2].

1.2.1 Active Vision

Active vision is generally defined as the study of perception-focused modeling and control strategies. This is notably distinct from "active sensing," a term commonly seen in computer vision and robotics literature. Active sensors are those that emit a signal (e.g. light, laser, ultrasonic, etc.) and measure the reflected response from the environment. Active vision, however, deals with the process of employing passive sensors (those that receive emitted signals, but do not transmit), such as cameras, *actively*, a sensing approach which more closely resembles human perception. In nature, vision is a highly intentional and exploratory process: we are constantly adjusting and adapting our eyes and bodies to better view the world. More generally stated, we adhere to a sensing strategy, one which optimizes the intake of information. Active vision is simply the robotic parallel – real-time sensor manipulation and path planning, based on stored models, to maximize efficiency in pursuit of a goal.

Active vision techniques can be an incredibly useful tool for vision-based SLAM. Developing robust and accurate SLAM algorithms involves evaluating their performance on real-world data. Collecting useful data with passively controlled sensors can be a tedious and repetitive process for both autonomous and manually operated robots. Incorporating active vision optimizes the information being gathered, resulting in the quicker collection of more useful data.

Davison's 1998 PhD thesis at the University of Oxford [7] was an important early work in mobile robot navigation using active vision. Davison's system used the Yorick active pan-tilt stereo camera head developed by Sharkey et al. [28]. Using the Shi and Tomasi feature detector [29], Davison created a real-time active visual navigation system that was many years ahead of its time. Strong results were obtained using traditional feature-based computer vision techniques, by focusing the camera's attention on selected features of interest during the course of a navigation run. A complete visual SLAM system was demonstrated using computational resources that were extremely limited compared to what is available today. The superiority of the active vision approach was clearly demonstrated.

However, in many ways the active vision approach championed by Bajcsy, Davison, and others has fallen out of favor in modern robotic vision system development. Often, computer vision algorithms are applied to curated data sets, in which a human has guided the visual data collection process, rather than "turning a robot loose" to make its own observations of the world. Building truly autonomous, self-learning robots will require robots that can collect their own data, choosing "where to look" to find interesting objects and to self-navigate. Now is an interesting time to revisit historical active visual navigation objectives, as typified by Davison's seminal thesis, combining today's state-of-the-art 6DOF object detection and pose estimation techniques with modern pose graph robot localization capabilities.

This thesis focuses on the design and development of such a system, for use in object-based visual navigation experiments, by actively steering the pan and tilt angles of a mounted camera to keep a target object in the field of view (FOV). Another relevant active vision approach (and an interesting possibility for future research) is the active tuning of a camera's *intrinsic* parameters, as explored by Micheloni and Foresti [19].

1.3 Related Works

1.3.1 Visual Servoing Systems

Active vision goes hand in hand with visual servoing: the use of machine vision to perform closed-loop position control of a robot end-effector [14]. One of the major classifications of visual servoing systems distinguishes between image-based and position-based control schemes. [5, 26].

In image-based servoing, control values are computed using features extracted from the image. This technique is often seen in surveillance applications, with Pan Tilt Zoom (PTZ) cameras being used to track a moving target [9, 27]. Such systems often rely on tried and true feature extraction and optical flow algorithms [13, 29, 30], for estimating the motion of the target or region-of-interest (ROI). In addition to controlling the pan and tilt of a camera, active control of the camera's focal length has been used in conjunction with structure from motion (SFM) based estimations of object scale [8, 33]. This approach not only keeps the moving target centered in the FOV, but keeps its size consistent across frames.

In position-based servoing, feedback uses a combination of extracted image features, intrinsic camera parameters, and known geometric models of the target object. Together, these are used to estimate the target's 6-DOF pose with respect to the camera, and that pose estimate is what determines the control values. The error signal is then defined in coordinates of the task-space rather than directly in terms image features. While this a more computationally expensive process, the position based controller is not as highly coupled as its image-based counterpart. It is, however, more sensitive to camera calibration errors. Position-based visual servoing also depends on the availability of a known target object model, so feature-based approaches are more appropriate in situations where no such model is available. This has historically been the approach for space operations, for instance [11, 15].

The "right" visual servoing approach or feature extraction method to use is highly situation-dependent. The task is made even more complicated if dynamics are introduced. Visual servoing systems are limited by the frame rate of the camera, which can introduce a delay into the feedback system. Simple proportional controllers are often implemented (and tend to be sufficient) for stationary targets. To achieve decent performance when tracking a moving object, however, more care must be taken when designing the controller.

Since the system outlined in this thesis was intended for use in object-based navigation experiments (with known, stationary target objects), a position-based visual servoing approach was used in conjunction with a simple proportional-derivative (PD) controller. This is discussed in greater detail in the next chapter.

1.3.2 Object Pose Estimation

Object-based SLAM systems incorporate 6-DOF pose predictions of known visual landmarks in the construction of their maps. State-of-the-art pose estimation networks [16, 31, 32] rely on large amounts of training data, but real-world images annotated with high accuracy pose information are limited in availability and time-consuming to generate. Recent advances have made synthetic data an inexpensive and promising alternative for training learning-based estimators. The largest hurdle for synthetic data generation is what is referred to as "bridging the reality gap." Networks trained on synthetic data typically do not behave as expected or perform as well when placed in novel real-world environments. One major step towards spanning that gap, however, is the emergence of photo-realistic synthetic data, as introduced by Nvidia's Deep Object Pose Estimator (DOPE) [31].

DOPE uses a combination of photorealism and domain randomization to rapidly generate large quantities of high quality synthetic labeled training data. The DOPE network is trained on the popular YCB set of object models [4, 3], and achieves state-of-the-art 6-DOF object detection and pose estimation performance when tested in novel environments. To further boost performance of both object pose estimation and object-level SLAM, Lu *et al.* [18] introduced a SLAM-supported self-training procedure for fine tuning pose estimations during robotic navigation.

The active object-based navigation system outlined in this thesis used both fiducial markers (specifically AprilTags [23]) and YCB objects as targets in various experiments. Detection and pose estimation for YCB objects was performed using the DOPE detector [31]. The visual servoing module that was implemented in the system successfully tracked both classes of target object and kept them in the camera's FOV. This system could be a useful tool for conducting future object-level SLAM experiments.

Chapter 2

Robotic Navigation System Design

2.1 Mobile Robot Platform

The robotic base of the system is the Jackal Unmanned Ground Vehicle (UGV) developed by Clearpath Robotics (see Figure 2-1) [6]. This four-wheeled, rover-style mobile robot is marketed as an out-of-the-box robotics research platform, and comes equipped with GPS, Wireless and Bluetooth modules, an Inertial Measurement Unit (IMU), and an internal x86 PC running Ubuntu, all of which are integrated with the Robot Operating System (ROS). The Jackal UGV is small, rugged, and comes with a mounting platform designed for easy mechanical integration of custom payloads.

The Jackal has four fixed wheels, with the left and right pairs belted together such that they are constrained to move at the same velocity. The Jackal can then be analyzed as a two-wheeled differential drive system for the purposes of tracking odometry. The wheels are driven by two motors, each equipped with a set of optical encoders. The robot's key specifications are summarized in Table 2.1. This sort of commercial solution dramatically reduces the startup time for conducting experiments.

