Intelligent Data Acquisition and Processing for Unmanned Aerial Vehicles

by

Justin Griggs B.S.E.E

A Thesis
In
Electrical Engineering
Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment for
The Requirements for
The Degree of
Master of Science

Approved

Richard Gale
Chair of Committee

Mohan Sridharan

Peggy Gordon Miller
Dean of the Graduate School

August, 2012

Acknowledgments

First and foremost I'd like to thank my parents, Robert and Rosie Griggs, for their endless support. I would not be where I am today without them. They have provided encouragement and financial support through this process.

My sincere thanks go to Dr. Richard Gale for providing guidance and support during my undergraduate and graduate careers. His support for STEM robotics has brought me into leadership roles, challenged me intellectually, and created a support structure that helped carry me through my curriculum.

I'd like to thank FIRST Robotics for getting me interested in engineering during high school. Being involved in FIRST through college has challenged my engineering skills that supplement my classroom knowledge and make me a better person.

Table of Contents

Abstract

This paper describes the design and implementation of a quadrotor that autonomously analyses moving objects and generates real-time waypoints for persistent reconnaissance of objects of interest. The system is equipped with an autopilot to control the craft in flight and an onboard image processor. The processor analyses images to create real-time estimations of target direction and velocity. An algorithm to create a flight plan from the imaging data acquired is described. A simulation of an algorithm to generate optimal trajectories through a sequence of positions and yaw angles is developed. Experimental results of the system are shown.

List of Figures

List of Abbreviations

APM – ArduPilot Mega

CAD - Computer Aided Design

ESC – Electronic Speed Controller

FOV – Field of View

GCS – Ground Control Station

IMU – Inertial Measurement Unit

ISR – Intelligence, Surveillance, and Reconnaissance

MAV – Micro Air Vehicle

UAS – Unmanned Aircraft System

UAV – Unmanned Aerial Vehicle

Introduction

Unmanned Aerial Vehicles (UAVs) are gaining in popularity and capability. Their applications include search and rescue, surveillance, and mobile sensor networks. UAVs currently in use for military surveillance applications include the MQ-9 Reaper, RQ-7 Shadow, and the RQ-11 Raven. These systems consist of an aerial vehicle with a sensor package and data link to communicate sensor data to an associated ground station. The purpose of this is to provide the Warfighter with near real-time situational awareness of the battle space. As of 2010, the US Army possesses more than 4,000 unmanned aerial systems (UAS) with still more planned (Dempsey 2010).

Current UAS technologies operate with a low degree of autonomy. This may include automatic takeoff and landing, or following a flight plan. While the ground control station (GCS) is used to receive the data collected from the system, it is also used to control the UAS. This project seeks to add autonomy to the control aspects of the UAS and reduce the workload on the Warfighter.

The ability to automatically and intelligently process UAS sensor data is a key capability when operating in an urban environment or indoors. With recent advances in embedded system technology, sensor data can now be processed onboard to determine the needed movements of the vehicle. This increases the degree of autonomy the system can operate at and allows for the user to receive the data while not needing to control the UAS.

Motivation

I became interested in starting this project over the course of a year or so while working in the defense industry as an electrical engineer. I have seen how the aerospace industry is evolving in response to the presence of advancing UAS technology. The need for higher degrees of autonomy is a key aspect that will make the system easier to operate and allow users to focus on the primary goal of the system.

Additional inspiration comes from University of Pennsylvania's GRASP Lab. Dr. Vijay Kumar and his associates have put out some fantastic videos highlighting their work on agile quadrotors (Aggressive Quadrotor Part II 2010). This led me to the concept of building a quadrotor of my own and adding additional capabilities to it.

Background

History

The first successful flight of a quadrotor was in 1907. The first experiments of the *Gyroplane No. 1* were carried out in France where the machine is reported to have lifted into flight (Leishman n.d.). The craft lifted as high as 5 ft for a brief amount of time. The pilot had little means of control besides a throttle to change the rotor speed. However, the inability to control four motors simultaneously with sufficient bandwidth rendered the platform ineffective (Hoffmann, et al. 2007).

Recent Developments

Recently with the advent of modern microcontrollers and advances in sensor technology has spawned a surging interest in quadrotor design and capability. Today, the multicopter market is expanding with more and more vendors and various platforms. Uses and applications for these platforms are being researched and expanded upon (Drone journalism Lab 2011) (Ungerleider 2012).

Multicopters typically give high agility at the expense of range. The multcopter must generate all of its lift with propellers and it unaided by airfoils such as is a fixed wing aircraft.

The civilian population contains creativity like none other. Multicopter video platforms are being used for finding 10,000 year trails in Kenya (Autodesk Octo-Copters Scouring for Kenyan Trails 2012). A San Francisco based company aims to deliver food by using multicopters (Tacocopter 2012). New uses for these platforms are appearing frequently in places that are unexpected.

Modeling Quadrotor Dynamics

The dynamic model of a quad rotor is presented to illustrate the needed control algorithms. Z, X, and Y are used to define the roll, pitch, and yaw angles on a local coordinate system.

Frame Configuration

Typically, the motors are placed along the X and Y axes, known as the "+" configuration. In this project, the motors are rotated 45 degrees, known as the "X" configuration (Attaching the props to the motors 2011). Illustrations are shown below. The "X" configuration is used to allow the camera's Field of View (FOV) to be unobstructed by the motor props. Modeling the control of the quadrotor is complicated by laying the motors off of the axes.

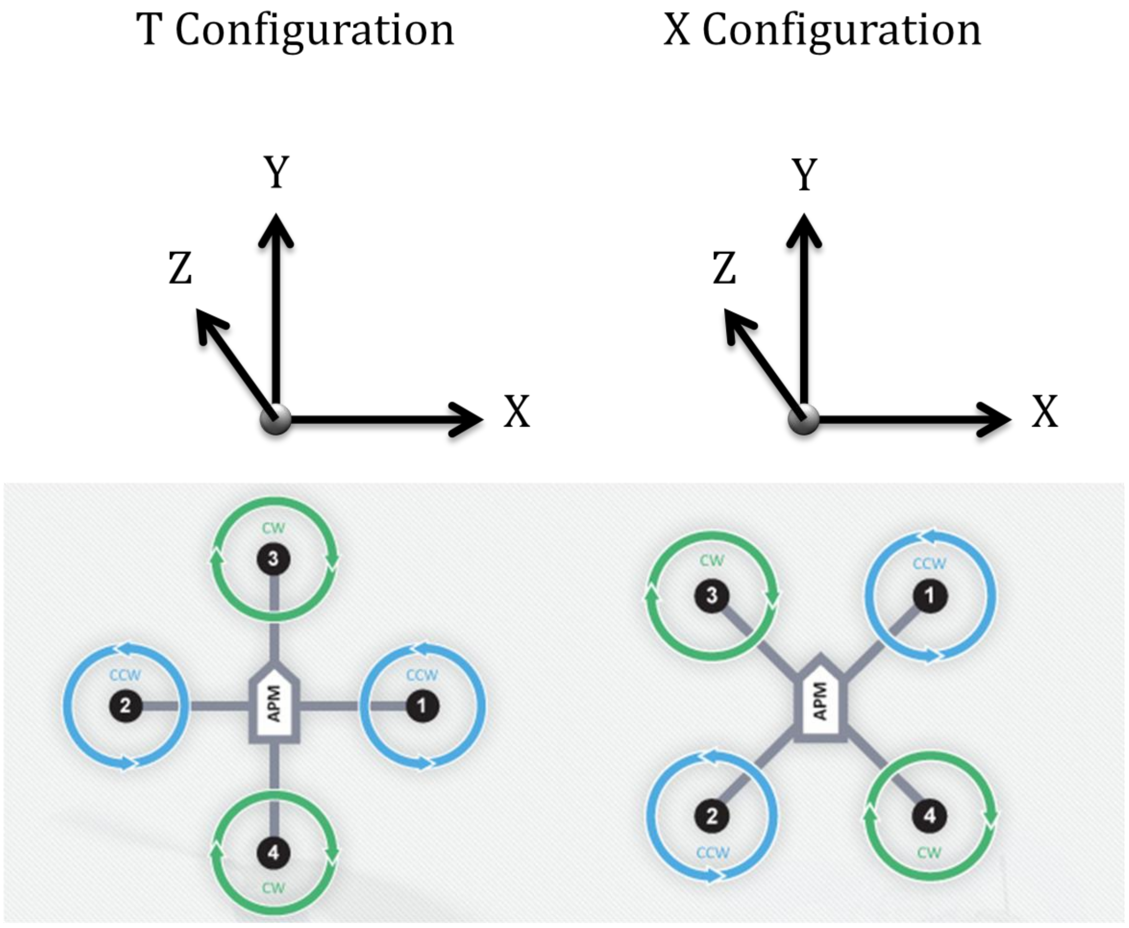


Figure 1: Quadrotor Frame Configurations

Dynamics

Two pairs of propellers (1, 2) and (3, 4) spin in opposite directions. The four thrust combinations F1, F2, F3 and F4 can be used to create stable and controllable flight.

System Goals

These are the high level considerations for the design of the complete system. The concepts and requirements described here are the goals of the project.

System Requirements

To design the system, requirements are needed. The system has the following requirements:

1. Capability of stable flight.
2. Flight time of at least 20 minutes.
3. Ability to identify an object of interest while in flight
4. Ability to follow the identified target while in flight

Action Language Model

When designing an autonomous agent, it can be viewed as a representation of knowledge and reasoning (Gelfond and Kahl 2012). The goal is an agent capable of reasoning and acting in a changing environment. Several principle ideas are implemented to allow the agent, in this case a quadrotor, to “act intelligently”. A model of the dynamic domain using *Action Language* is presented. Action language is used to specify state transitions. Sets of beliefs that are held by the agent are guided by the following informal principles:

1. Do not believe in contradictions.
2. Adhere to “the rationality principle” which says: “Believe nothing you are not forced to believe.”

Modeling Dynamic Domains

The properties of the domain are described using *statics* and *fluents*. Fluents are the properties of the system which can be changed by actions (i.e. the location of the quadrotor). Statics cannot be changed by an action (i.e. the number of motors).

An action language model is composed of the following statements:

1. Causal Laws:

   a **causes** *l* **if** *p*

2. State Constraints:

   l **if** *p*

3. Executability Conditions:

   impossible $a_0, \ldots, a_k$ **if** *p*

Where *a* is an action, *l* is an arbitrary domain literal, *p* is a set of domain literals, and $k \geq 0$. A set *S* of domain literals is called complete if for any domain property *p* either *p* or $-p$ is in *S* (Gelfond and Kahl 2012).

Inertia Axiom

At any given period in time, the agent must have a set of beliefs about its environment and its state. As a dynamic domain evolves and changes, so to should the agents beliefs. The Inertia Axiom creates rules stating that “without evidence to the contrary, things stay the way they are.” Any fluent that is not observed to have changed is assumed to have not changed.

Awareness Axiom

At all times, the agent must be able to take into consideration all fluents of the system (Gelfond and Kahl 2012). At any given period in time, including history, the agent must be able to know whether a fluent was true or false. If the agent is unaware of the state of a fluent, then both cases are thereby possible, which is undesired.

Objects of the domain

Stating the objects of the domain simply acknowledges their existence. It makes no observation on their initial state or their properties. The objects of this domain are:

quadrotor

target

This creates the assumption that there is only one quadrotor in the domain and only one target to be tracked.

Static relationships

The static relationships must be known to describe the environment to the agent and to describe relationships that do not change by actions. This allows us to create constraints for the agent to direct reasoning. The statics of the domain are:

World Size

Camera FOV

Camera Length

Fluents

Fluents is this domain change with actions and time. The fluents of this domain are:

quad_location – The location of the quadrotor.

target_location - The location of the target.

detected – whether a target has been detected.

searching – whether the quadrotor is searching for a target.

Actions

Actions are, as they sound, used to change the environment. These actions are used to change the state of the system and modify the value of fluents. The objects of this domain are:

move - moves the quadrotor to a new location

detect - detects the presence of a target.

execute_strategy – causes the desired tracking strategy to be executed

execute_search– causes the desired search to be executed

The following actions are used by the *detect* action to find information about the target.

find_location

find_distance

find_yaw

find_velocity

System Description

The model has the following system description. This describes relations between knowledge and dynamic properties.

move **causes** *location*

searching **causes** *execute_search*

detected **causes** *execute_strategy*

searching **if** *–detected*

detected **if** *–searching*

impossible *execute_search* **if** *–searching*

impossible *execute_strategy* **if** *–detected*

impossible *find_location, find_distance, find_yaw, find_velocity* **if** *–detected*

Model Limitations

There are several noted limitations in this model. The previously mentioned implied assumptions of one target object limit the ability to track multiple targets. In addition, AL models are typically unable to handle uncertainty. In order to create valid state transitions, all values of all fluents must be known (Gelfond and Kahl 2012).

AL models when used in application of intelligent agents are typically used with a declarative programming language, such as Prolog. When this model is simulated using MATLAB, translation to a procedural programming language means the model must be adapted to fit the environment.

The presented model is used to create a framework that facilitates the development of a simulation. This model can be simulated or implemented in various languages to suit multiple applications.

Search and Track Modeling

The conceptual functions of the agent are presented here. The major functions of the agent are to search for and track a target. State transitions and user defined variables are described and developed.

Search

When the quadrotor is not tracking a target, it must be searching for one. Many methods of physical search are available. The methods used are able to take advantage of the agility of the platform. Two methods of searching are presented, known as *Uninformed* and *Informed*.

Uninformed

When no previous information about a target is available, then the probability distribution of the target location is equally distributed. This is the initial condition when nothing has been found yet. The only user defined variable needed is the Tracking Strategy, *S*, which is to be executed once a target is found. The expressed purpose of the strategy is to cover area as quickly as possible, in order to find a target quickly, while minimizing overlap, which depends upon camera characteristics. The values used are the following:

Detected – True if a target is detected, false otherwise.
S – The tracking strategy to be executed

The following actions are used:

Detect – detects the presence of a target.
Execute_strategy(S) – when a target is detected, a strategy is called

The state transition diagram below shows the search starting when first initialized. The first state is the initial undetected state which then begins the search. When a "move" command is called, the next command called is "detect", which evaluates the data from the camera at the current frame. If a target is detected from the "detect" command, the value, *Detected* becomes true. When *Detected* becomes true, *Execute_strategy* calls the desired tracking strategy. The quadrotor leaves the search state and begins tracking its acquired target.

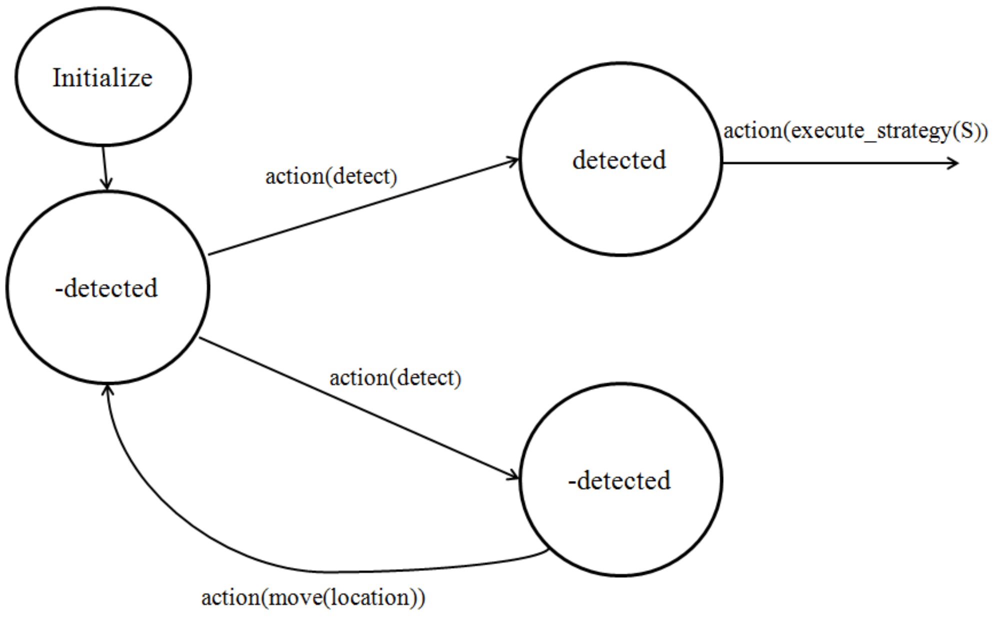


Figure 2: Uninformed Search State Transition

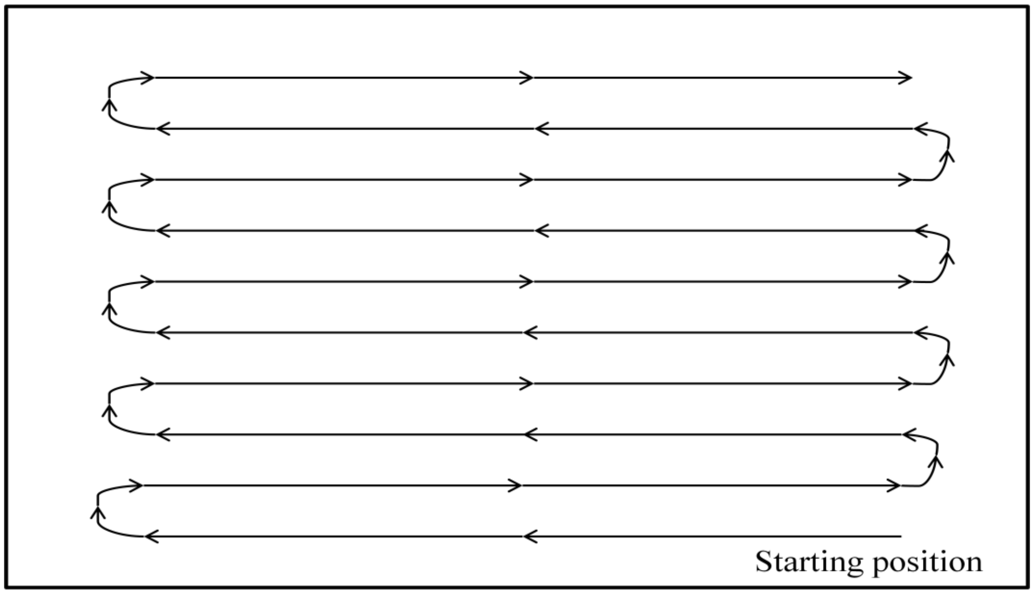


Figure 3: Uninformed Search Path

Informed

When a target is identified then later lost, the probability of its location is dependent upon its previous position. The needed variables are the last known location of the target.

X_L – the last recorded location of the target in X

Y_L – the last recorded location of the target in Y

The transitions are similar to that of the uninformed search, except the initial condition used is X_L and Y_L. Location is described using the following:

(1)

(2)

Where

(3)

— **(4)**

Where is the radius propagation factor, is the angular propagation factor, and is the current frame in time. By varying and , the rate of radial expansion can be controlled.

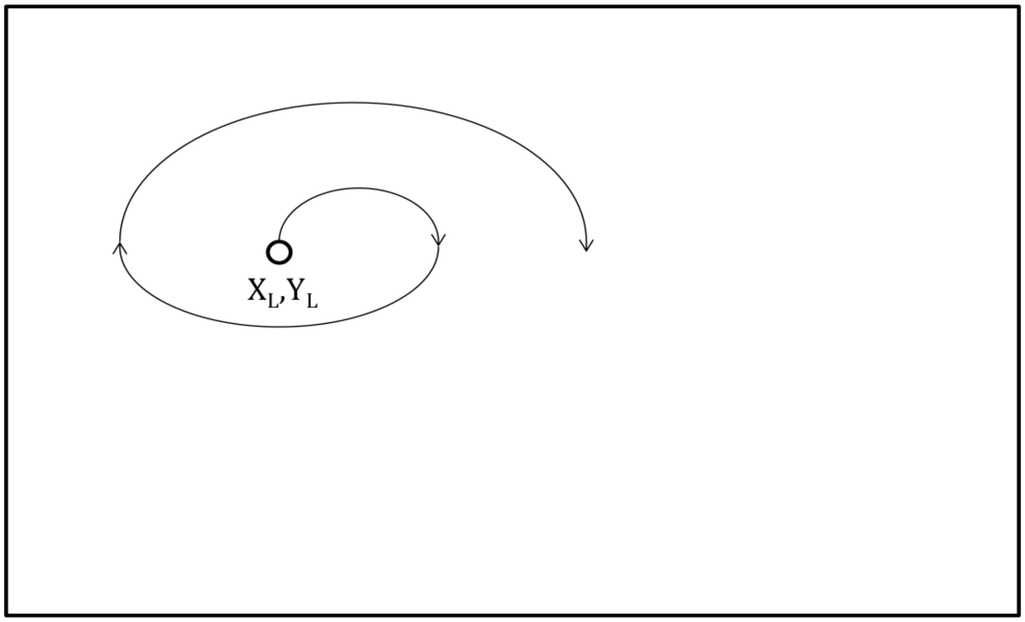


Figure 4: Informed Search Path

Tracking

There are many possible ways to track a target. Two are presented here to take advantage of the quadrotor's inherit agility with respect to forward velocity and strafing.

When a fixed wing aircraft is used for surveilling a moving target, the velocity of the aircraft is constrained to its operating envelope. Typically, the aircraft cannot slow down to the velocity of a target such as a vehicle. The aircraft is then forced to circle around the target, following a forward trajectory. This necessitates adding complexity such as a camera gimbal to rotate the camera view (L-3 Communications - WESCAM 2012). Several variables are needed to describe the model. The quadrotor related variables are the following:

- quadrotor location

V_t - quadrotor velocity

The target related variables are the following:

- target location

D_t - distance from the target to the quadrotor

V_t - target velocity

Station Keeping

The purpose of the Station Keeping strategy is to maintain a desired angle relative to the target's movement. By matching the velocity of the quadrotor and the target when in a linear trajectory, a consistent view of the target is accomplished.

The needed user defined variables are the following:

Phi_d - The desired yaw angle relative to target yaw phi_t

Where

(5)

Circle

The circle strategy is used to give a rotating view of the target. A rotating view gives a more complete model of the target while forcing the quad rotor to maneuver more aggressively.

The needed user defined variables are the following:

$Rate_{phi}$ = The desired rate of rotation.

The simulation of the circle strategy is described below.

Simulation

A Matlab simulation was written to develop, test, and illustrate the autonomous functions. A simulation environment allows for control over process noise and allows for a controlled model of the vision system.

World

The model of the world is viewed as a 100 meter by 100 meter space. The quadrotor and the target are modeled as single points in space due to the large size of the world. Time is discretely modeled with 30 frames per second.

Target Trajectory

The Target trajectory is hard coded into the simulation to give a deterministic output. A variety of trajectories and velocities were simulation with a maximum velocity of 10m/s. An example trajectory path is shown below.

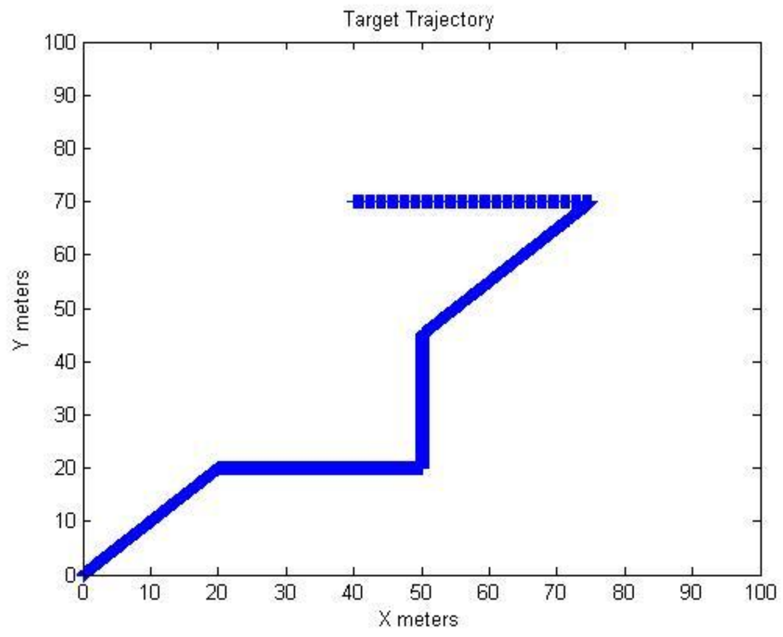


Figure 5: Example Target Trajectory

PID-Vision Based Control

The controller for the quadrotor is modeled with PID control. Separate controllers for X, Y, and yaw movements provide response to the input error and setpoint. The setpoints and error change as the states are transitioned.

The model of the vision system is used to create a desired setpoint in X and Y. From the quadrotor's position, using the FOV of the camera and a distance, a triangular polygon is created to represent the space the camera can see.

With the camera's sight modeled, the set point can be created. The set point is set at 55% the length that the camera sees and in the center of the FOV. By creating this setpoint, the object's location is used to create an error in X and Y to create the desired response.

And

(6)

(7)

(8)

Where C is the distance the camera's effective range.

The PID controller used is a discrete version of the classical analog PID controller in matrix form.

(9)

Where is the error function at a time . Errors for all aspects of position are calculated simultaneously.

After tuning the PID constants, the impulse response is tested. The object is placed such that it is on the edge of the FOV of the camera in X and Y.

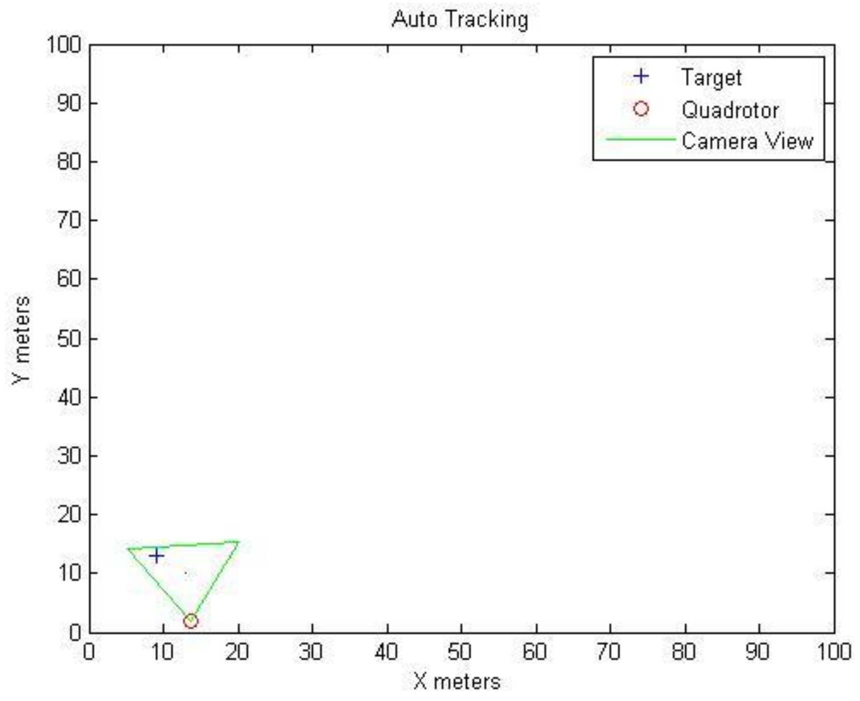


Figure 6: Camera PID Impulse Test Start

The quadrotor corrects for the error in X,Y, and yaw (*phi*) all at once. The end of the test after 100 frames is shown below.

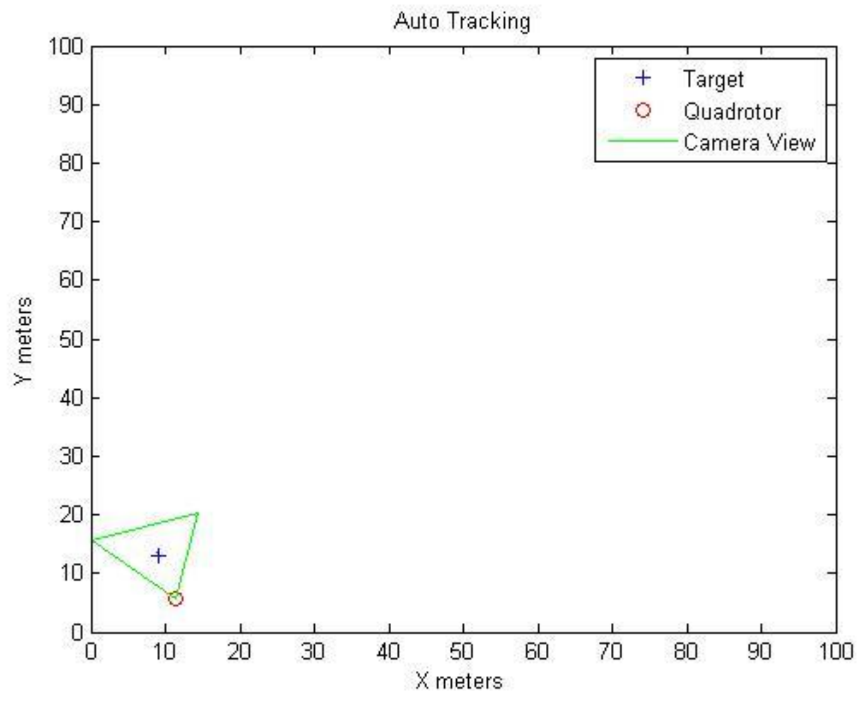


Figure 7: Camera PID Impulse Test End

The impulse responses for X,Y,and phi are shown below where 100 frames are rendered.

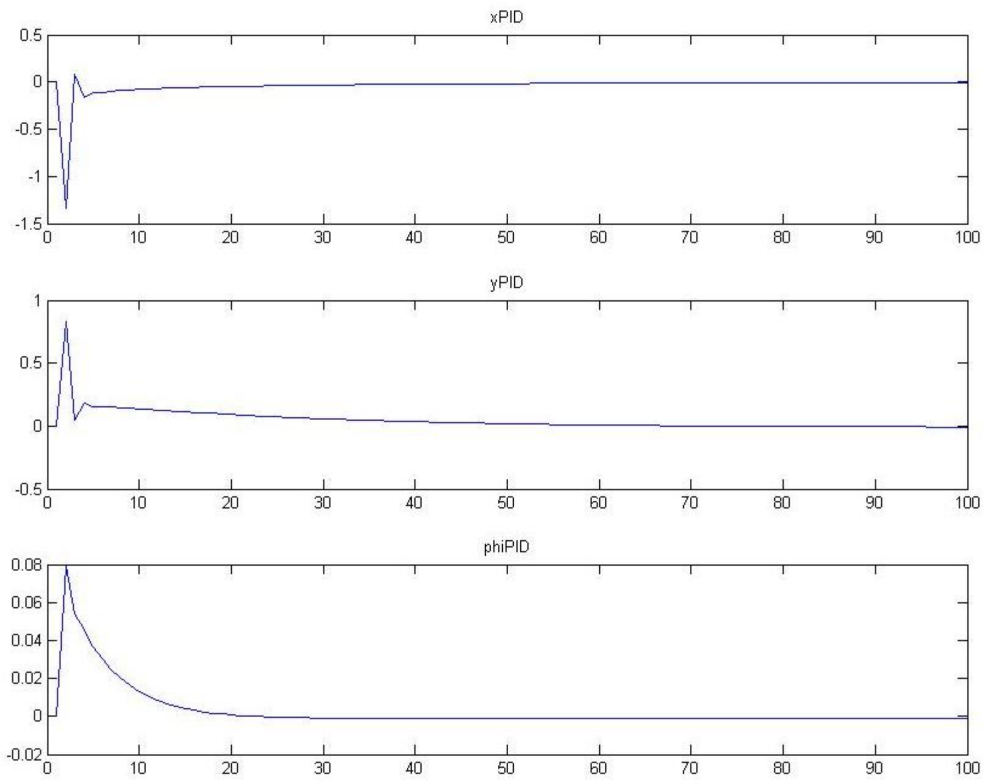


Figure 8: Camera PID Impulse Test Response

The response data shows the advantage of rotating over translation to minimize error. This control scheme forms the basis for all needed movements. By varying the vision-based error, search and track are possible.

Uninformed Searching

Uninformed Searching is run by creating a state machine to guide the execution of a series of paths. Four paths are created to describe the necessary movements.

These paths are called in succession until all world surface area is covered. When the Y maximum is reached, the search is restarted with a Y minimum. The following image series illustrate the pattern taken. The yaw during these movements is set to coordinate with the quadrotor's direction, so it always points forward.