The Jackal has a dedicated set of existing ROS packages for simulation, navigation, and control. When the Jackal's internal computer is powered on, the top-level ROS launch file is started automatically, and subsequently calls lower-level launch files to establish WiFi communication and start the integrated sensors. The Jackal's

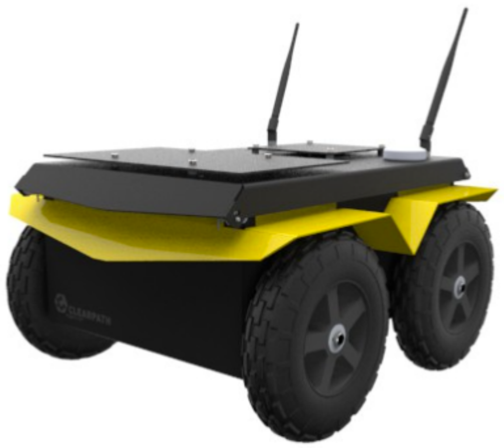


Figure 2-1: A depiction of the Jackal Unmanned Ground Vehicle (UGV) from Clearpath Robotics, which was used as a mobile robot base for this iteration of the system.

External Dimensions	508 x 430 x 250 mm (20 x 17 x 10 in)	**Battery**	270 watt-hours Lithium Ion
Weight	17kg (37lbs)	**User Power**	5V at 5A 12V at 10A 24V at 20A
Max. Payload	20kg (44lbs)	**Max.Speed**	2.0m/s

Table 2.1: The table above lists the relevant technical specs for the Jackal UGV. [6]

initial position on wake is captured and used as a reference for subsequent odometry calculations, and also serves as the fixed reference frame (/base_link) for the Jackal's various coordinate frame transforms. All of these transforms are tracked over time and published to the \tf ROS topic.

The Jackal's navigation stack makes heavy use of the move_base ROS package, a navigation tool that generates plans for a mobile base to navigate to a global goal position. This package can be used with or without an environment map. Given a tolerance specified by the user, the ROS node uses available odometry information (and laser scan data if present) to output safe velocity commands for 2D navigation. Alternatively, the user can send velocity commands to the robot manually, using a handheld controller connected to the Jackal via Bluetooth. By designating the Jackal's IP address as the ROS Master, a remote computer on the same network can access and share ROS topic information with the Jackal. The remote computer can also be used to SSH into the Jackal's onboard computer for streamlined control.

2.2 Position-Based Visual Servoing Pan-Tilt Module

Initial plans for the data-collection system incorporated an off-the-shelf three-axis camera-stabilizing gimbal. Due to various hardware and software difficulties, however, this was replaced by a simple servo-driven pan-tilt system, pictured in Figure 2-2. The pan-tilt module was driven by two goBilda 2000 Series Dual Mode Servo motors (see Figure 2-3), which can easily be programmed into one of two different control modes. The default mode operates on position feedback control, in which a PWM signal is used to control the servo's angular position with up to 300° of travel. When toggled into continuous rotation mode, the servo uses proportional speed control, with speed commands based on the sent PWM signal. For this thesis, both the pan and tilt servos were used in the default configuration and commanded via position control. The motors were run at 5V, with power drawn from the Jackal's wired 5V supply. These particular motors were chosen for their technical specifications, which are described in Table 2.2.

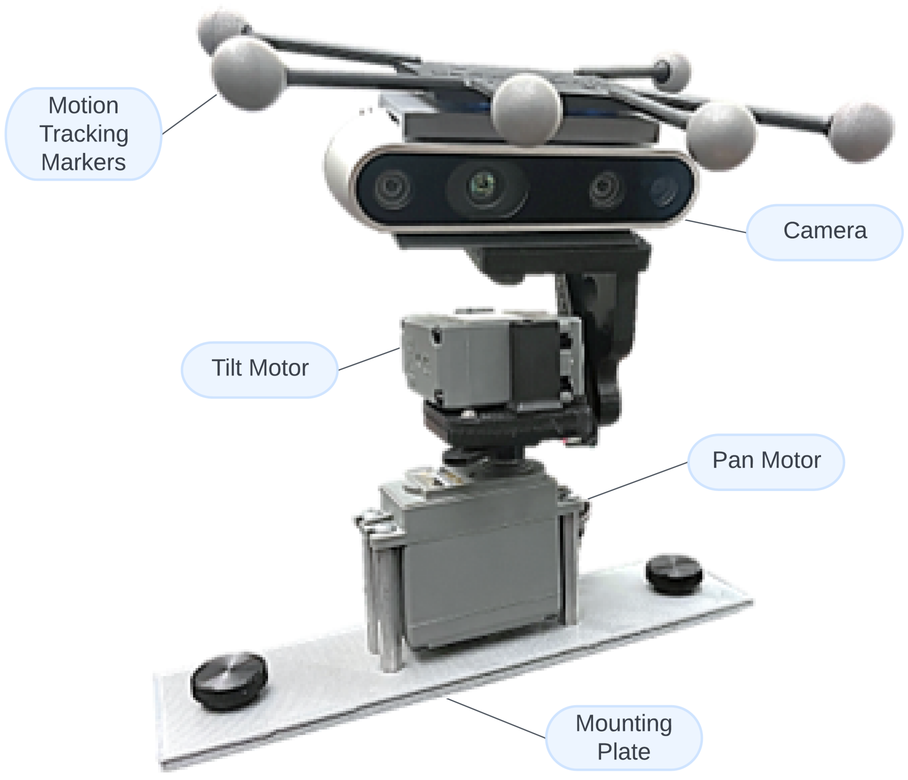


Figure 2-2: The above figure shows a labeled depiction of the visual servoing module. Two goBilda Servo Motors are stacked in a pan-tilt configuration, with a mounted RGB-D camera. The camera is topped with a set of retro-reflective markers for motion tracking, and the whole assembly is affixed directly to the Jackal using 3D-printed mounting plate.

Figure 2-3: The visual servoing module used two goBilda 2000 Series servo motors, like the one shown above, to actively steer the pan and tilt angles of the camera.

Weight	58g
Gear Ratio	135:1
Voltage Range	4.8V $\sim$ 7.4V
No-load Speed (4.8V)	0.11 sec/60° (90RPM)
Stall Torque (4.8V)	7.9 kg.cm (110 oz-in)
Max PWM Range (Default Mode	500 - 2500μs
Travel per μs (Default Mode)	0.15°/μs

Table 2.2: The table above lists the relevant technical specifications for the goBilda servo motors used. [10]

2.2.1 USB Servo Controller

The pan-tilt system's two servo motors were connected to a Micro Maestro 6-Channel USB Servo Controller from Pololu, shown in Figure 2-4. This device supports USB, TTY Serial (5V), and internal scripting control methods. Each of the Micro Maestro's six channels has individual built-in speed and acceleration control, and can be configured as a servo output, electronic speed control, digital output, or analog input. For the purposes of these experiments, the assigned pan and tilt channels were configured as servo outputs and controlled via USB. The USB cable was connected to the Jackal's onboard computer.

The Maestro device outputs precise servo pulse signals, with $0.25\mu s$ pulse width resolution, making it well-suited for multi-servo robotic applications. To execute commands, Maestro has its own stack-based scripting language. To make the code more comprehensible for debugging, a modified version of a custom python class was used to run various Maestro scripts over USB serial. A ROS node would receive the desired pan or tilt motor positions, and call the methods in this python class in order to access motor information, or to send Maestro commands to the corresponding motor channel.

It is important to note that the Maestro PWM signals were sent in units of quarter-microseconds. The goBilda motors used had a travel of 0.15° per microsecond, and maximum travel of 300° in the default control mode. The pan motor was free to travel the entirety of this distance, but the tilt motor was constrained due to the pan-tilt module's mechanical configuration. To avoid stalling the motors, the dedicated ROS

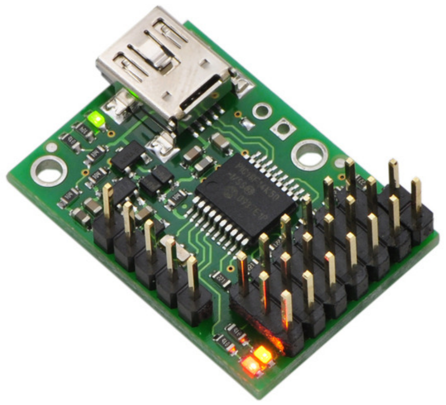


Figure 2-4: The above figure depicts the Micro Maestro 6-Channel USB Servo Controller. This device was used to send USB servo commands from the Jackal to the motors to control the pan and tilt angle of the camera.

node would convert the desired angular positions to quarter-microseconds, and clamp the resulting value to the known bounds of each axis before sending any commands to the Maestro controller.