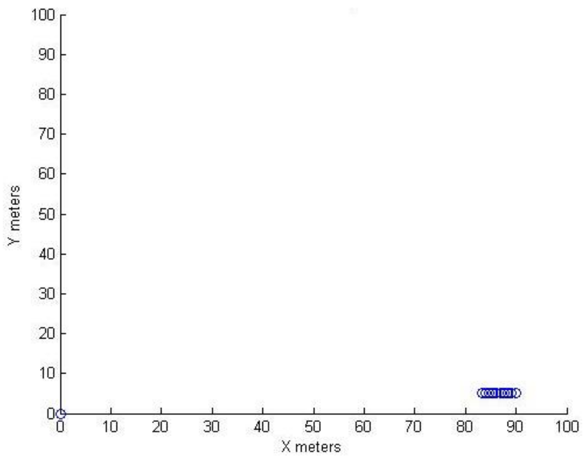


Figure 9: Uninformed Search at frame 10

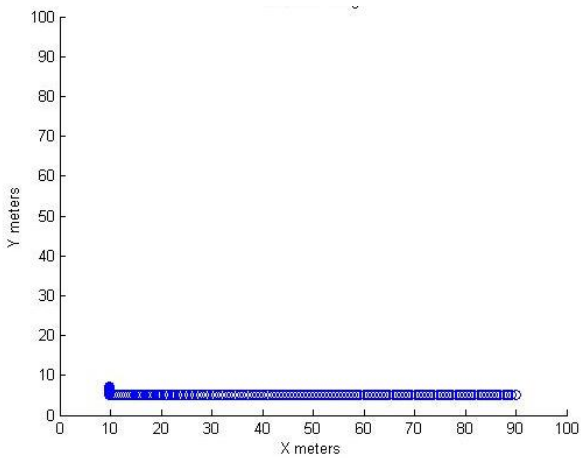


Figure 10: Uninformed Search at frame 140

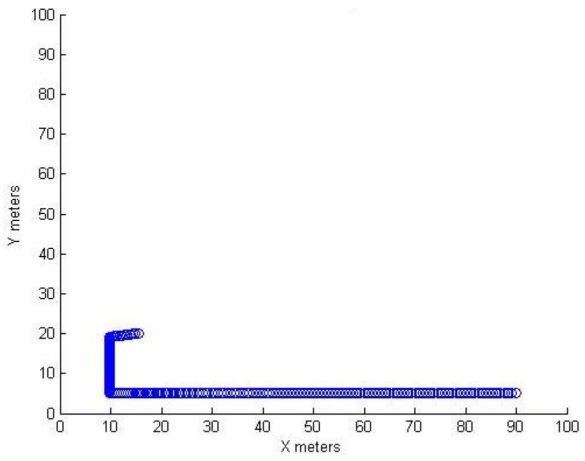


Figure 11: Uninformed Search at frame 250

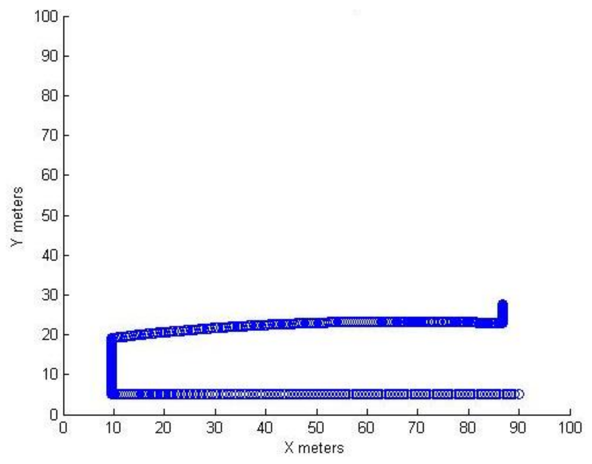


Figure 12: Uninformed Search at frame 440

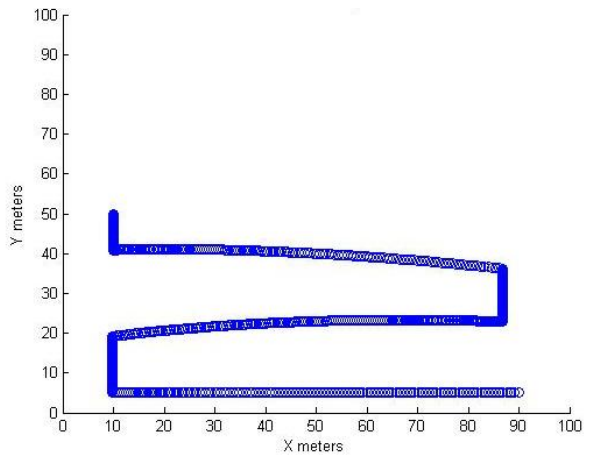


Figure 13: Uninformed Search at frame 700

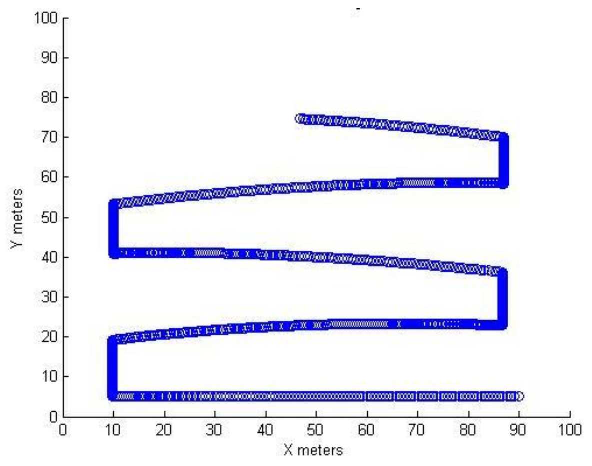


Figure 14: Uninformed Search at frame 1000

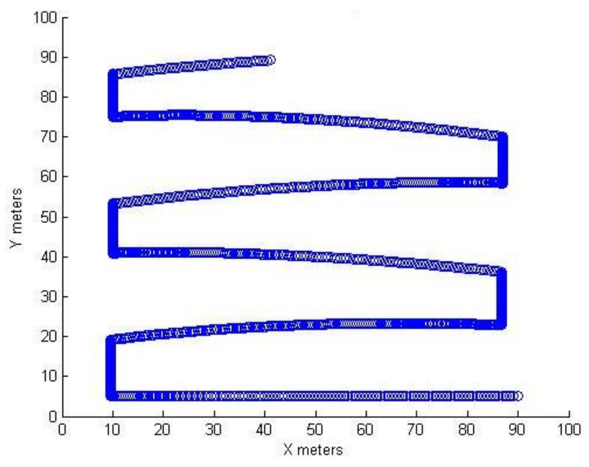


Figure 15: Uninformed Search at frame 1200

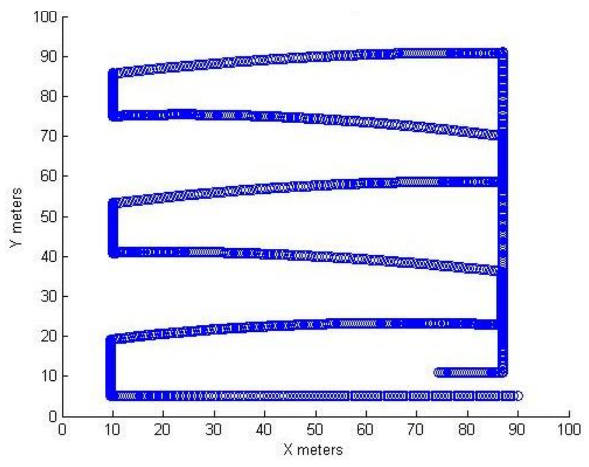


Figure 16: Uninformed Search at frame 1450

Informed Searching

Informed searching is used when a target has been detected but then is lost. The test shown below places the target in the detection range of the quadrotor. The target is then moved instantaneously to another location, causing the target location to be lost. The quadrotor moves into the *informed searching* state. The error for the PID controller is constructed as follows:

— **(10)**

— **(11)**

(12)

Where *i* is the current frame. Rotation is simple set to a constant rate. In this simulation, = 22 and = 0.33

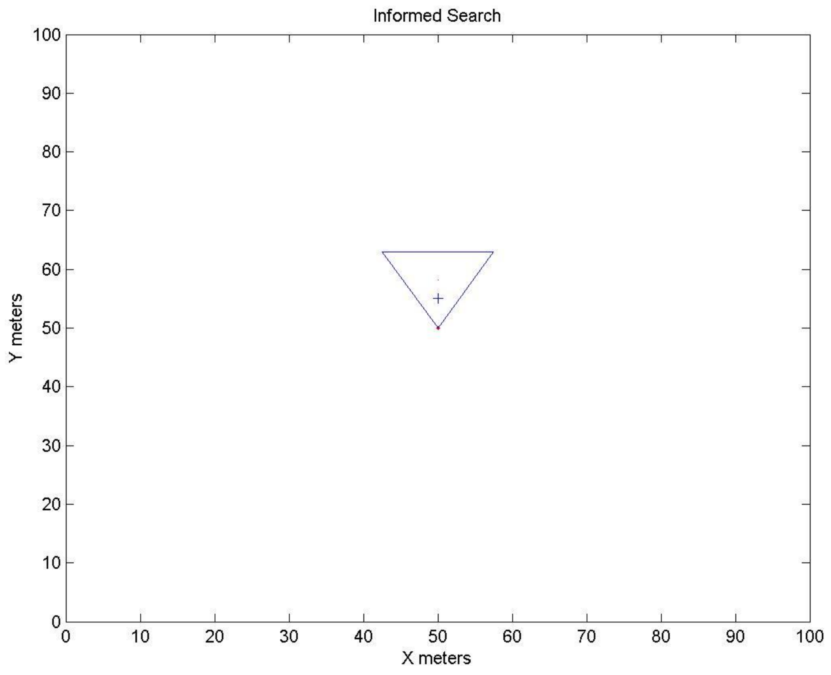


Figure 17: Informed Search at Frame 5

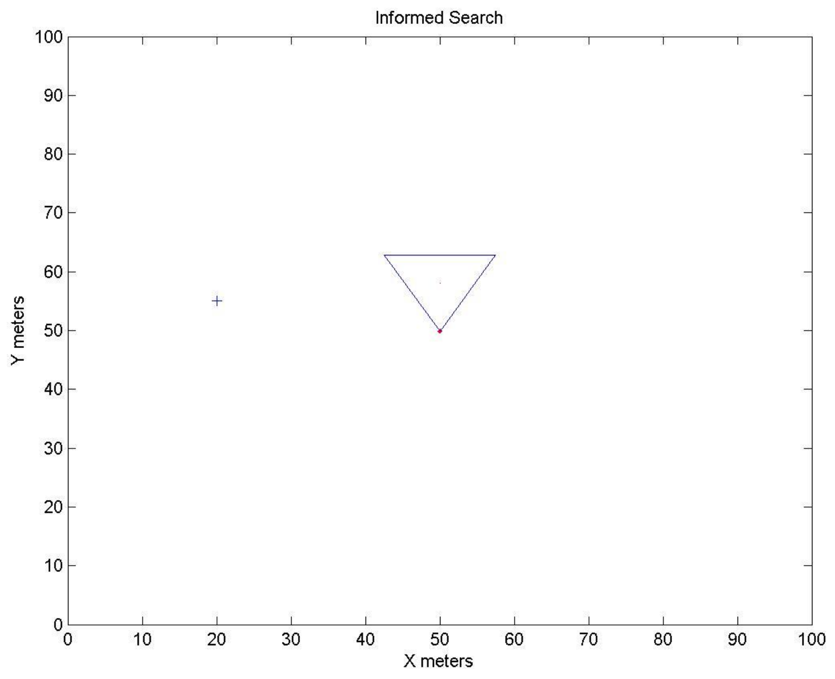


Figure 18: Informed Search at Frame 50

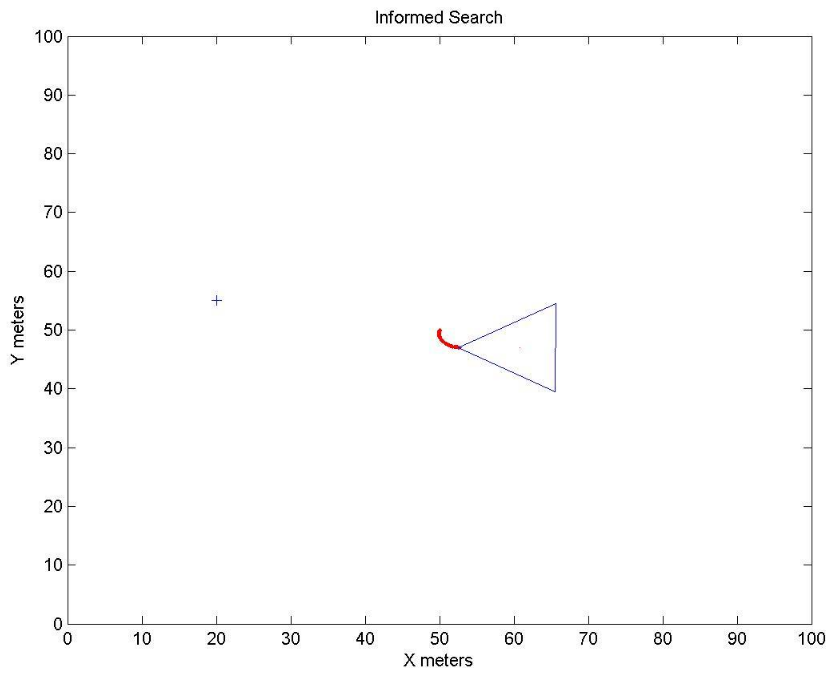


Figure 19: Informed Search at Frame 110

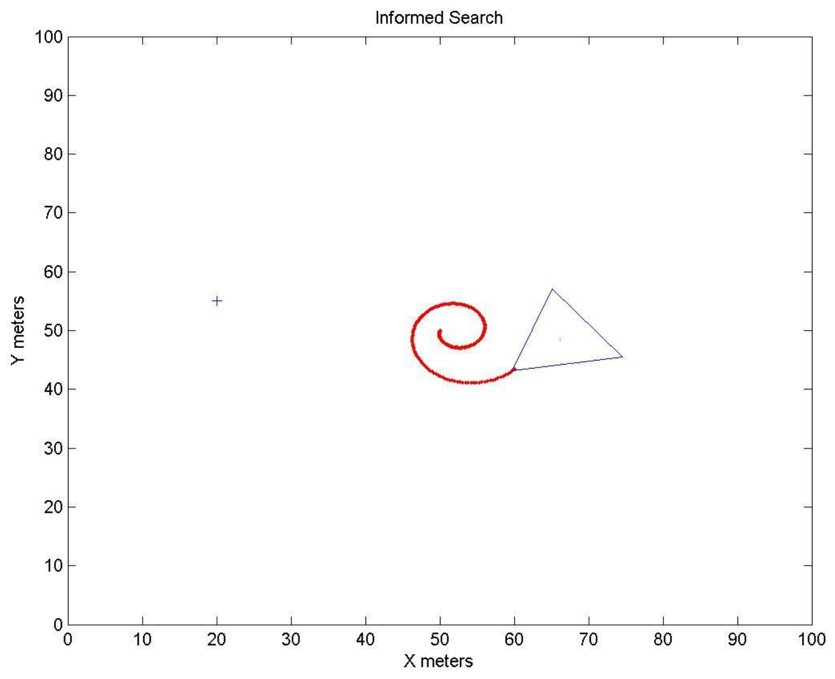


Figure 20: Informed Search at Frame 265

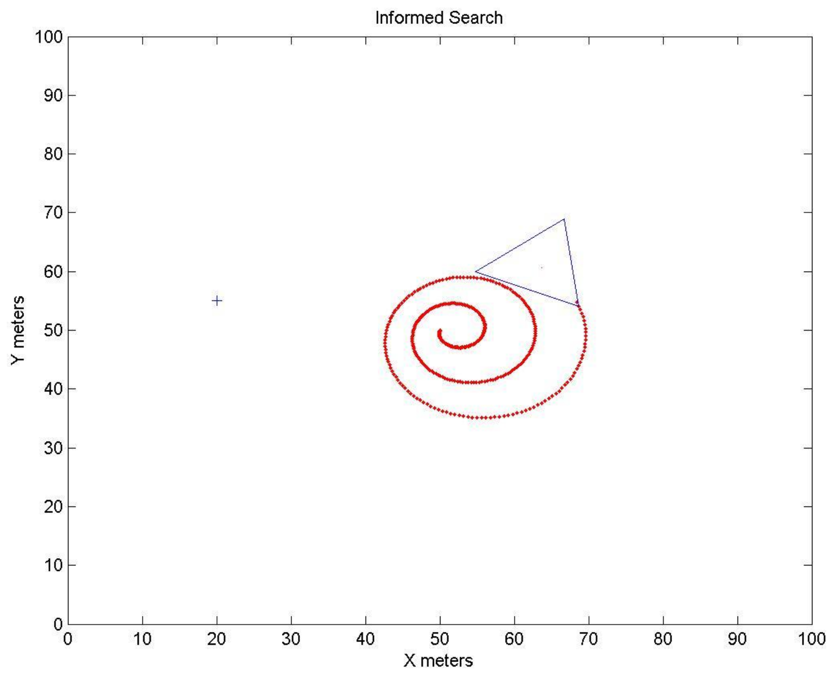


Figure 21: Informed Search at Frame 430

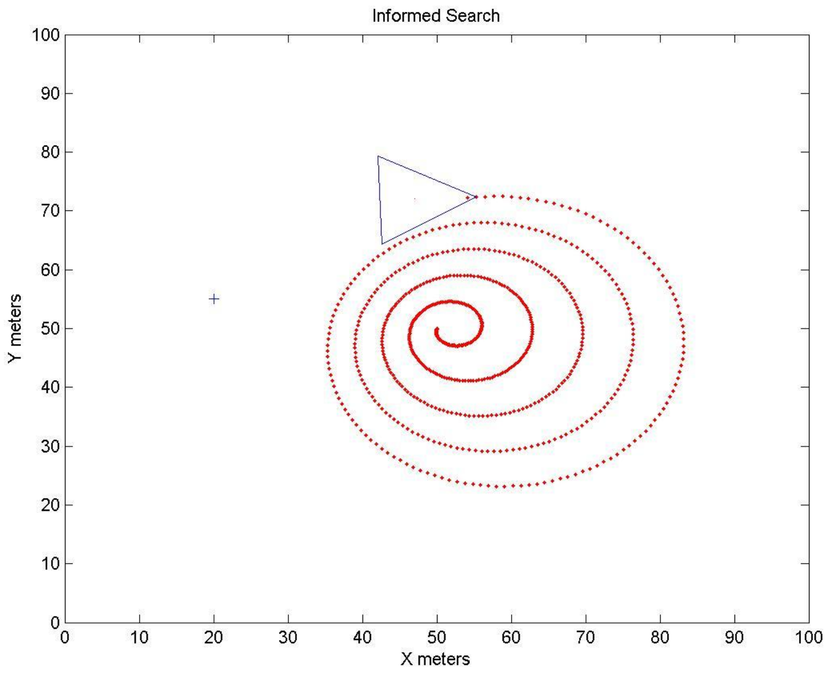


Figure 22: Informed Search at Frame 735

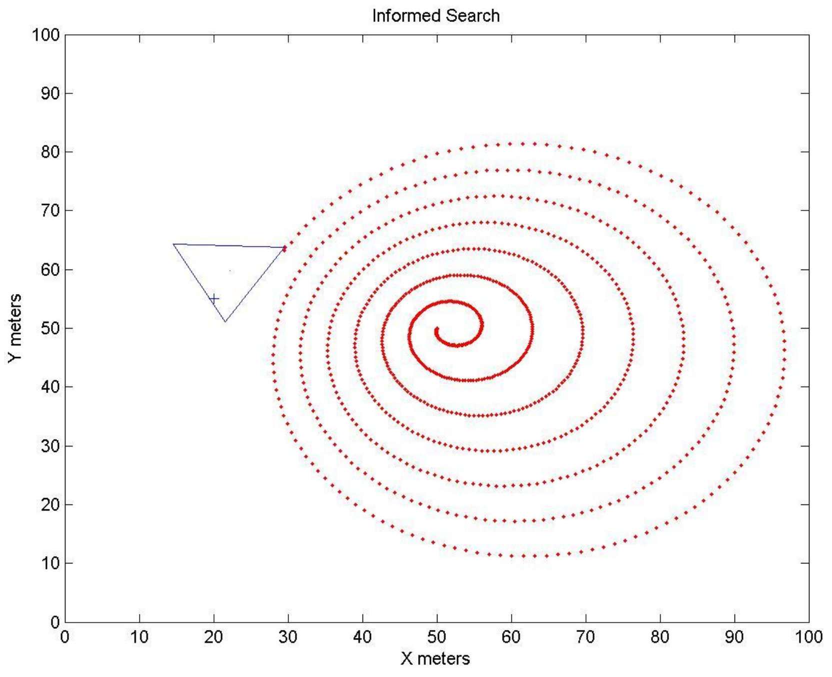


Figure 23: Informed Search at Frame 1035

When the target is detected again, the quadrotor moves back into a tracking state; in this case *auto* is used. The associated PID responses are shown below.

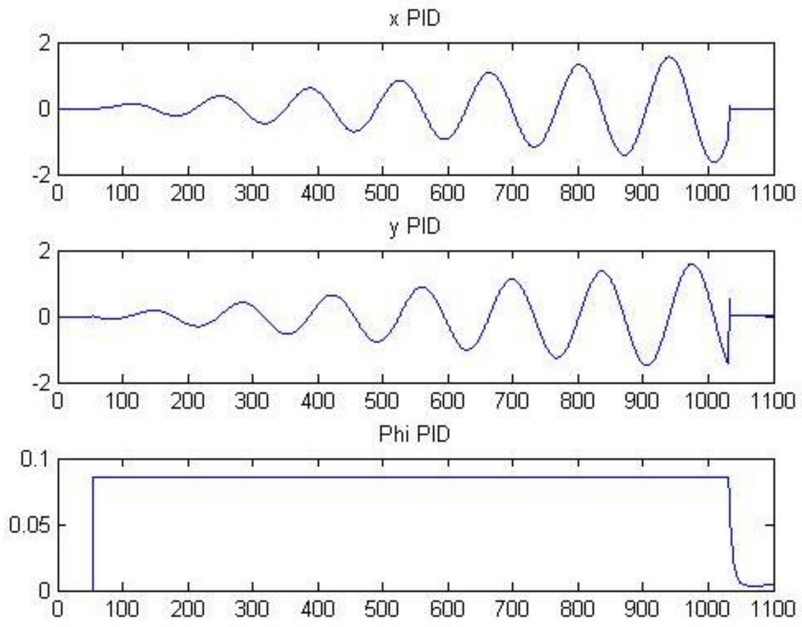


Figure 24: Informed Search Error Response

Auto Tracking

The Auto strategy is used when no user specified strategy is given. If there are no constraints on the tracking strategy given, then full control over the quadrotors movements is given to the vision based PID controllers. X, Y, and yaw errors are corrected for simultaneously.

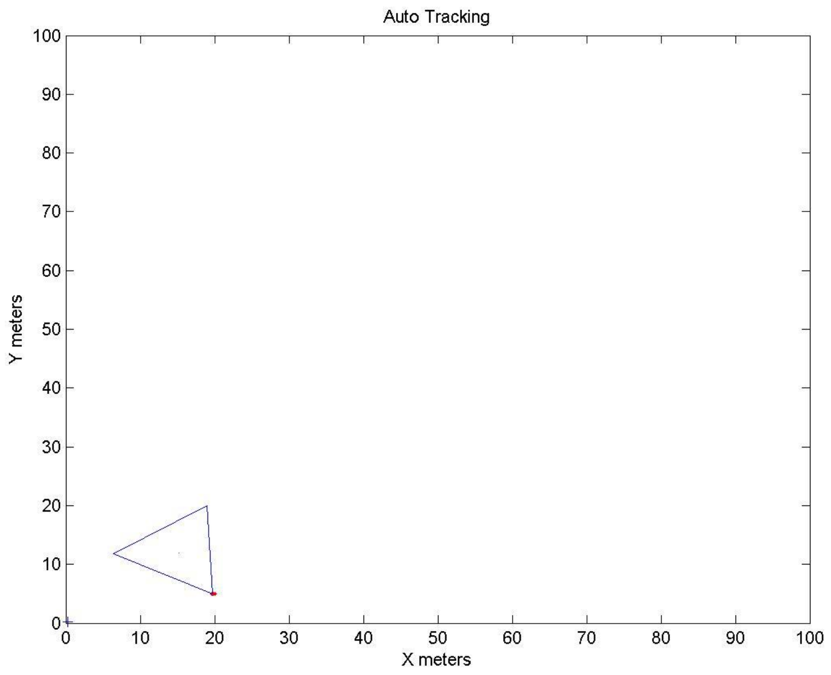


Figure 25: Auto Tracking at Frame 2

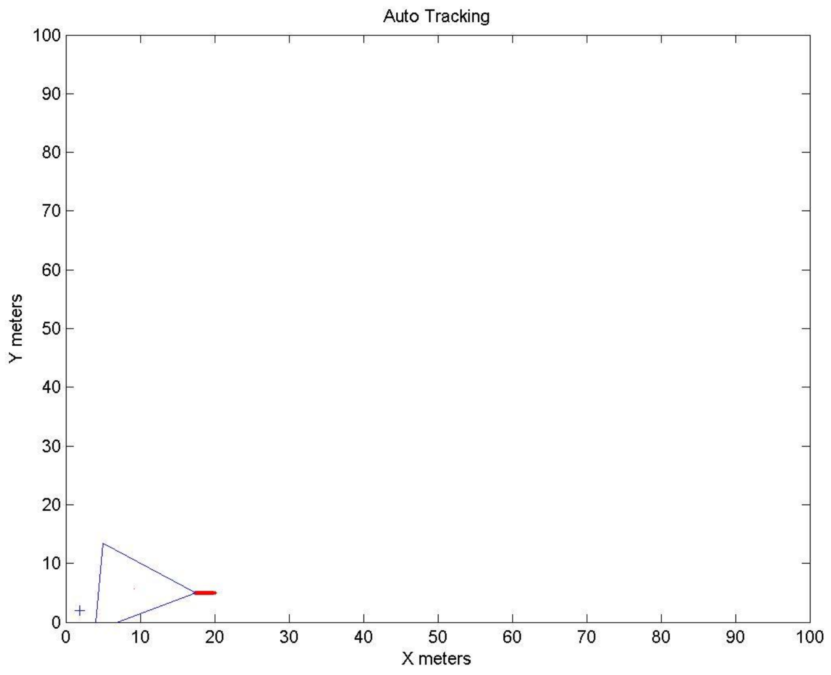


Figure 26: Auto Tracking at Frame 30

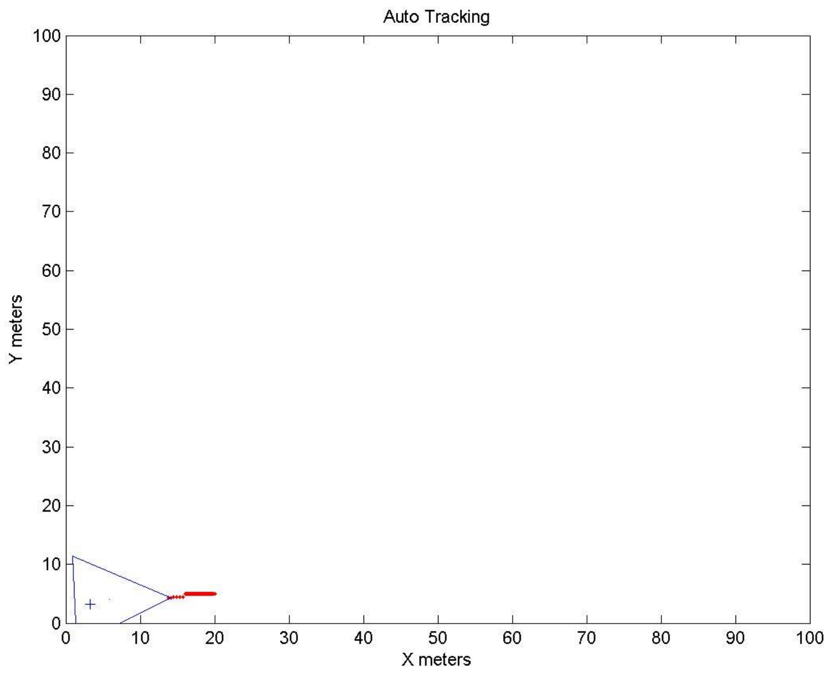


Figure 27: Auto Tracking at Frame 50

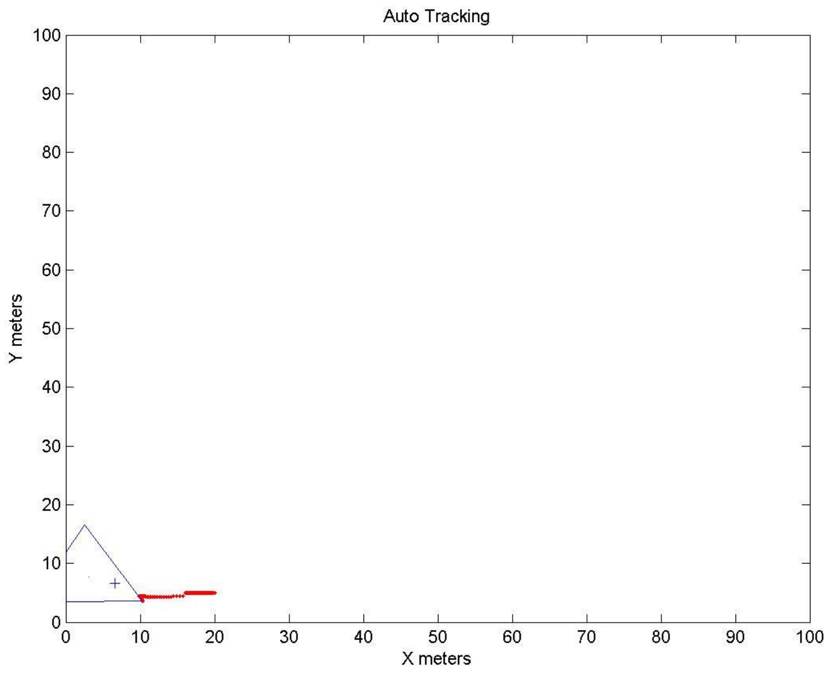


Figure 28: Auto Tracking at Frame 100

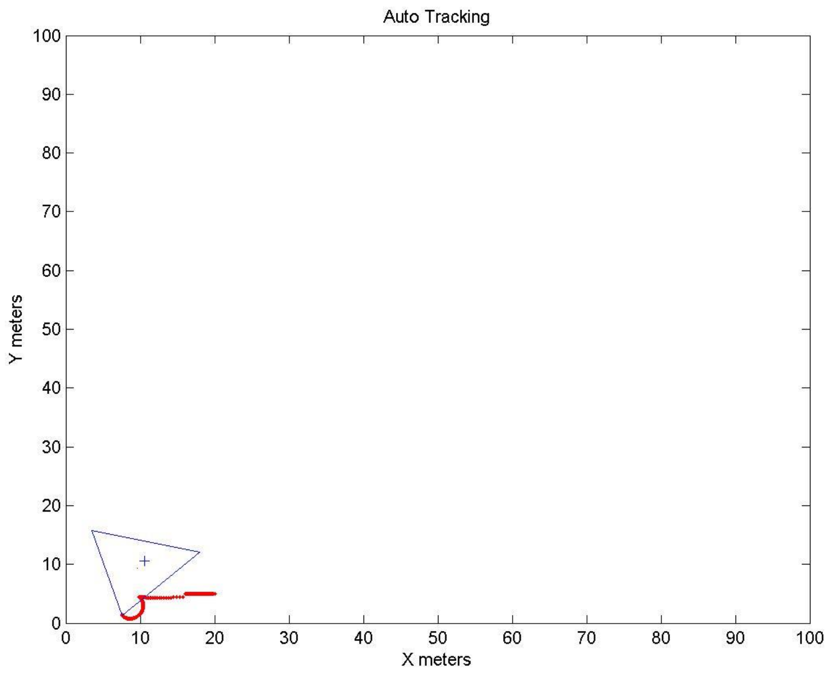


Figure 29: Auto Tracking at Frame 160

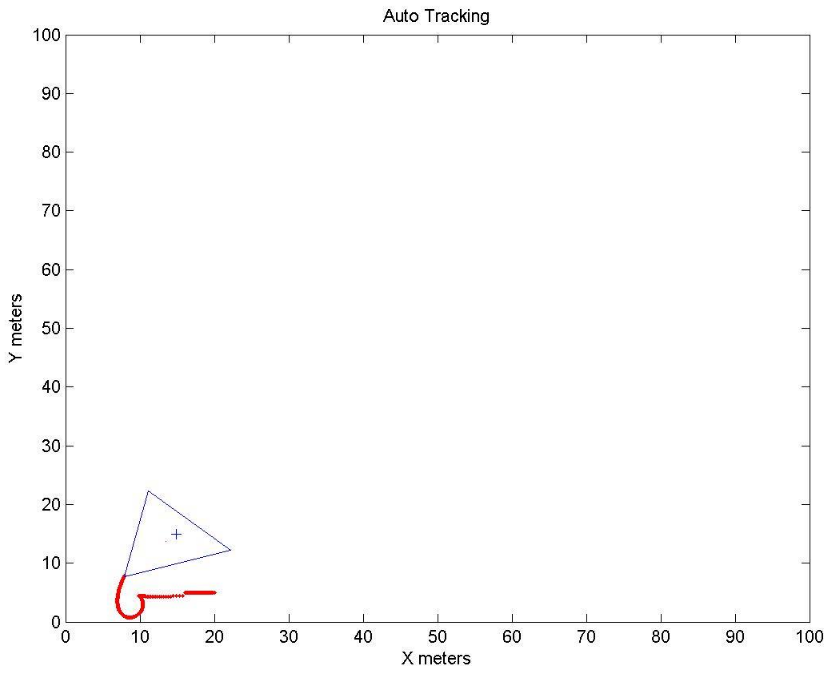


Figure 30: Auto Tracking at Frame 225

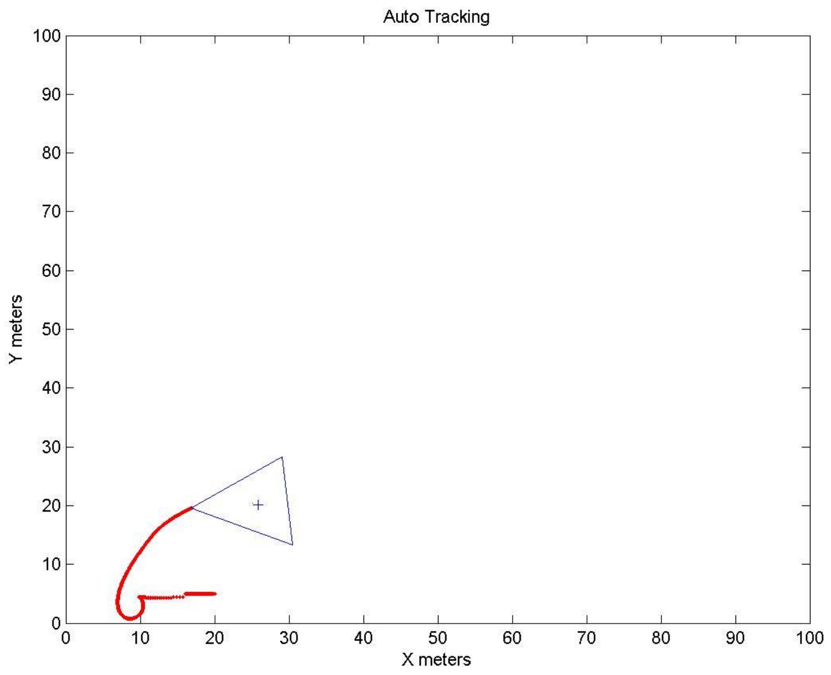


Figure 31: Auto Tracking at Frame 360

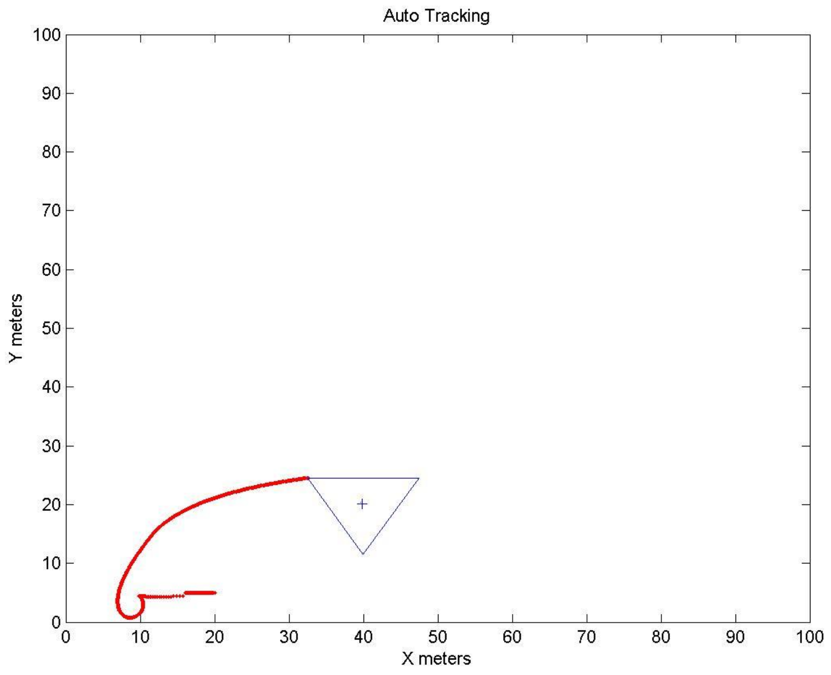


Figure 32: Auto Tracking at Frame 500

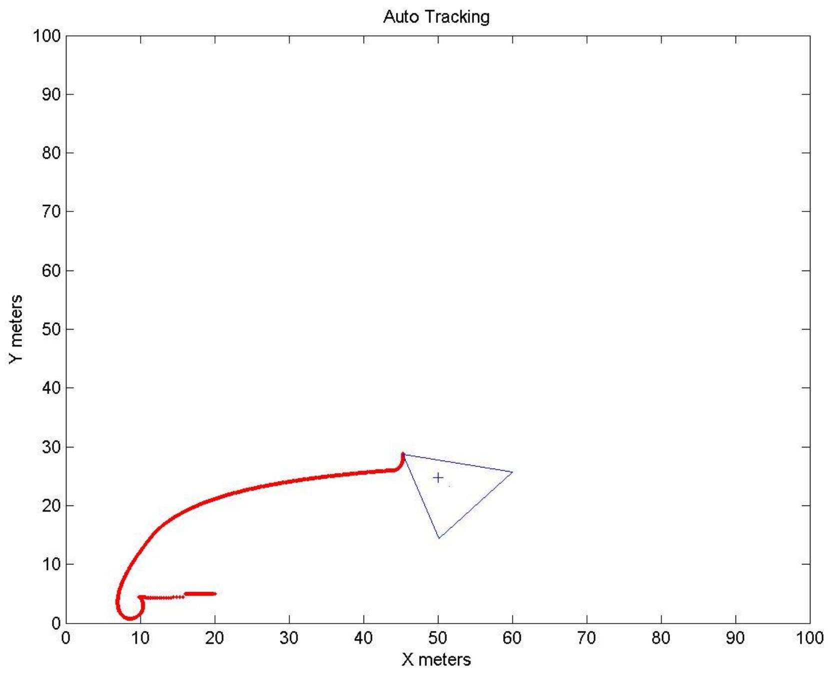


Figure 33: Auto Tracking at Frame 630

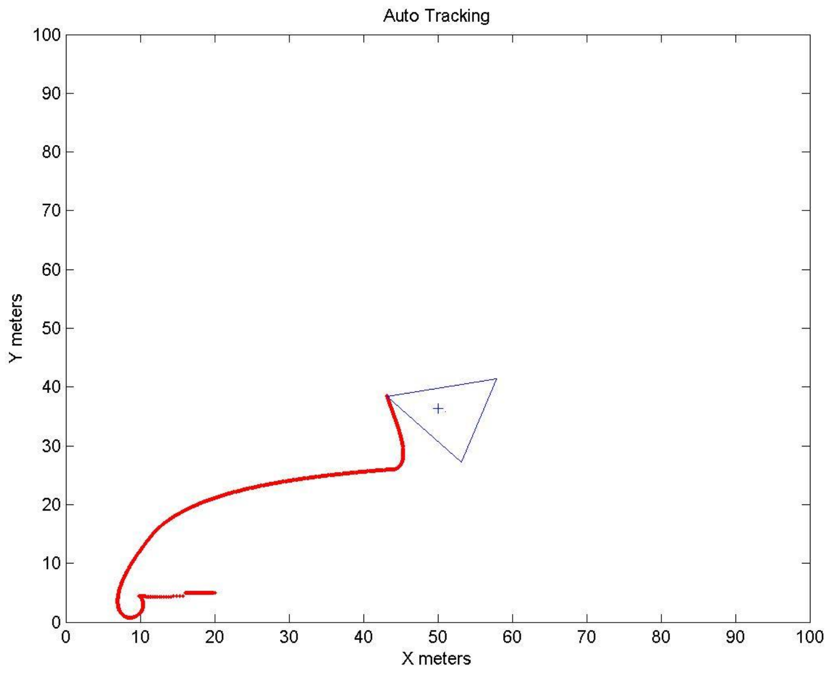


Figure 34: Auto Tracking at Frame 700

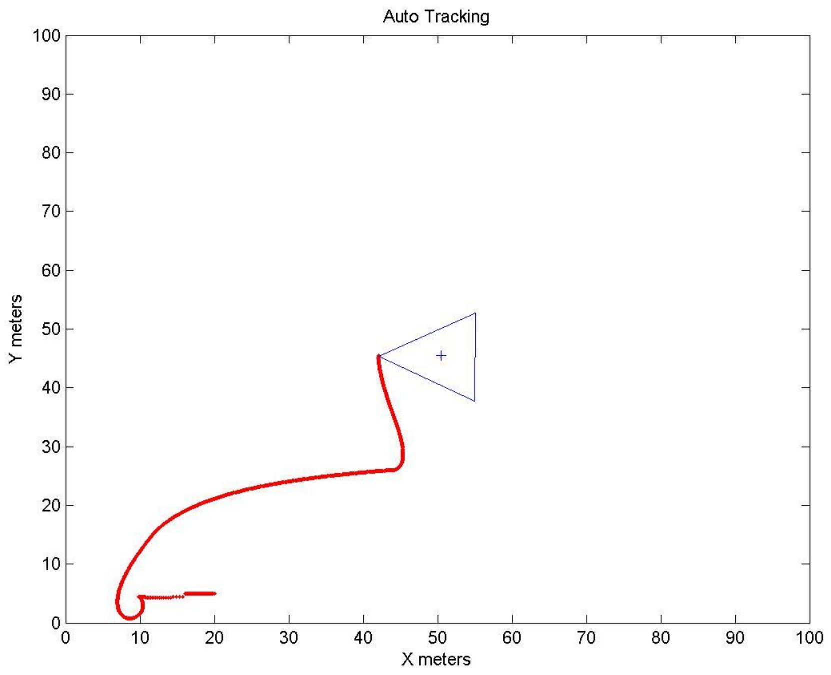


Figure 35: Auto Tracking at Frame 755

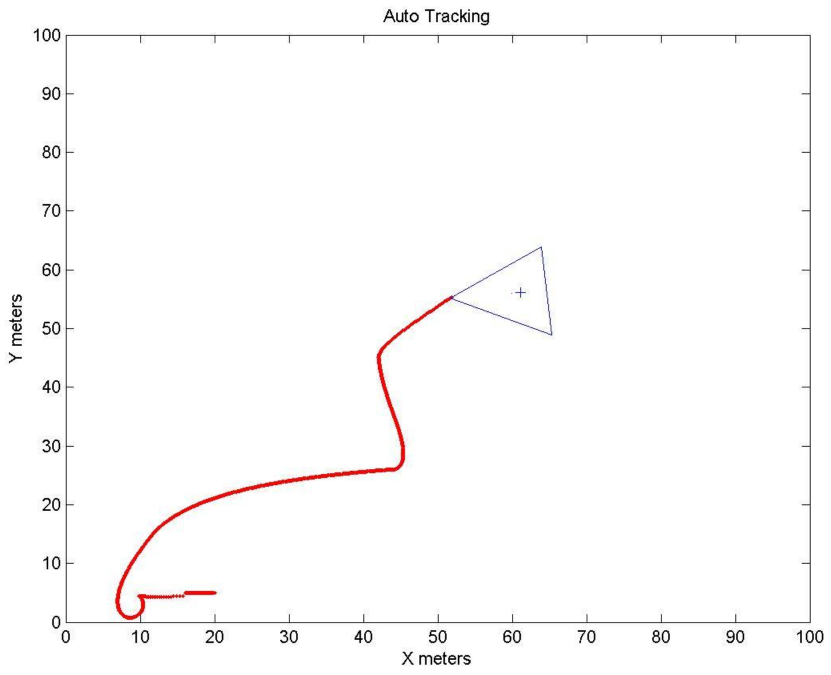


Figure 36: Auto Tracking at Frame 835

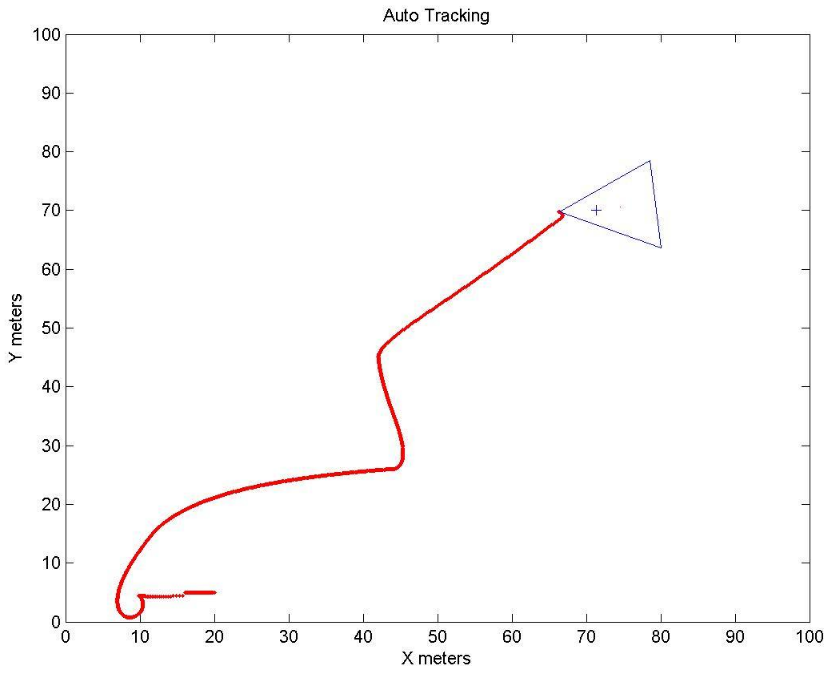


Figure 37: Auto Tracking at Frame 950

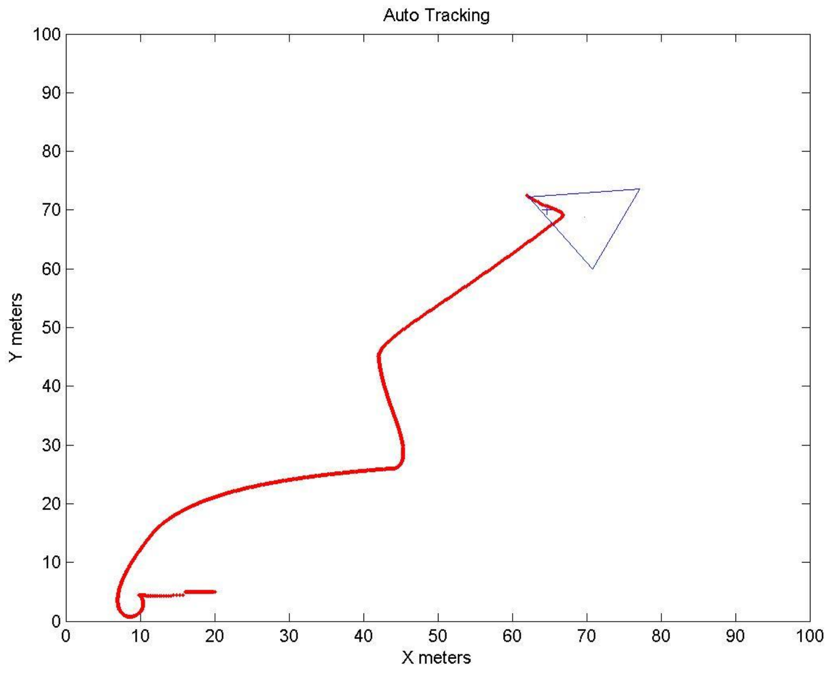


Figure 38: Auto Tracking at Frame 970

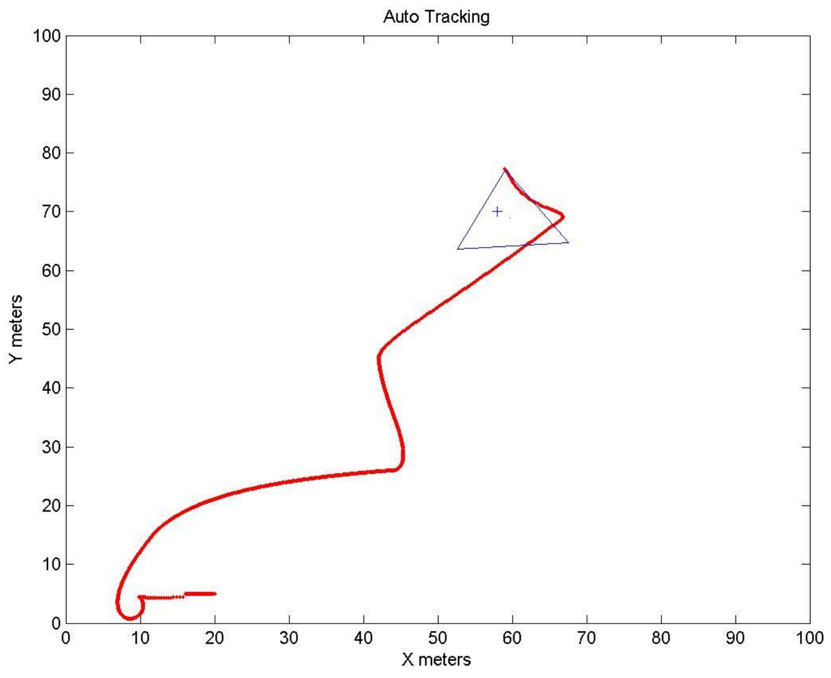


Figure 39: Auto Tracking at Frame 990

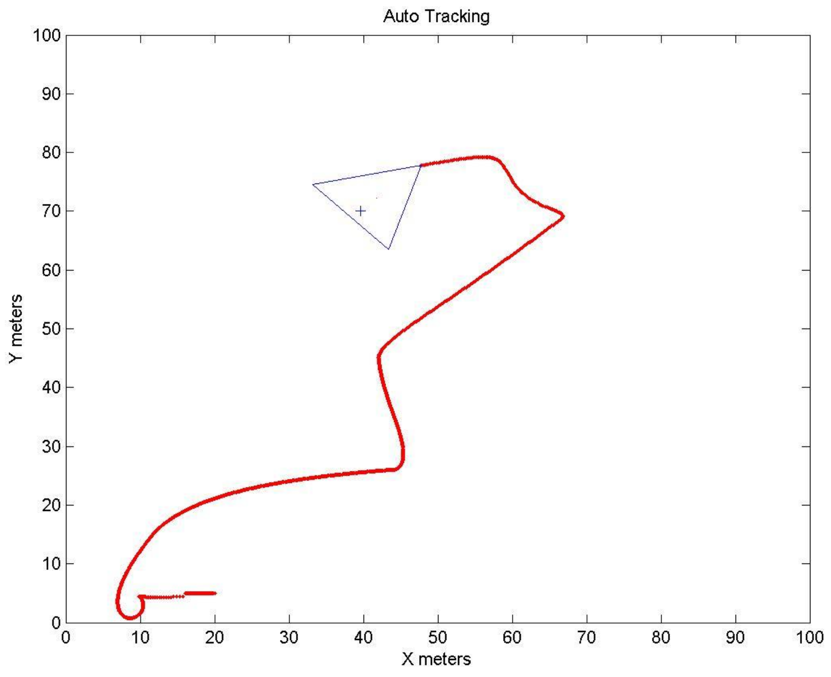


Figure 40: Auto Tracking at Frame 1050

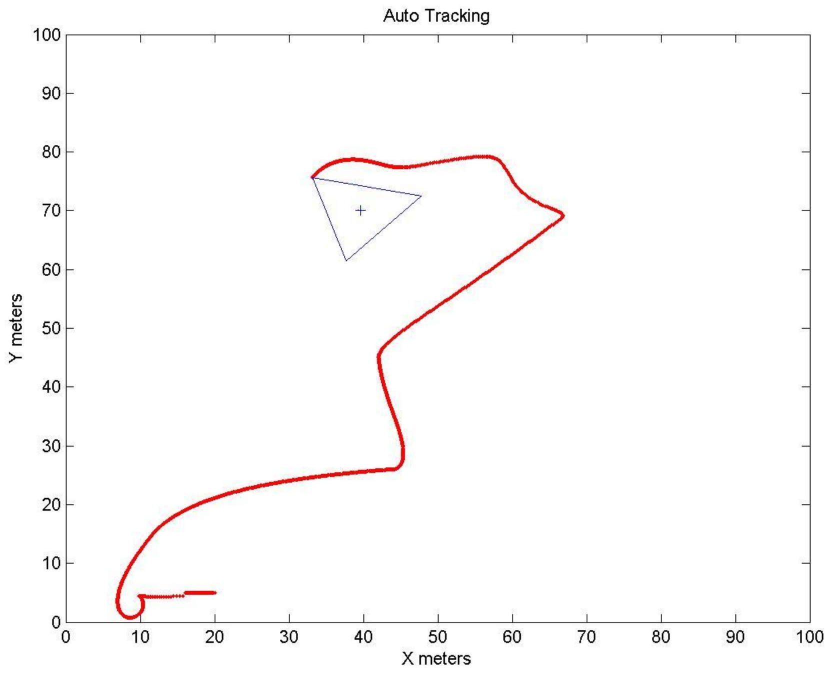


Figure 41: Auto Tracking at Frame 1300