2.2.2 Visual Servoing Control Algorithm

The pan-tilt module was controlled using position-based visual servoing techniques. The motor commands were determined by a simple proportional-derivative (PD) control algorithm. The reference signal for the controller was a tuple containing the target object's estimated pose relative to the camera frame. The goal was to keep the target object centered in the camera's FOV, so the error metric at any point was the distance from the camera frame's center (determined by the known depth and width of the image stream) to the projected pixel coordinates of the target object's centroid. For a given error term, e, the resulting command, U' was then:

$$U' = U + K_p * e + K_d \frac{de}{dt}$$

Where U is the current outgoing motor command (accessed via the Maestro) for the given channel, K_p is a proportional control parameter, and K_d is a derivative control parameter. K_p and K_d were tuned empirically.

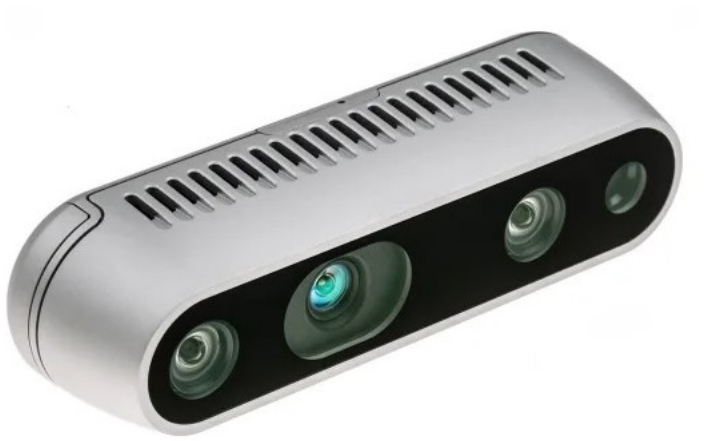


Figure 2-5: The Intel RealSense D435 RGB-D camera, shown here, is a stereoscopic depth camera that was mounted on top of the pan-tilt module. The visual data from the camera was used for object detection and 6-DOF pose estimation.

Dimensions	**Connector**
90mm x 25mm x 25mm	USB-C 3.1
Depth	**RGB**
FOV: 87° x 58°	FOV: 69° x 42°
Resolution: Up to 1280 x 720	Resolution: 1920 x 1080
Frame Rate: Up to 90fps	Frame Rate: 30fps

Table 2.3: The table above lists relevant technical specifications for the Intel RealSense D435 camera.

2.3 RGB-D Camera

The camera mounted to the pan-tilt module was an Intel RealSense D435, pictured in Figure 2-5. The D435 is a stereoscopic depth camera well-suited for high-speed robotic applications, with a wide field of view (FOV), a global shutter, and a range of up to 10m. It features four distinct imaging modules: a left and right IR-camera pair for streaming depth information, an RGB module for mapping color image data to depth data, and a laser IR dot projector for increased depth information accuracy. The D435's technical specifications are summarized in table 2.3.

2.4 System Assembly

The pan-tilt module was mounted directly to the Jackal using a 3D-printed bracket. The D435 camera was attached to the pan-tilt, with its neutral position configured such that it was facing the side of the Jackal. In this configuration, the camera was free to point inwards as the base circumnavigated a target object. The Maestro device was also mounted directly to the Jackal, and the pan and tilt servos were wired to two of its channels. The Maestro USB cable was connected to the Jackal's computer. The D435, however, was connected to a separate remote laptop running Ubuntu. While the Jackal is capable of streaming the video data, its onboard computer does not have enough processing power to efficiently detect the target objects. So for this iteration of the system, the camera was tethered to a remote workstation, configured to be a ROS client of the Jackal. The final system assembly is pictured in Figure 2-6. Note the addition of masking tape to the outside of the wheels. The experiments were performed in a carpeted room, and the interaction between the tire tread and the carpet caused the Jackal to shake aggressively as it rotated, resulting in poor quality video data. Adding tape to the tires is the official recommended solution from Clearpath Robotics, and the addition resulted in much smoother robot movements.

2.5 Motion Capture System

The experiments were conducted in a room fitted with an OptiTrack motion capture system [24], which was used to obtain ground truth measurements. The setup consisted of 6 OptiTrack Prime-13 cameras (see Figure 2-7a) mounted in the corners and sides of the ceiling. Each camera was connected to a central Power over Ethernet (PoE) Switch, which distributed data and power to each device while also transmitting data to a dedicated host PC running Ubuntu. Due to the camera system's large data bandwidth, the host PC was running on an isolated network.

The OptiTrack system utilized a motion capture software, Motive 3.0, for calibration and tracking operations. Any object intended to be tracked, such as the Jackal

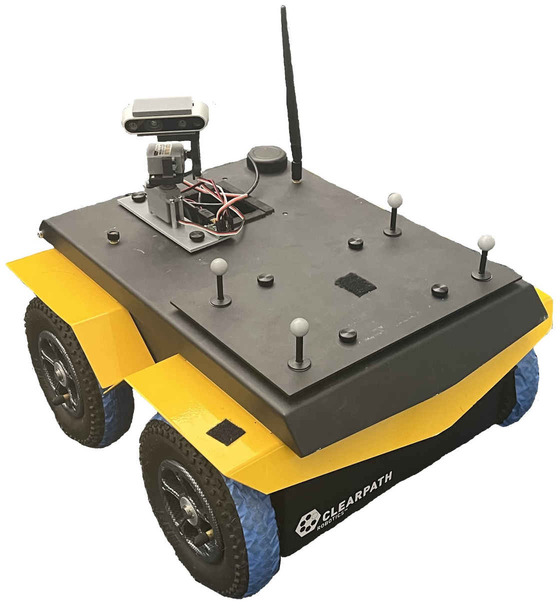


Figure 2-6: The above figure depicts the object-based navigation system described in this chapter in its fully assembled state.

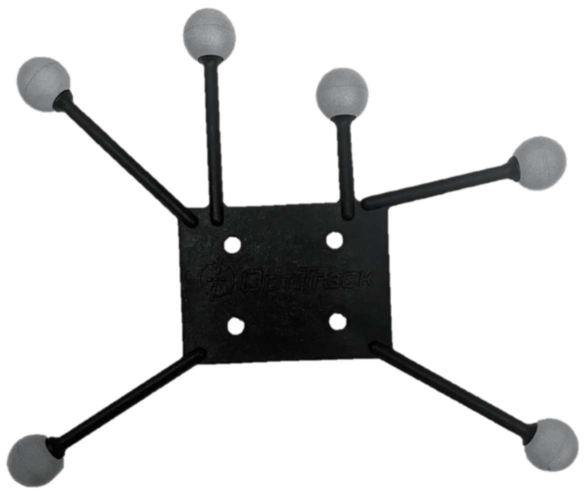
or camera, was outfitted with a series of special retro-reflective markers (see Figure 2-7b) arranged in a unique, asymmetrical configuration, as in Figure 2-7c. Once a marker configuration was calibrated in the software, the associated object's position and orientation could be precisely tracked and published to the network.