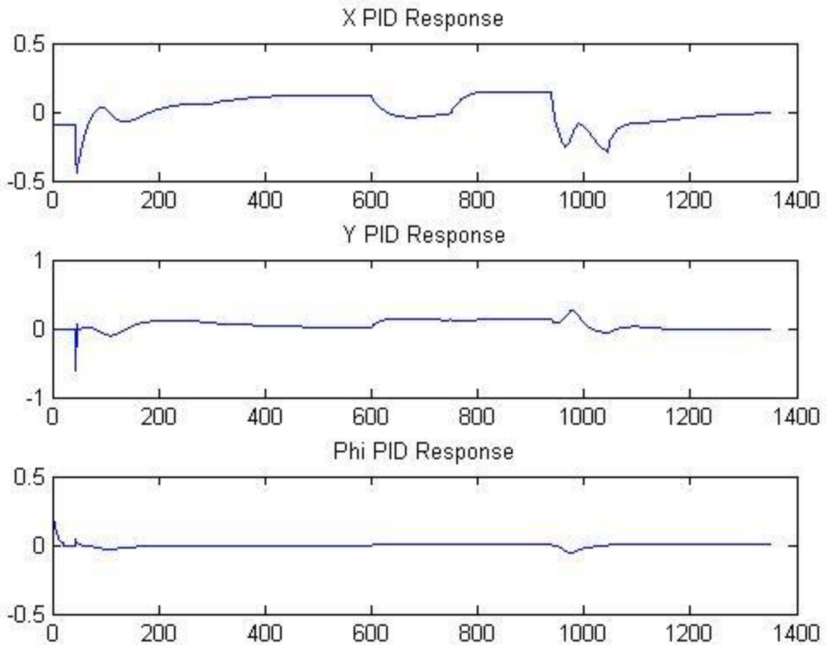


Figure 42: Auto Track Error Response

Station Keeping Tracking

The Station Keeping strategy is created by guiding the input error of the yaw PID controller. The following operations are performed:

(13)

The current yaw of the target object plus the input *Phi*$_d$ desired relative yaw angle creates the setpoint yaw.

(14)

The error is then the setpoint minus the current yaw, which is fed into the PID yaw controller. If the error is too high and the target is close to the edge of the FOV, then the yaw controller is reverted back to the auto track mode and the error is the relative yaw from the target to the quadrotor. This prevents the target from being lost when a target trajectory with high velocities is observed.

An example of the strategy simulation is shown below using the previously noted target trajectory where *Phi*$_d$ = 90 degrees. The initial condition is shown where the target starts at 0,0 and the quadrotor is at 20,5,50.

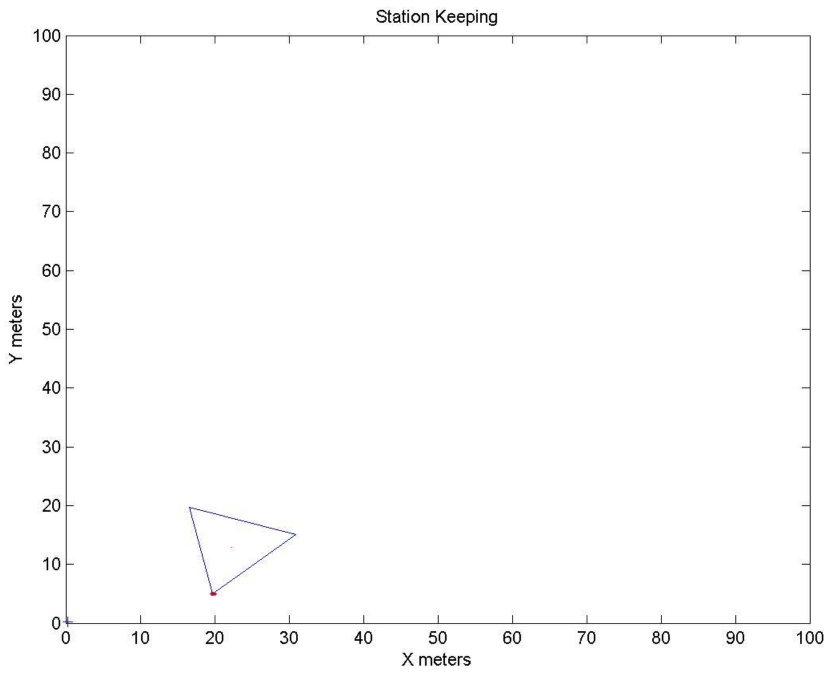


Figure 43: Station Keeping at Frame 0

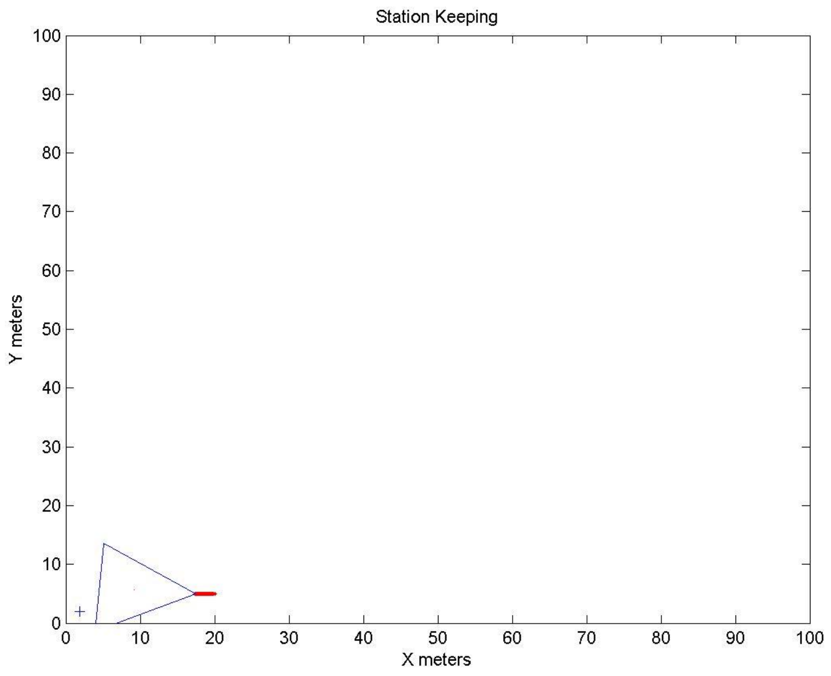


Figure 44: Station Keeping at Frame 30

The quadrotor moves into the uninformed search state before the target is detected. When the target is detected, the station keeping state is called and the quadrotor begins to move into the desired position.