VRPN

This thesis used the Virtual Reality Peripheral Network (VRPN) libraries, a toolset for implementing network interfaces for peripheral hardware devices. The dedicated PC acting as the host computer for the peripherals (in this case the various tracking cameras) publishes data to a VRPN Server. An existing ROS wrapper, vrpn_client_ros, was used to establish a VRPN Client to access the messages being published to the server. The client was run on a separate laptop, which was concurrently communicating with the Jackal. As the client was implemented with ROS, the tracking data for each object could be published in the form of ROS Pose messages, or as separate coordinate frames in the \tf tree. A diagram illustrating the network architecture and the flow of information in the system is shown in Figure 2-8

(a) Six OptiTrack Prime-13 cameras were used for motion capture of various calibrated target objects.

(b) A depiction of a single retro-reflective marker used tracked by the motion capture system.



(c) This multi-marker configuration was used for tracking the position of the camera.

Figure 2-7: Various components used for motion capture.

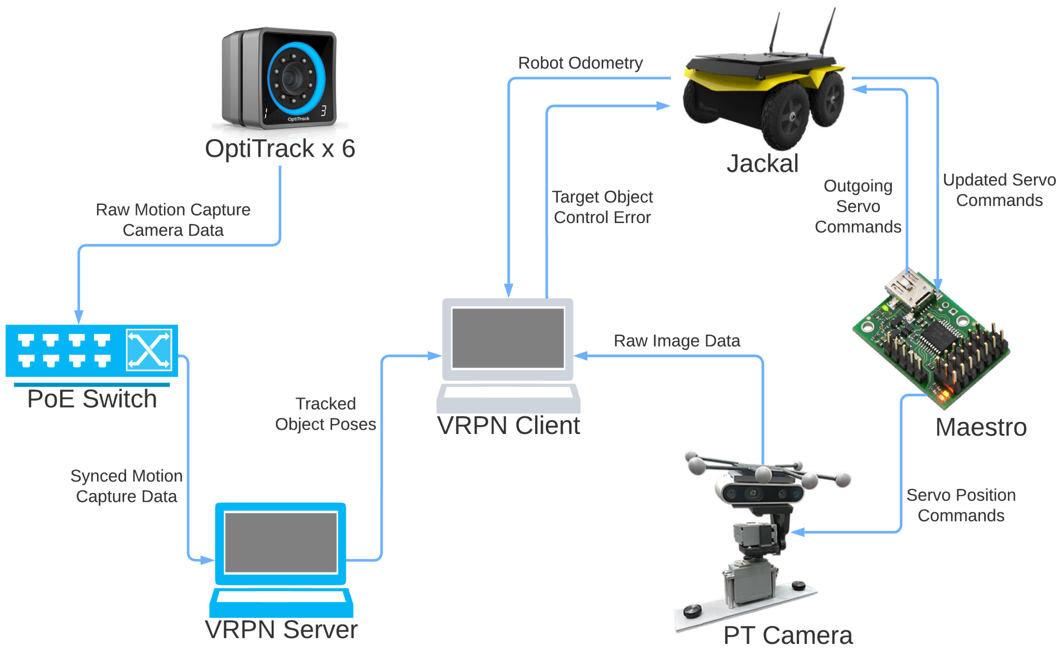


Figure 2-8: The above diagram shows the full network architecture for the system described in this thesis. Raw data from six motion capture cameras is collected and synced by a PoE switch device, which is sent to a remote computer acting as a VRPN server. A separate computer formatted as a VRPN client receives that data via ROS as a series of pose messages. Concurrently, the VRPN Client subscribes to and logs the Jackal's odometry estimates, and raw camera data from the D435. The latter is processed by pose estimators and is converted into an error metric which is sent back to the Jackal. The Jackal uses that error in conjunction with the outgoing commands read from the Maestro, and sends an updated set of servo commands to the pan-tilt module via the Maestro.

Chapter 3

Experiments

The assembled system's performance was tested through a series of experiments in which the robot, manually driven via handheld controller, was used to circumnavigate a chosen target object. The raw camera data from the D435 was used for 6-DOF object pose estimation, which in turn was used to generate visual servoing commands that would maximize the object detections. The object pose estimations were used to extract the estimated pose of the D435 camera, which was then compared to a ground truth camera trajectory from the motion capture system.

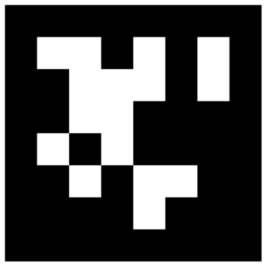


Figure 3-1: The above pattern is a fiducial marker, specifically tag 0 from the 36h11 family of AprilTags. This tag was used as an object target for the first series of experiments in this thesis.

3.1 AprilTag Detection and Pose Estimation

The first set of experiments was collected with an AprilTag fiducial marker as a target object [23]. The specific AprilTag family that was used was 36h11, and focused on the tag with ID 0, shown in Figure 3-1. To extract the pose used to generate visual servoing commands, a simple script was implemented which coupled OpenCV image processing and a python-based tag detector. To analyze the collected data in post-processing, this thesis used apriltag_ros, a ROS wrapper for a popular visual fiducial detection algorithm. Figure 3-2 illustrates how the pan-tilt camera automatically tracked the AprilTag during circumnavigation. A set of example images with labeled pose estimates (as seen by the actively steered camera) is shown in Figure 3-3.

The robot circumnavigated the tag for one, three, or five laps in each trial. As a "stress test" of the object pose estimation, and to motivate the potential benefits of the active-object navigation approach for an underwater application, we repeated the experiment with the AprilTag detector in a dark room. A flashlight was affixed to the pan-tilt module such that it would travel with the camera and act as a spotlight. Figure 3-4 shows typical detections for this situation, including a handful of outlier detections. Under these conditions, the AprilTag detector performance is slightly degraded, however the system still achieved good object tracking.

As shown, the pan-tilt module was able to actively center the AprilTag target from every viewing angle along its trajectory in both the well-lit and poorly-lit environment. The AprilTag's high-contrast pattern and non-reflective surface allow for robust detection in both scenarios. A bird's-eye view of the camera pose as estimated by the AprilTag detector throughout the trajectory is shown in Figure 3-5, and plotted against the ground truth camera position taken from the motion capture system during each trial. A circular marker in each plot shows the location of the target object during the experiment. The detector's estimated camera trajectory was compared to that of the motion capture system, using euclidean distance as an error metric. The average distance error is recorded in Table 3.1.