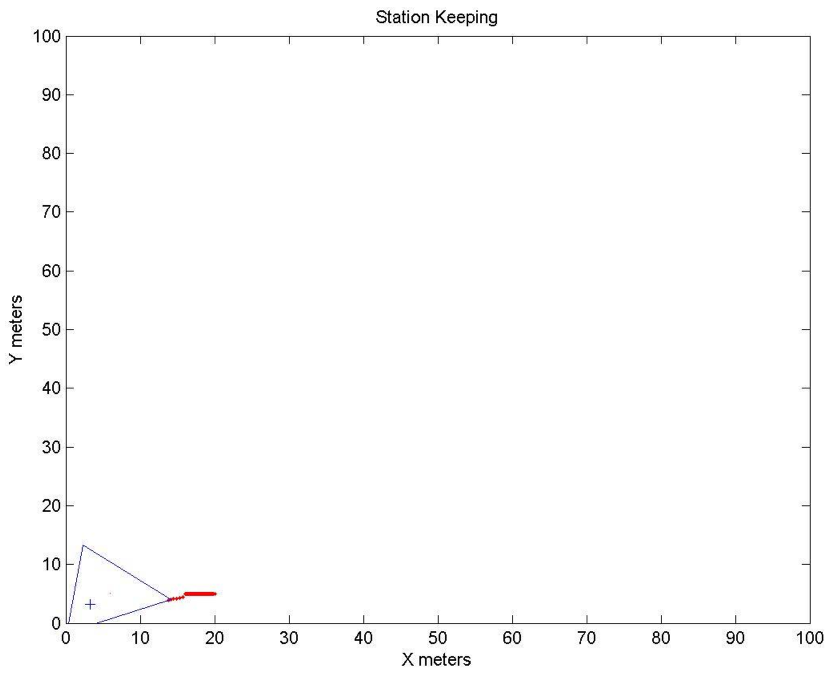


Figure 45: Station Keeping at Frame 50

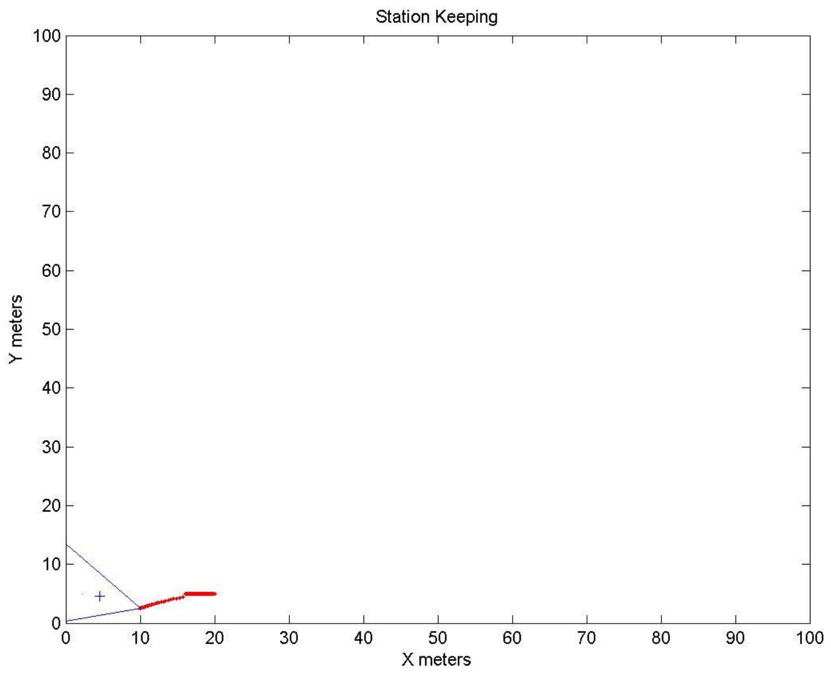


Figure 46: Station Keeping at Frame 70

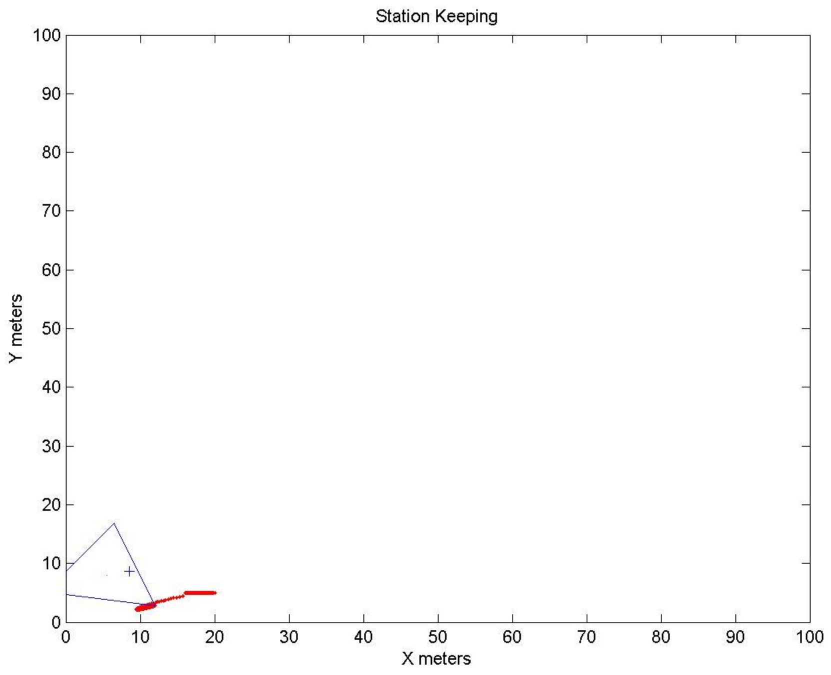


Figure 47: Station Keeping at Frame 130

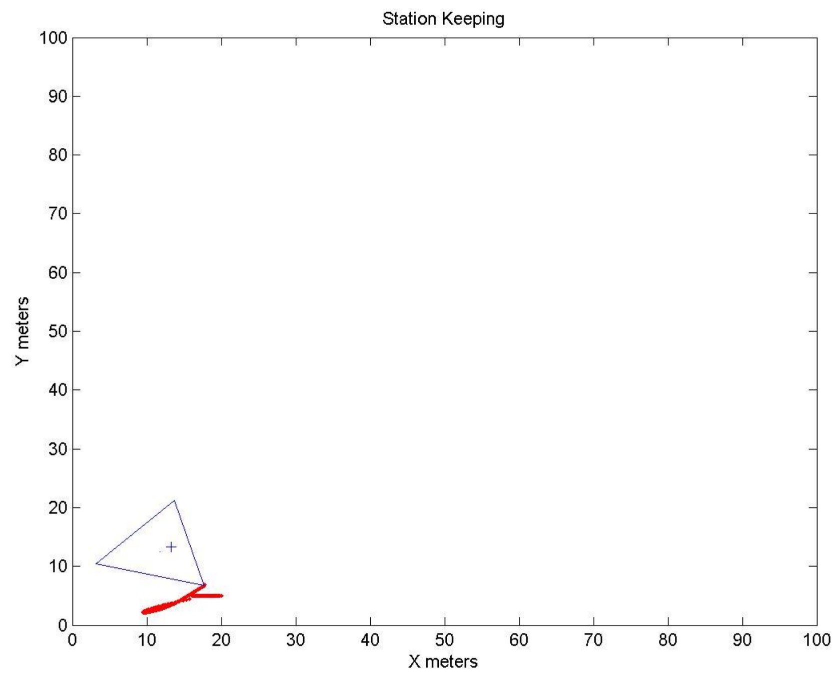


Figure 48: Station Keeping at Frame 200

As the target trajectory changes, the target yaw is update and the quadrotor yaw is updated.

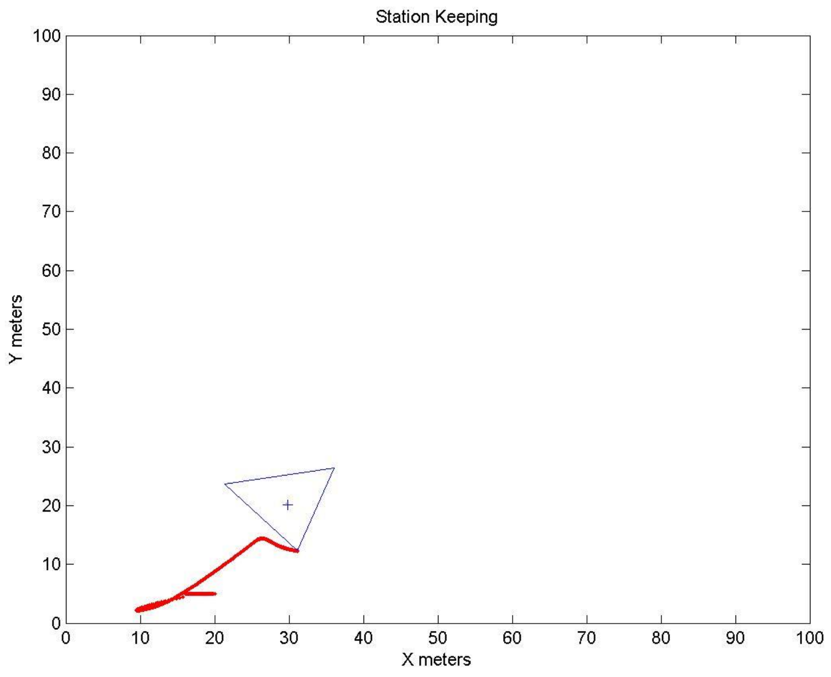


Figure 49: Station Keeping at Frame 400

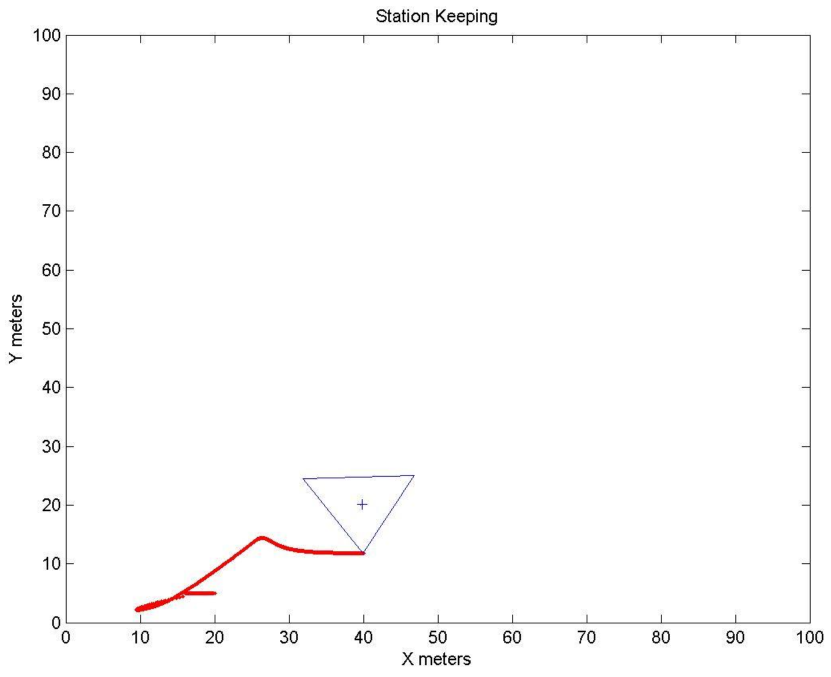


Figure 50: Station Keeping at Frame 500

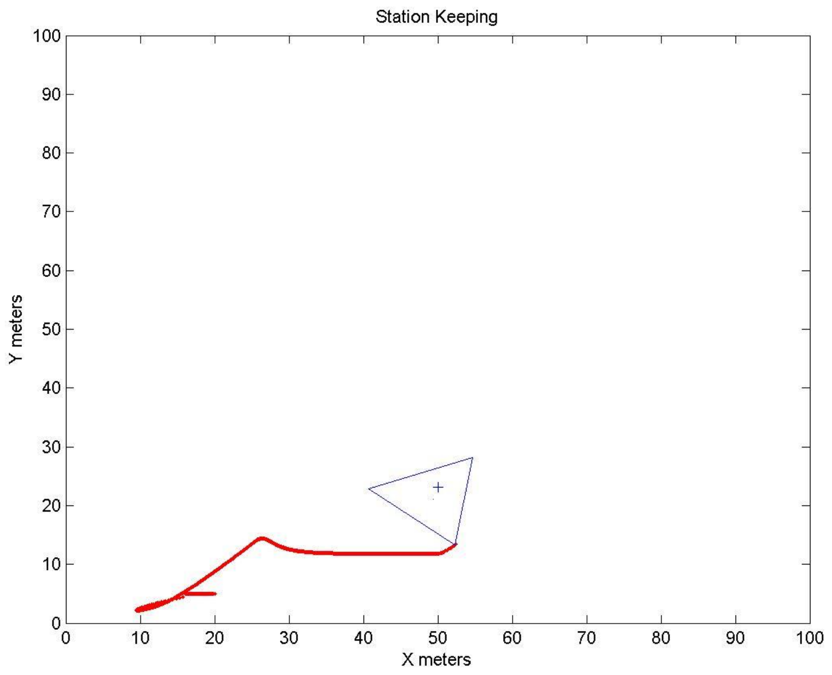


Figure 51: Station Keeping at Frame 620

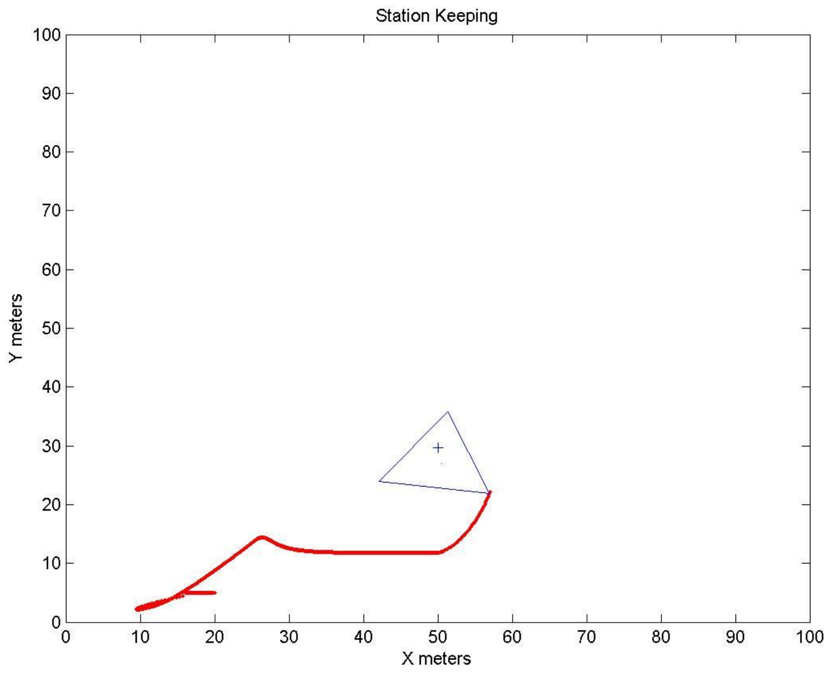


Figure 52: Station Keeping at Frame 660

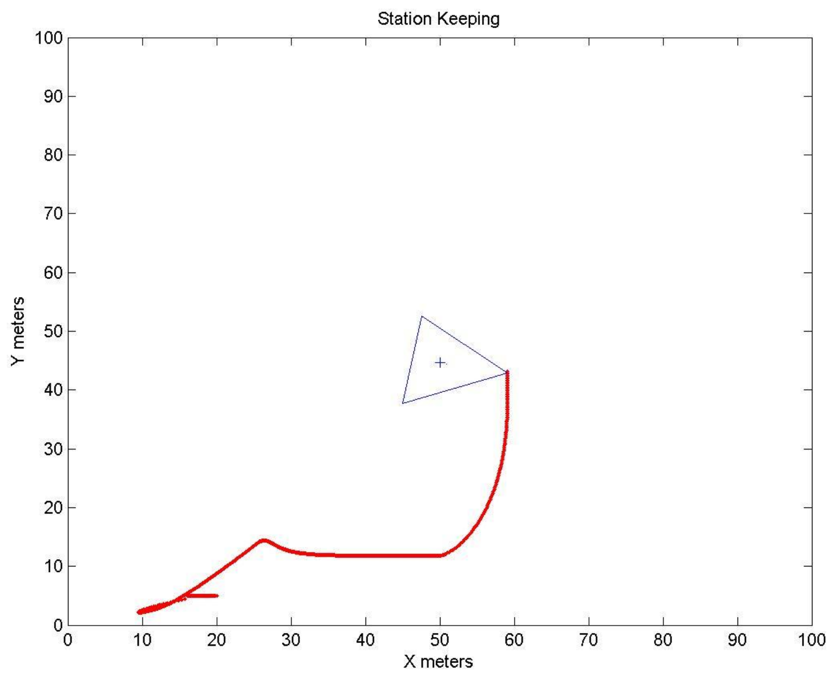


Figure 53: Station Keeping at Frame 750

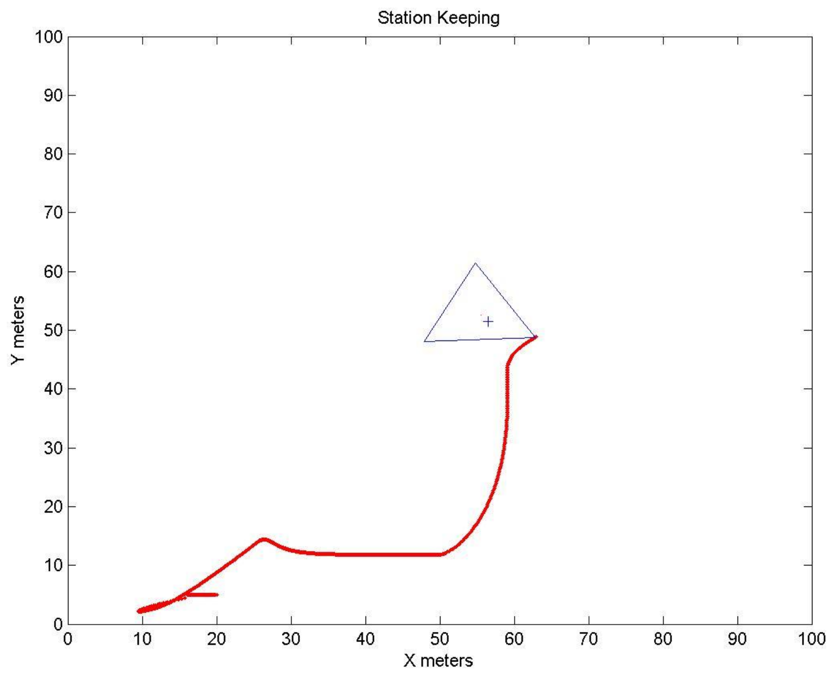


Figure 54: Station Keeping at Frame 800

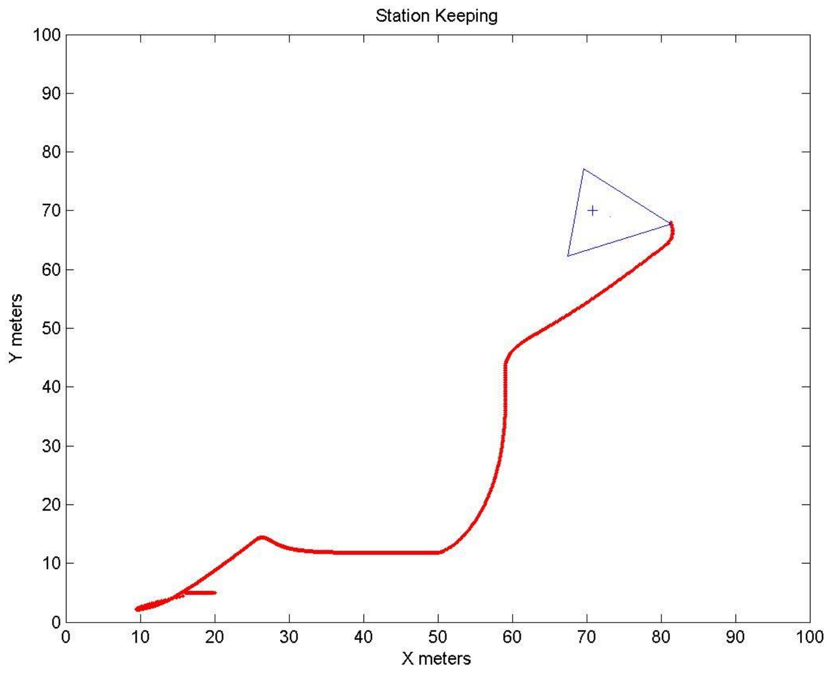


Figure 55: Station Keeping at Frame 960

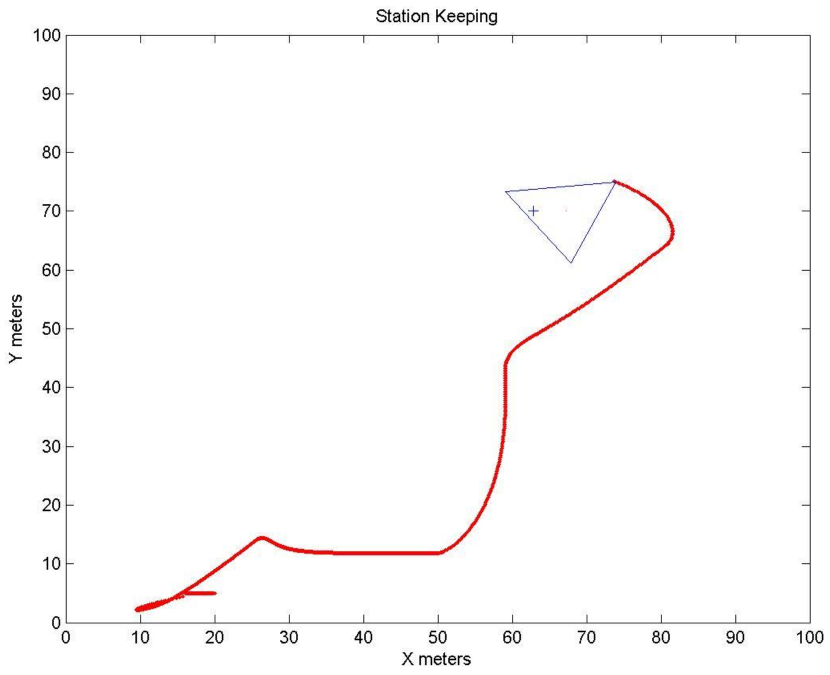


Figure 56: Station Keeping at Frame 1000

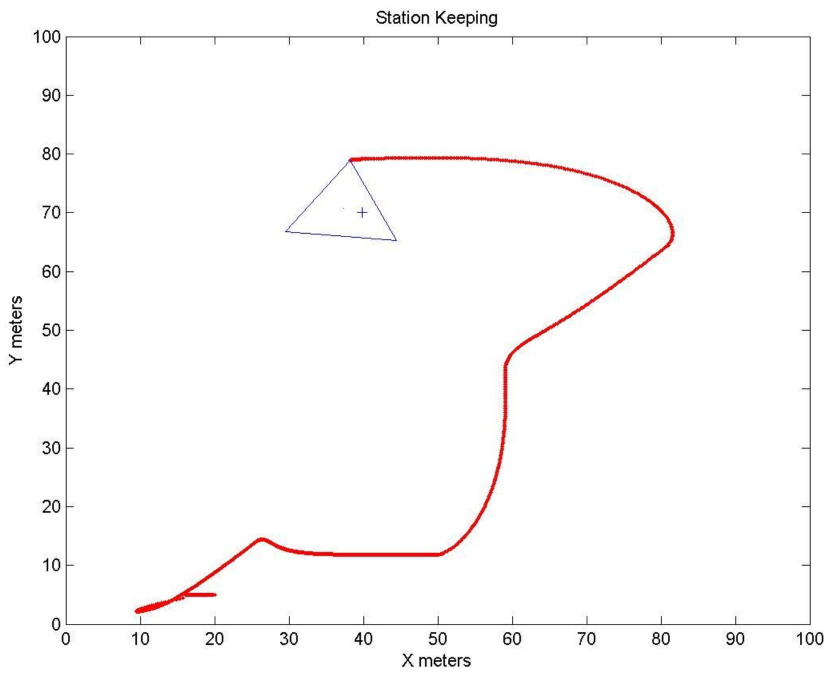


Figure 57: Station Keeping at Frame 1150

At frame 1200, the target velocity drops to zero so the steady state response can be observed.

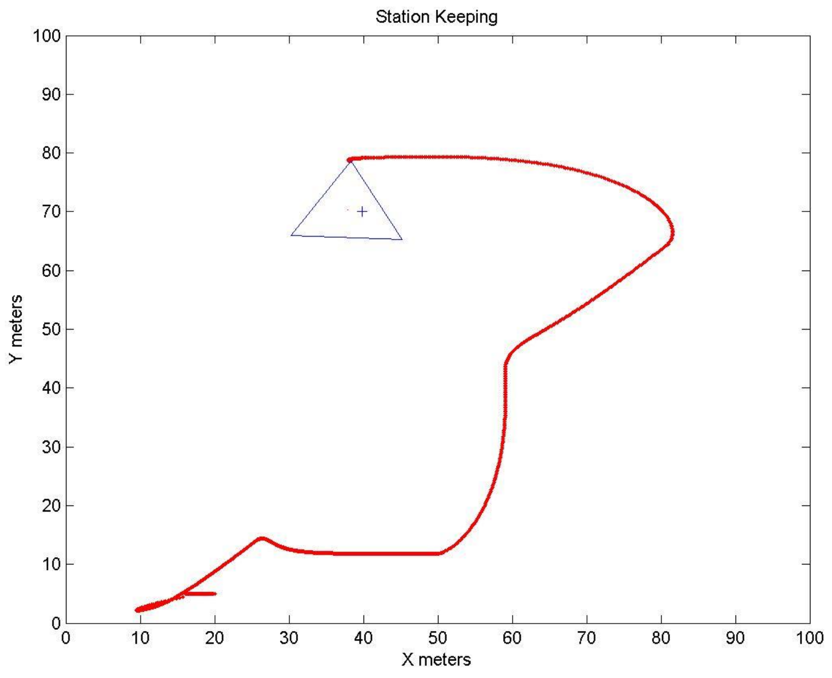


Figure 58: Station Keeping at Frame 1200

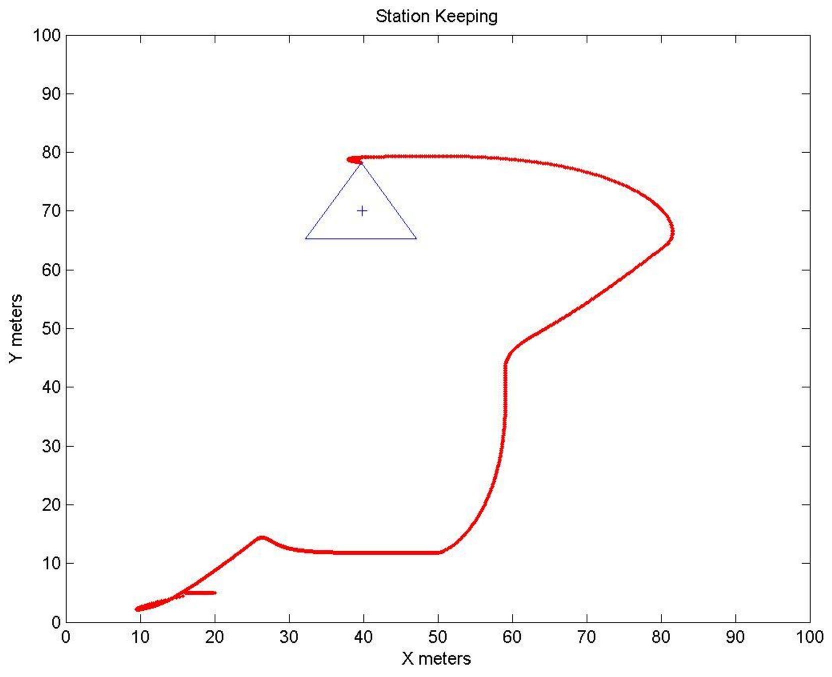


Figure 59: Station Keeping at Frame 1400

After the trajectory and tracking is rendered, the associated PID error response for X,Y, and Phi are illustrated below.

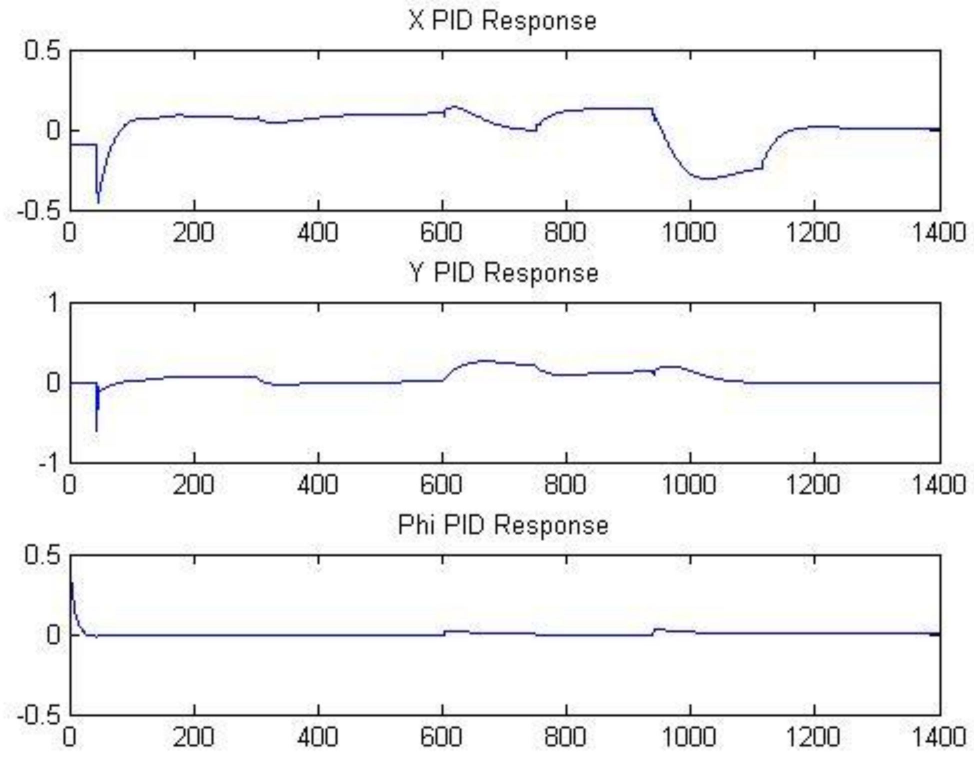


Figure 60: Station Keeping Strategy Error Response

Desired responses are shown in that there is low ripple, overshoot is low, and steady-state error is within acceptable limits.

Circle Tracking

The Circle strategy creates a constant error in the yaw controller which forces constant rotation.

(15)

Where

(16)

This allows $Rate_{phi}$ to stay in terms of radians per second, which is converted from degrees to allow for easy user input.

In the following illustration, $Rate_{phi}$ is set to 2 degrees per second. A higher $Rate_{phi}$ will cause proportionally higher input error which can cause instability. This was typically observed where $Rate_{phi} > 5$.