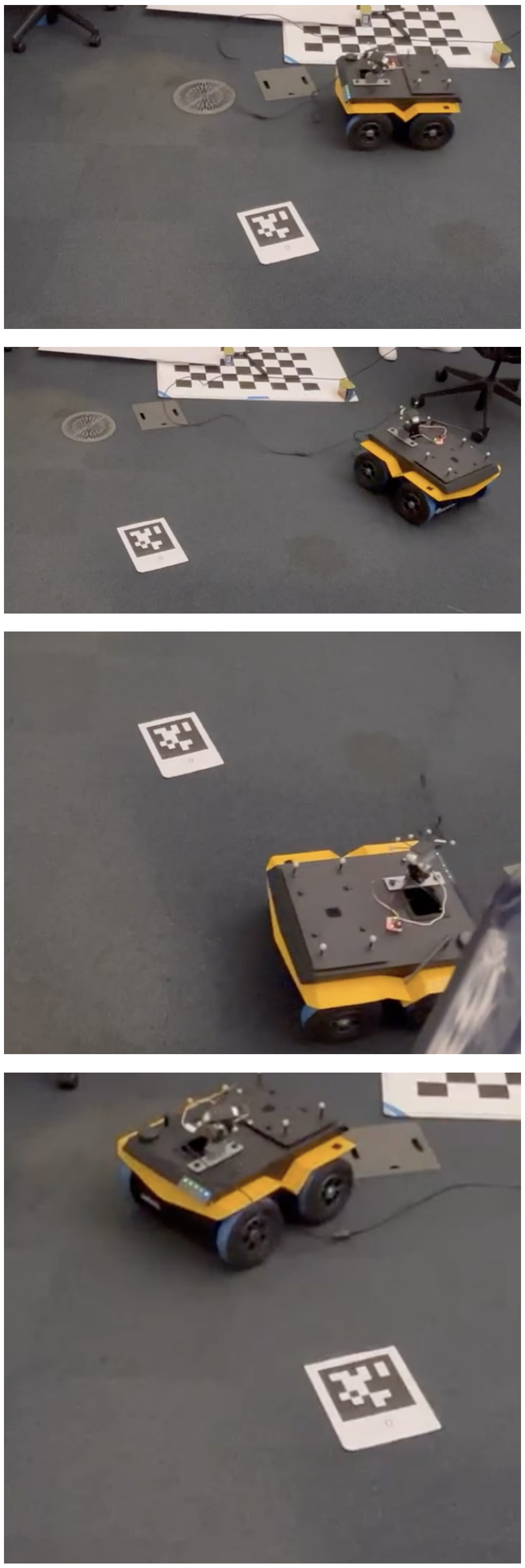


Figure 3-2: Several views of the pan-tilt camera tracking an object during circum-navigation.

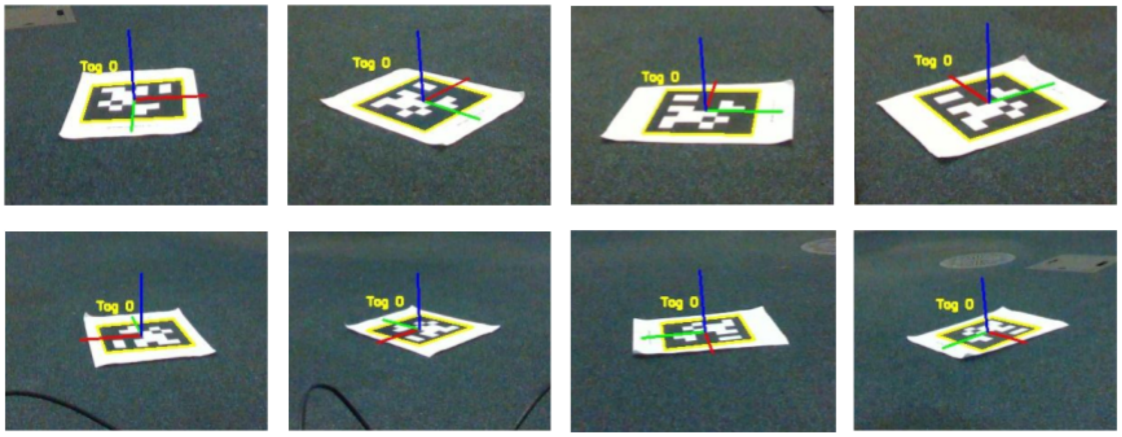


Figure 3-3: Above is a collection of images labeled with 6-DOF pose estimations as seen by the D435 camera while the Jackal circumnavigated an AprilTag.

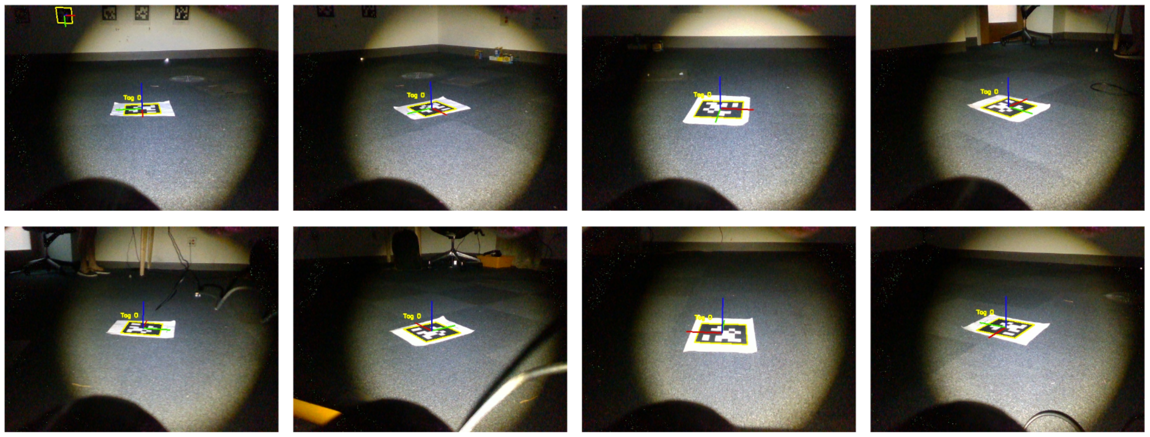
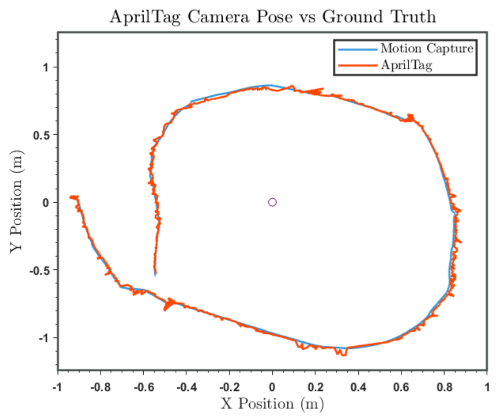


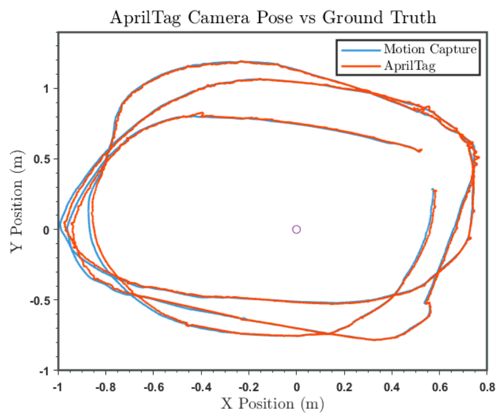
Figure 3-4: Above is a collection of images labeled with 6-DOF pose estimations as seen by the D435 camera while the Jackal circumnavigated an AprilTag in low-light conditions.

	1 Lap	**3 Laps**	**5 laps**	**Dark**
Mean Error(m)	0.0107	0.0091	0.0098	0.0106

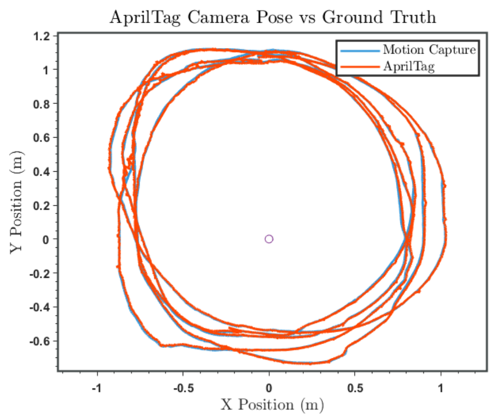
Table 3.1: Above is a table showing the average euclidean distance between the pose estimation of an AprilTag detector and the ground truth estimate of a motion capture system along a trajectory. The error is calculated for a 1, 3, and 5-lap path in ideal lighting conditions, as well as one "dark" trajectory taken in a low-light environment.



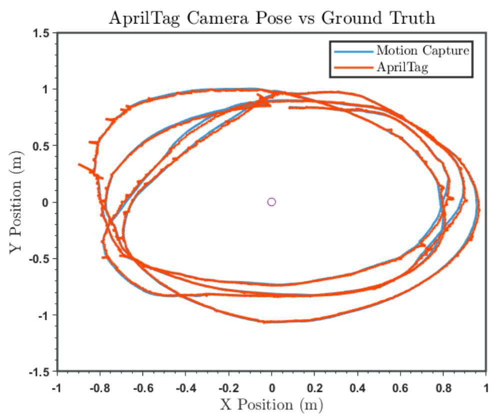
(a) 1 Lap



(b) 3 Laps



(c) 5 Laps



(d) Low-Light

Figure 3-5: The above plots show the X and Y position of the camera as estimated by the AprilTag detector in red. The corresponding ground truth measurements from the motion capture system are plotted in blue. The circular marker at the centroid of the trajectory marks the location of the target object being circumnavigated. The first three plots were generated in ideal lighting conditions, while the last plot was taken in a low-light scenario.

	1 Lap	**3 Laps**	**5 laps**	**Dark**
Mean Error(m)	0.0988	0.0444	0.0453	0.1576

Table 3.2: Above is a table showing the average euclidean distance between the object-pose estimation of a DOPE detector and the ground truth estimate of a motion capture system along a trajectory. The error is calculated for a 1, 3, and 5-lap path in ideal lighting conditions, as well as one "dark" trajectory taken in a low-light environment.

3.2 YCB Object Detection and Pose Estimation

The second set of experiments repeated the process with a YCB object as the target, and with a DOPE detector [31] performing object detection and pose estimation. The specific YCB object used was the *003 cracker box.* The robot once again circumnavigated the target object for one, three, or five laps in ideal lighting conditions, followed by a dark room. An example set of images with labeled pose estimates as seen by the camera in a well-lit environment is shown in Figure 3-6. The images clearly show that the DOPE detector is able to detect the object from every viewing angle and provide an accurate pose estimate. In the dark scenario, however, the DOPE detector displays a noticeable drop in performance. The target object has a much more reflective surface than the AprilTag, particularly the back panel which has significant white space. The result is a major increase in missed or false detections. An example set of images for this experiment are shown in Figure 3-7. Future work will investigate strategies to achieve improved tracking in the presence of these types of outliers.

‘A bird’s-eye view of the camera pose was calculated using the DOPE detections, and was plotted against the ground truth trajectory from the motion capture system for each experiment. The results are shown in Figure 3-8. As with the AprilTag detector, the DOPE detector’s camera pose estimates were compared to the ground truth measurements using euclidean distance as an error metric. The average distance error for the DOPE experiments is recorded in Table 3.2. A graphical comparison of the error values for the AprilTag detector and the DOPE detector is shown in Figure 3-9.

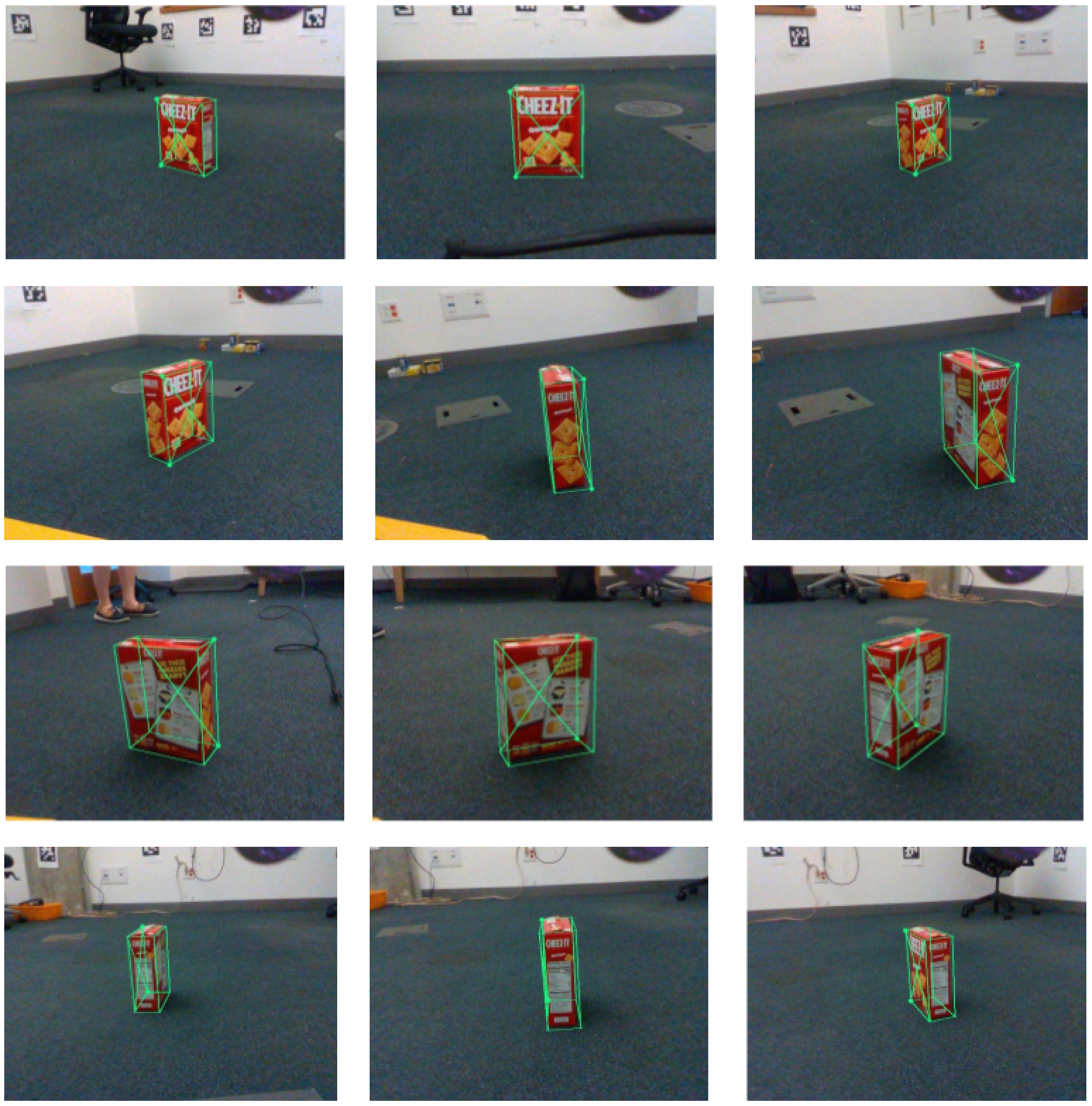


Figure 3-6: Object detection and pose estimation for various times during a typical mission, while circumnavigating an object.

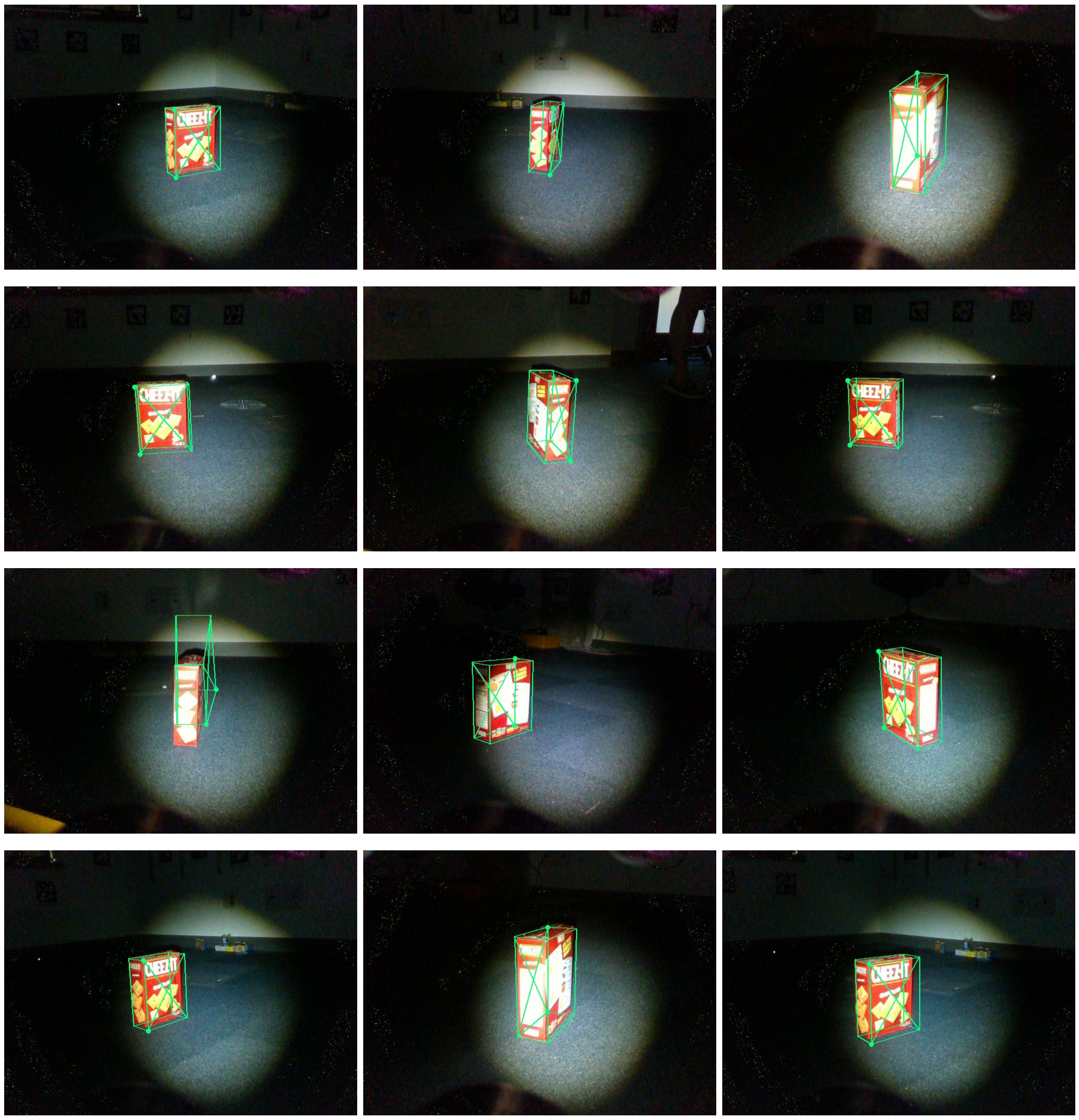
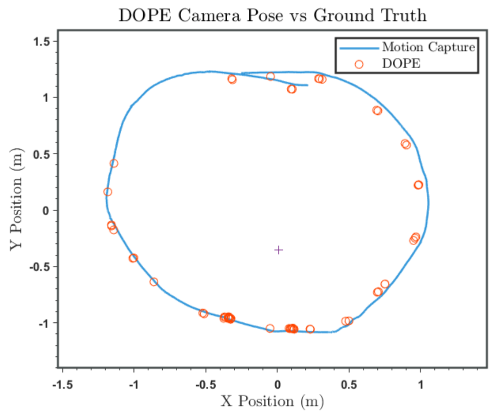
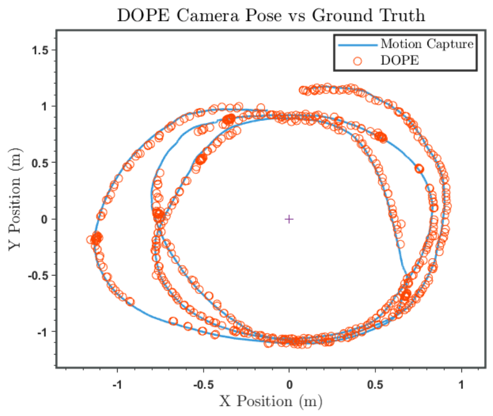


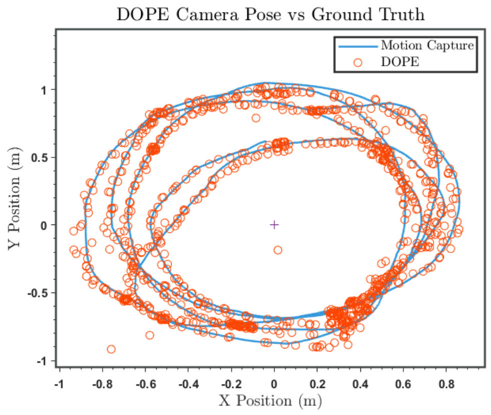
Figure 3-7: Object detection and pose estimation for various times during a typical mission, while circumnavigating an object in a low-light environment.



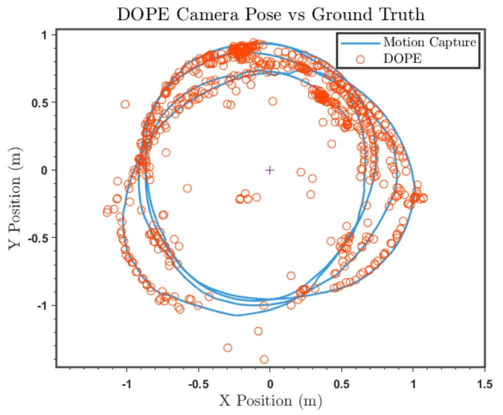
(a) 1 Lap



(b) 3 Laps



(c) 5 Laps



(d) Low-Light

Figure 3-8: The above plots show the X and Y position of the camera as estimated by the DOPE detector as red circles. The corresponding ground truth measurements from the motion capture system are plotted in blue. The cross-shaped marker at the centroid of the trajectory marks the location of the target object being circumnavigated. The first three plots were generated in ideal lighting conditions, while the last plot was taken in a dark scenario. The data from the dark experiment shows significantly more outliers, particularly when the detector is looking at the back of the target, which has more white space and is highly reflective.

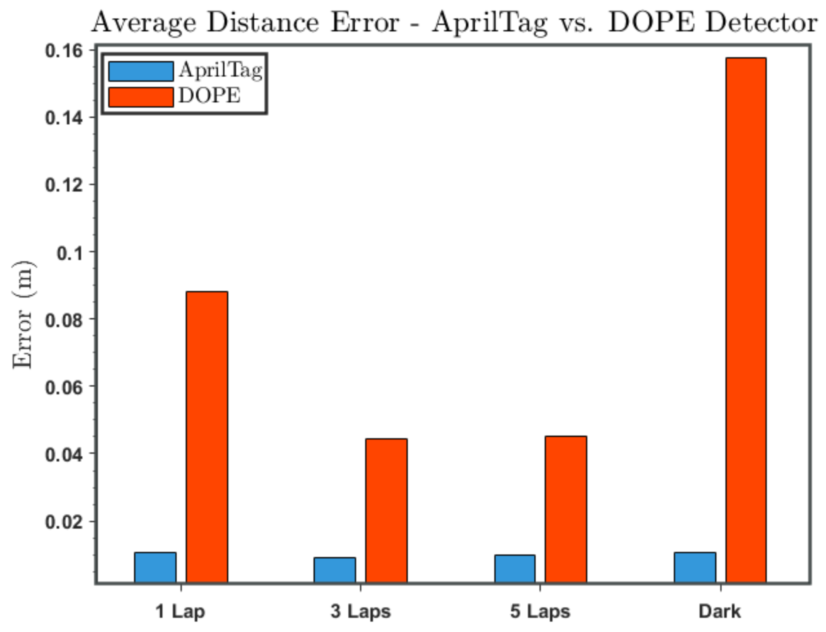


Figure 3-9: The above bar graph compares the average distance error of the AprilTag detector with that of the DOPE detector across all trials. The AprilTag detector shows very consistent performance. The DOPE detector produced more outliers, particularly in the dark scenario which saw a significant drop in performance.

Chapter 4

Conclusion

4.1 Summary

This thesis laid out the design and development of a modular robotic platform for active object-based navigation. This first iteration of the platform was used to collect data for comparing multiple object detection and 6-DOF pose estimation algorithms, specifically an AprilTag detector and NVIDIA's DOPE detector, against ground truth trajectory data from a motion capture system. The platform's pan-tilt module allowed for a camera sensor to be controlled (via visual servoing) separately from the movement of the platform's robotic base. This resulted in much smoother maneuvers than those typically achievable by a tank-drive style robot trying to maintain focus on a circumnavigated target object.

4.2 Future Work

The largest limiting factor with the current iteration of the system is the Jackal's onboard processor. While the default computer is sufficient for controlling the mobile base and performing dead reckoning, it shows significant lag when running AprilTag detection and pose estimation on a live camera feed. The published detections are only sufficient for visual servoing with the pan-tilt module when the mobile base travels at very low speeds. Tracking DOPE objects in this manner is not even an option at the moment. The weights that DOPE uses to detect a known object in a scene are stored in files that are too large to keep on the Jackal. Additionally, DOPE detection is far more computationally expensive than AprilTag detection, so the detector would not publish nearly fast enough to use for visual servoing.

Due to the Jackal's limited onboard memory and processing speed, the data collection was carried out with the camera tethered to a separate laptop running all of the image processing and object pose estimation operations. While tethering the Jackal is effective, it requires much more careful maneuvering when collecting data, to ensure that the cable does not occlude the camera as the robot rotates. Ideally the Jackal would be running untethered, and all computer vision functionality would be run locally. The next iteration of the system will upgrade the Jackal's computer to a more advanced alternative, such as an NVIDIA Jetson, which has the processing capability to quickly run DOPE detection. This is a common update for the Jackal, and the process for upgrading the onboard computer is well-documented by Clearpath Robotics.

Future iterations of the system will also incorporate multi-object tracking. The current system has the pan-tilt module tracking a single class of object at any given time (be it an AprilTag with a known ID, or a single YCB object). While the current control algorithm can be configured to recognize multiple classes of object in a single scene, those objects must be explicitly programmed in order of priority. At the moment, there is no method for handling multiple instances of the same object in a single scene; in the presence of multiple identical objects, the pan-tilt module will sporadically shift its focus between them. For the system to successfully operate in the presence of multiple objects, it needs to incorporate a generalized scheme for prioritizing which object class (and which instance of that class, if applicable) to focus on in a given moment. The primary target could, for example, be the YCB object with the highest detection score, or the closest visible AprilTag. An interesting avenue for future research would be to employ an information-based metric to balance exploration vs. exploitation in object selection, as investigated by Mu *et al.* [20, 21].

Another option would be to incorporate additional elements of active vision. The simple servo-controlled pan-tilt could be replaced with a gimbal with more degrees of freedom (such as a roll axis). Initial plans for this system also involved the addition of a motorized zoom lens. In addition to active steering, another possibility is actively tuning the camera's intrinsic parameters, as explored by Micheloni and Foresti [19]. Having active control over camera exposure/brightness would be particularly useful in environments with difficult lighting conditions.

In addition, the system is expected to be useful to collect new data sets to enabling training of more effective 6DOF object pose estimators, for a broader class of objects — for example, using the self-improving object pose estimation approach developed by Lu *et al.* [18].

Bibliography

[1] Atharva Aalok. Professional plots, 2021. `https://www.mathworks.com/matlabcentral/fileexchange/100766-professional_plots`.

[2] R. K. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):996–1005, August 1988.

[3] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

[4] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set. *IEEE Robotics & Automation Magazine*, 22(3):36–52, 2015.

[5] François Chaumette and Seth Hutchinson. Visual servo control I. Basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[6] Clearpath Robotics. `https://clearpath.com`.

[7] Andrew Davison. *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford, 1998.

[8] Salam Dhou and Yuichi Motai. Scale-invariant optical flow in tracking using a pan-tilt-zoom camera. *Robotica*, 34(9):1923–1947, 2016.

[9] Nicholas R Gans, Guoqiang Hu, and Warren E Dixon. Keeping multiple objects in the field of view of a single PTZ camera. In *2009 American Control Conference*, pages 5259–5264. IEEE, 2009.

[10] GoBilda. `https://gobilda.com`.

[11] Gregory D Hager, Gerhard Grunwald, and Kentaro Toyama. Feature-based visual servoing and its application to telerobotics. In *Intelligent Robots and Systems*, pages 415–430. Elsevier, 1995.

[12] G. Hollinger, B. Englot, F. Hover, U. Mitra, and G. S. Sukhatme. Uncertainty-driven view planning for underwater inspection. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[13] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[14] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.

[15] Noriyasu Inaba, Mitsushige Oda, and Masato Hayashi. Visual servoing of space robot for autonomous satellite capture. *Transactions of the Japan Society for Aeronautical and Space Sciences*, 46(153):173–179, 2003.

[16] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. CosyPose: Consistent multi-view multi-object 6D pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020.

[17] C. D. Cadena Lerma, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Trans. Robotics*, 2016.

[18] Ziqi Lu, Yihao Zhang, Kevin Doherty, Odin A Severinsen, Ethan Yang, and John Leonard. SLAM-supported self-training for 6D object pose estimation. *arXiv preprint arXiv:2203.04424*, 2022.

[19] Christian Micheloni and G.L. Foresti. Active tuning of intrinsic camera parameters. *Automation Science and Engineering, IEEE Transactions on*, 6:577 – 587, 11 2009.

[20] Beipeng Mu, Matthew Giamou, Liam Paull, Ali-akbar Agha-mohammadi, John Leonard, and Jonathan How. Information-based active SLAM via topological feature graphs. In *2016 IEEE 55th Conference on decision and control (Cdc)*, pages 5583–5590. IEEE, 2016.

[21] Beipeng Mu, Shih-Yuan Liu, Liam Paull, John Leonard, and Jonathan P How. SLAM with objects using a nonparametric pose graph. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4602–4609. IEEE, 2016.

[22] David Nakath. *Active Perception for Autonomous Systems: In a Deep Space Navigation Scenario*. PhD thesis, Universität Bremen, 01 2019.

[23] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2011.

[24] Optitrack Motion Capture System. `https://optitrack.com`.

[25] Kyohei Otsu, Ali-Akbar Agha-Mohammadi, and Michael Paton. Where to look? predictive perception with applications to planetary exploration. *IEEE Robotics and Automation Letters*, 3(2):635–642, 2018.

[26] AC Sanderson. Image based visual servo control using relational graph error signal. In *Proc. Int. Conf. Cybernetics and Society, Cambridge, 1980*, 1980.

[27] Himanshu Shah and D. Morrell. A new adaptive zoom algorithm for tracking targets using pan-tilt-zoom cameras. In *Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005.*, pages 59–63, 2005.

[28] Paul M Sharkey, David W Murray, S Vandevelde, Ian D Reid, and Philip F McLauchlan. A modular head/eye platform for real-time reactive vision. *Mechatronics*, 3(4):517–535, 1993.

[29] J. Shi and C. Tomasi. Good features to track. Technical Report TR 93-1399, Cornell University, November 1993.

[30] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[31] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *CoRR*, abs/1809.10790, 2018.

[32] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[33] Yi Yao, Besma Abidi, and Mongi Abidi. 3D Target scale estimation and target feature separation for size preserving tracking in PTZ video. *International Journal of Computer Vision*, 82:244–263, 05 2009.