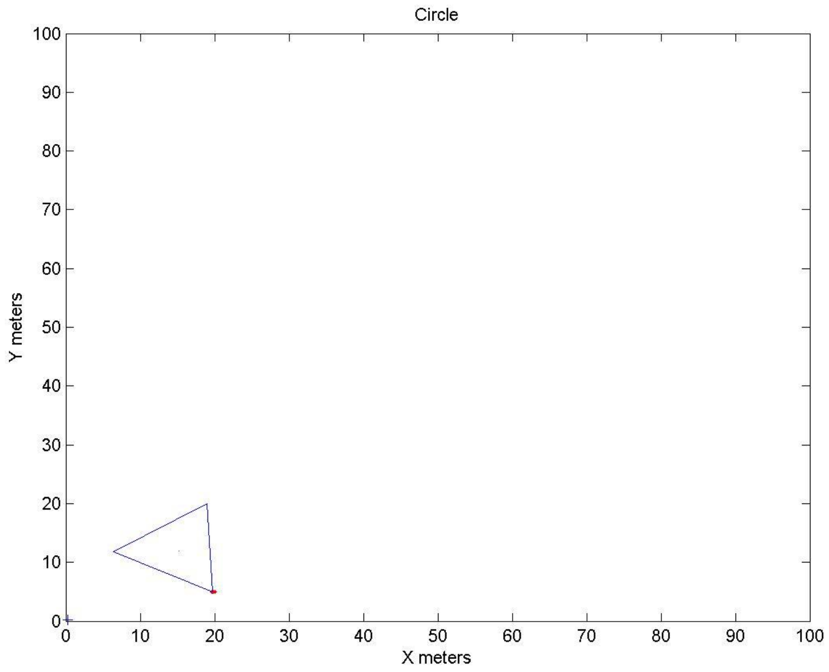


Figure 61: Circle at frame 0

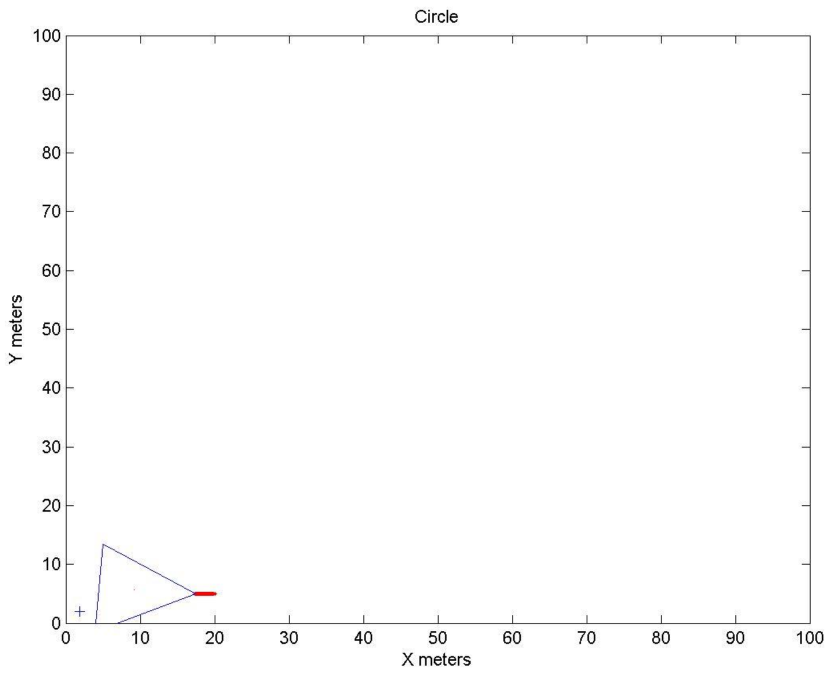


Figure 62: Circle at frame 30

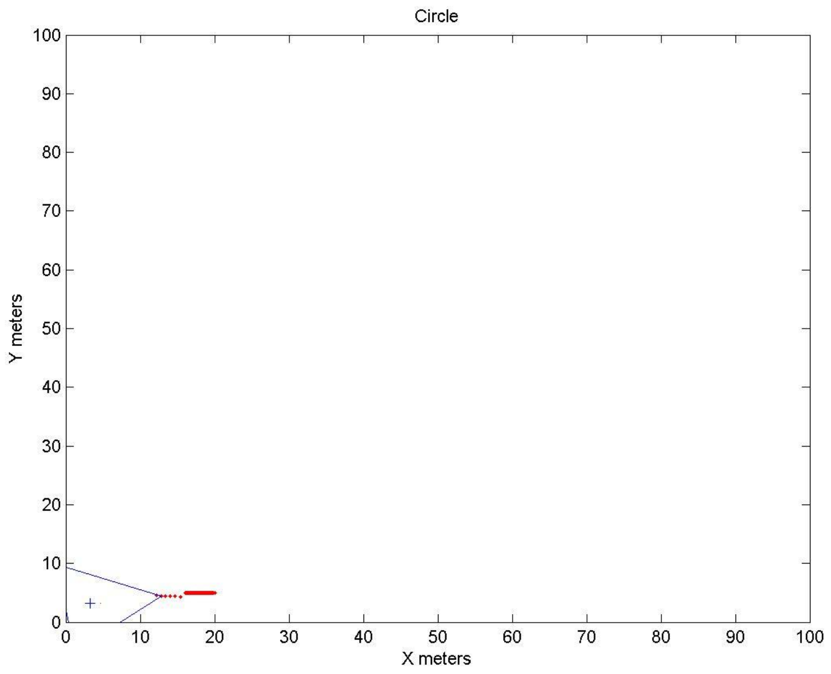


Figure 63: Circle at frame 50

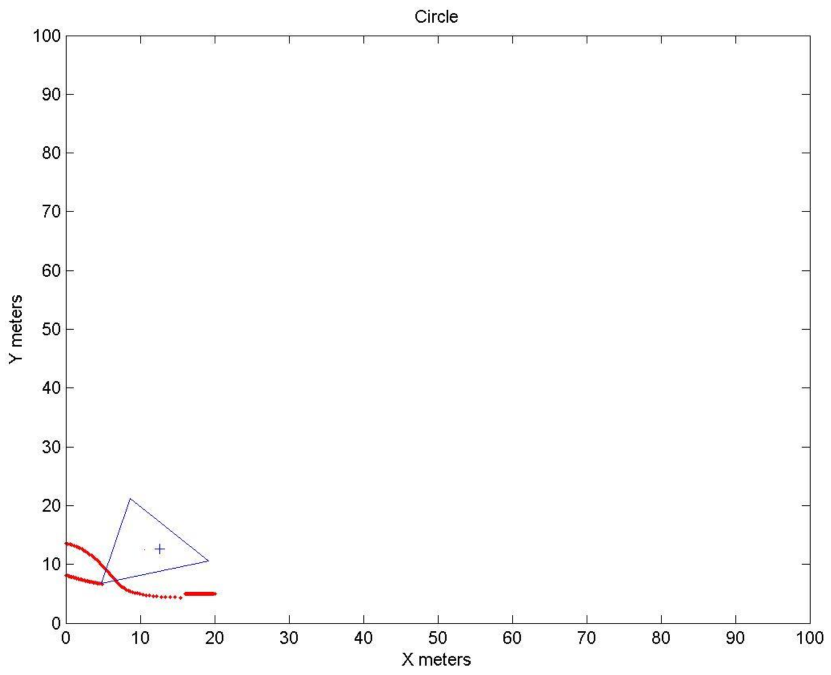


Figure 64: Circle at frame 190

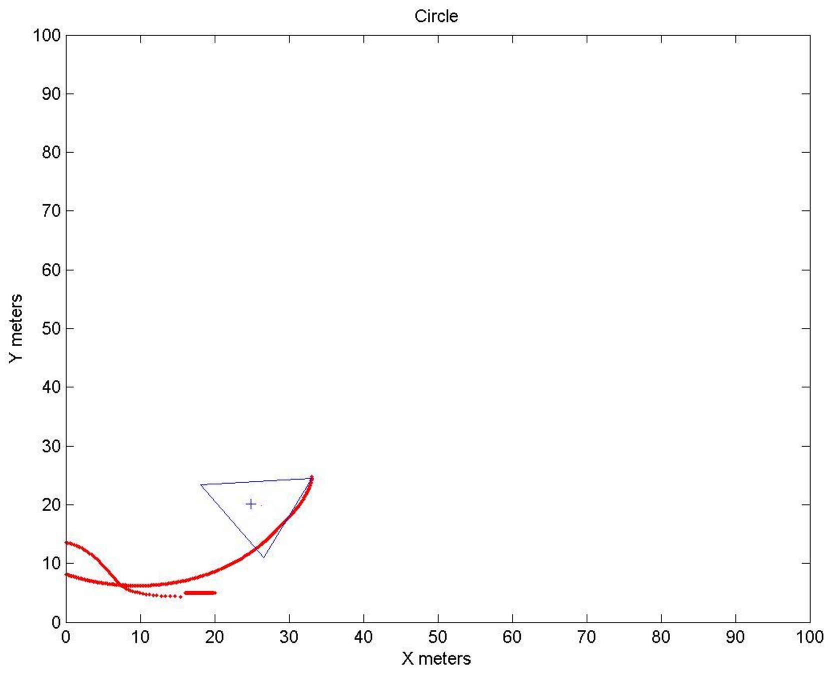


Figure 65: Circle at frame 350

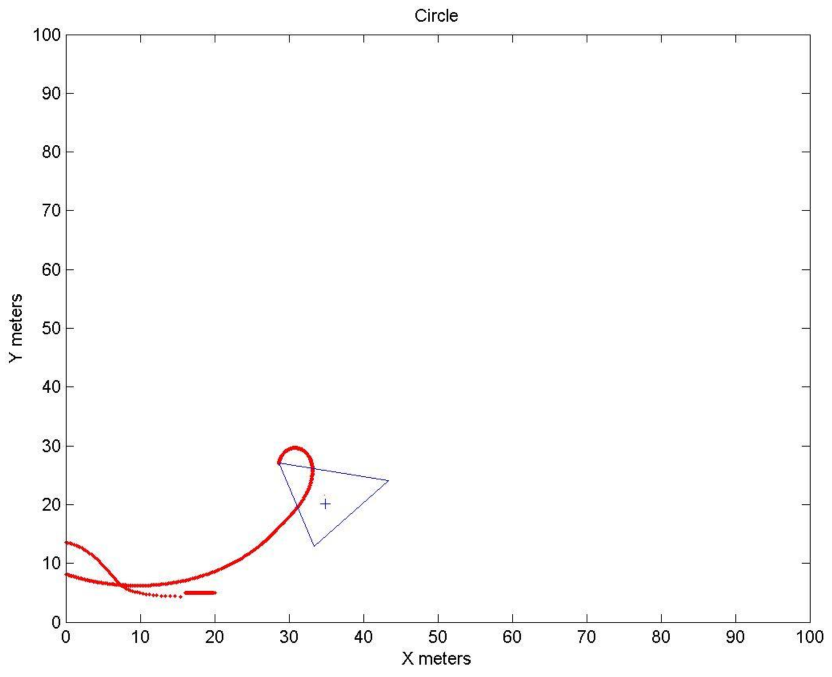


Figure 66: Circle at frame 450

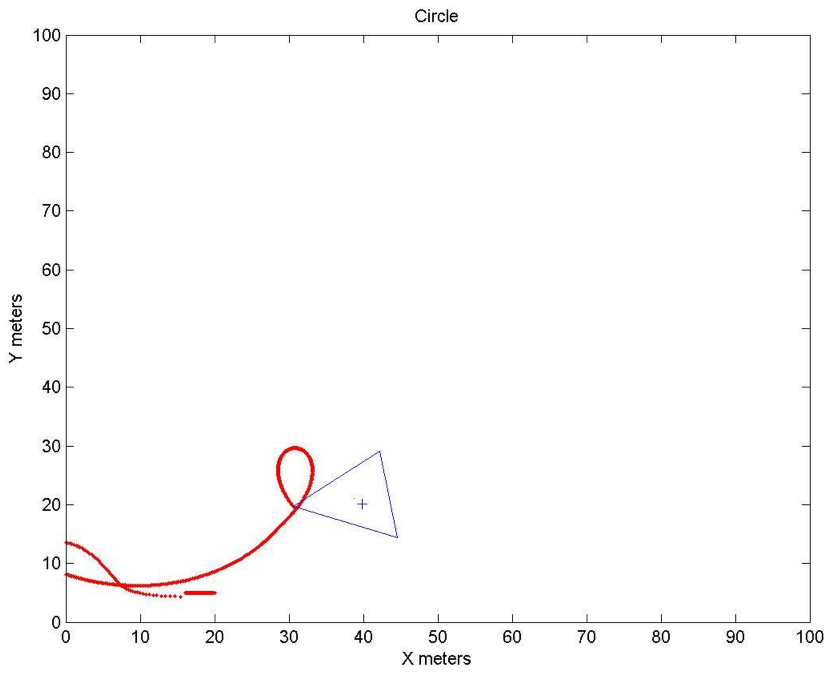


Figure 67: Circle at Frame 500

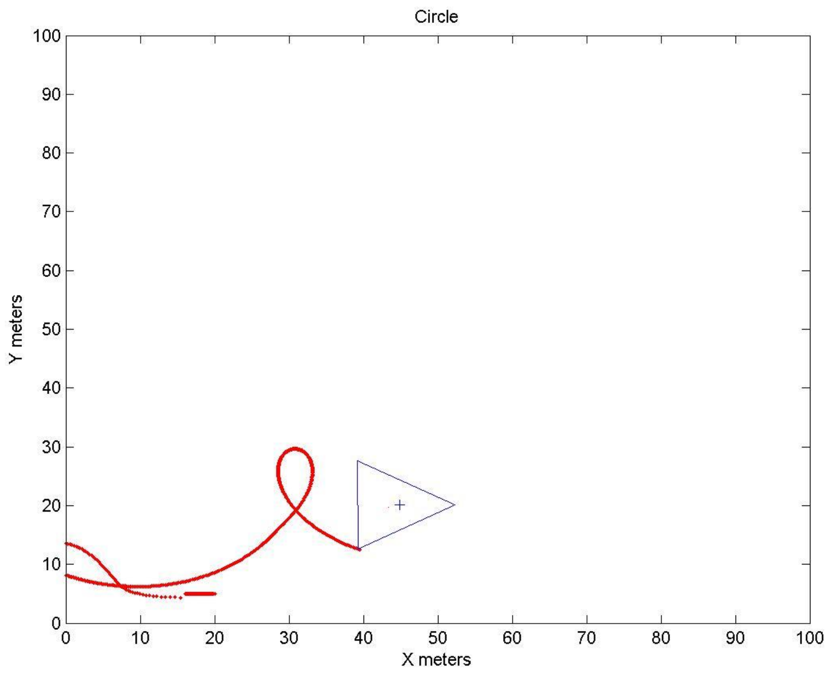


Figure 68: Circle at frame 550

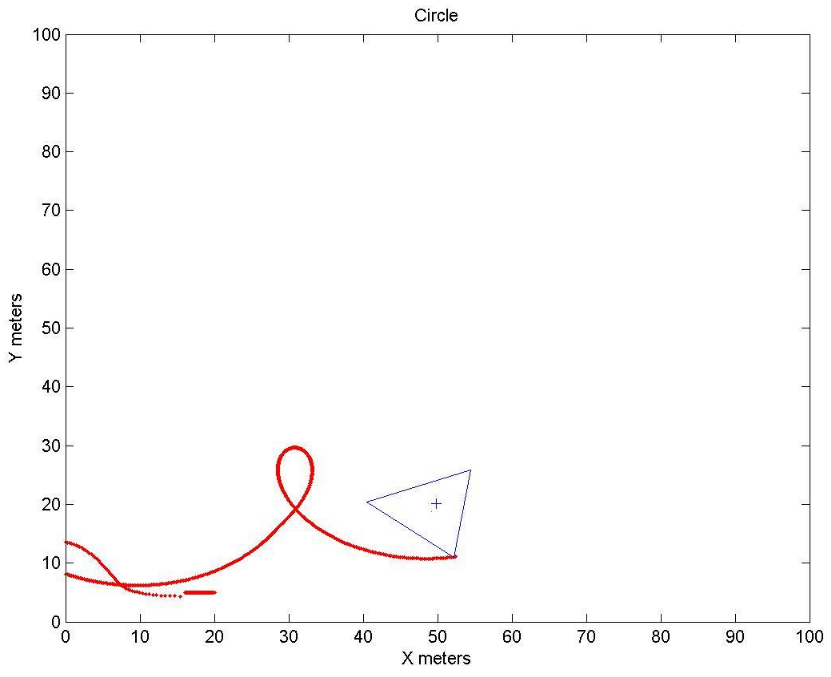


Figure 69: Circle at frame 600

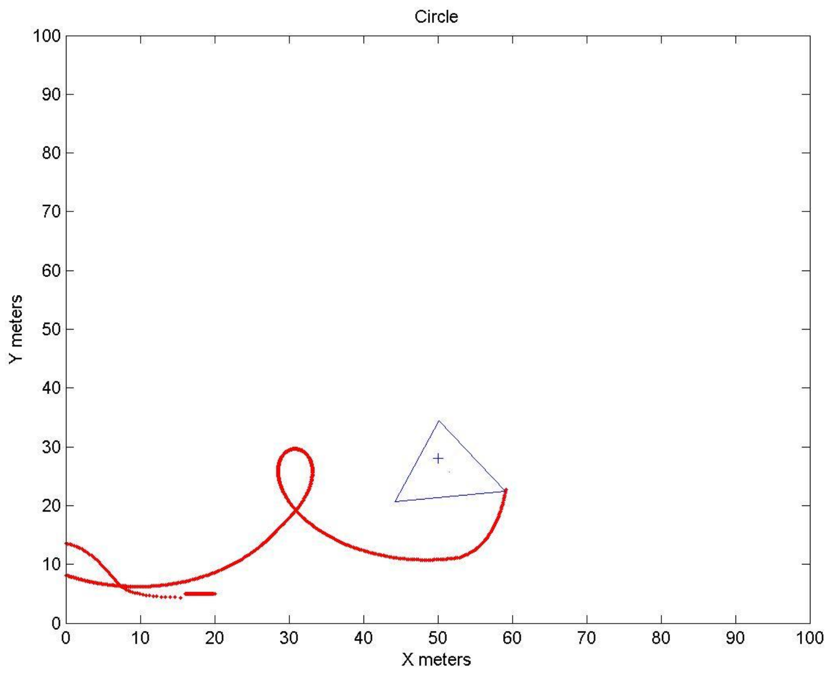


Figure 70: Circle at frame 650

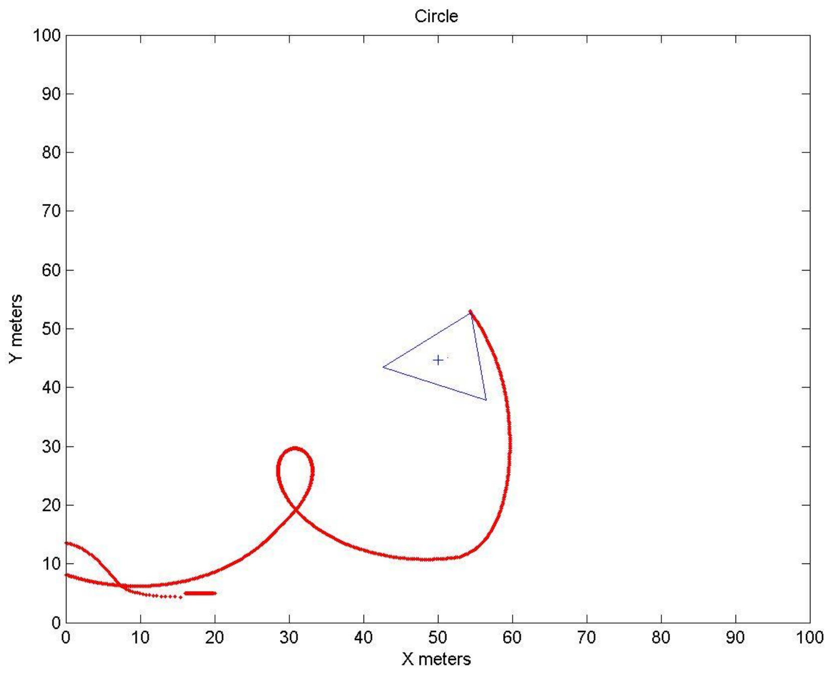


Figure 71: Circle at frame 750

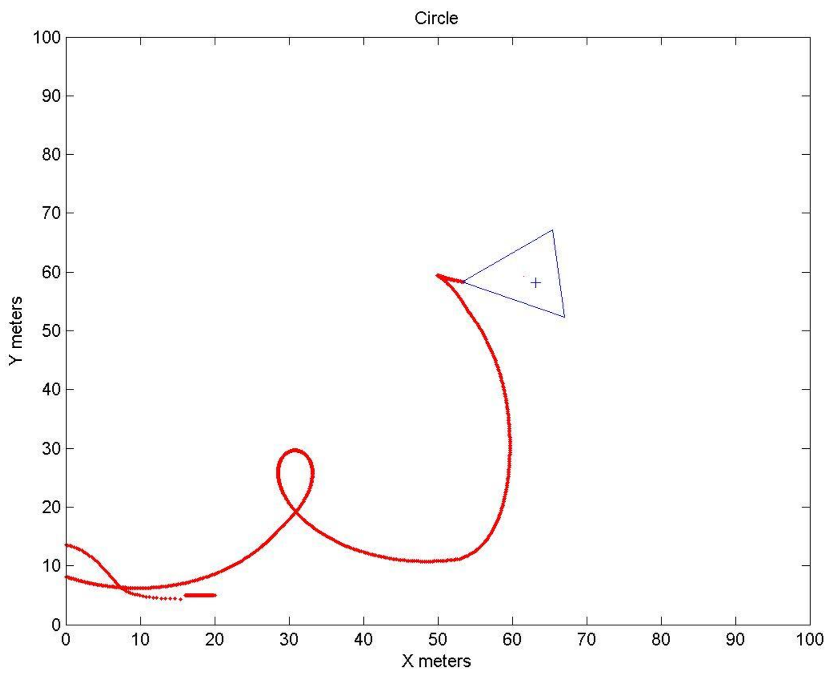


Figure 72: Circle at frame 850

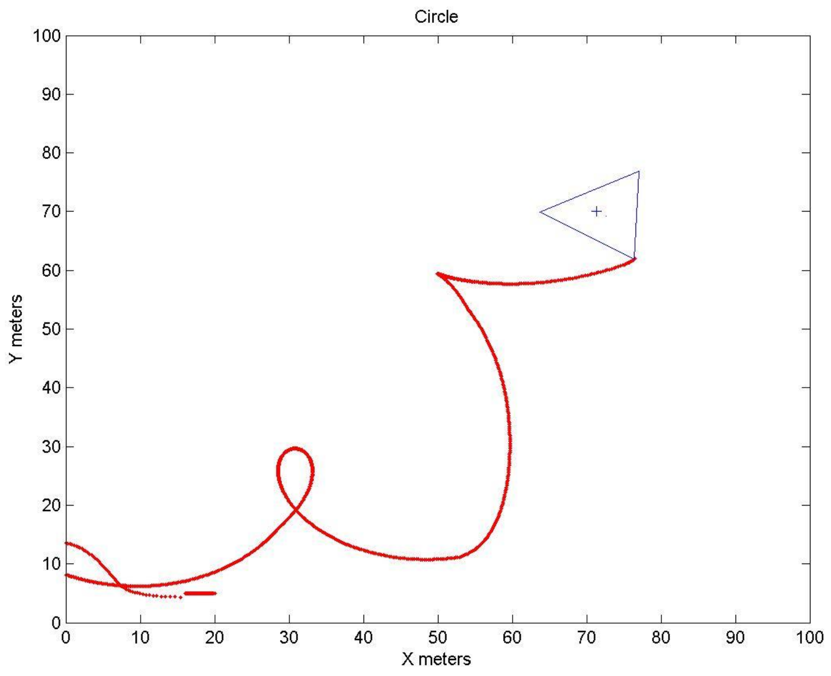


Figure 73: Circle at frame 950

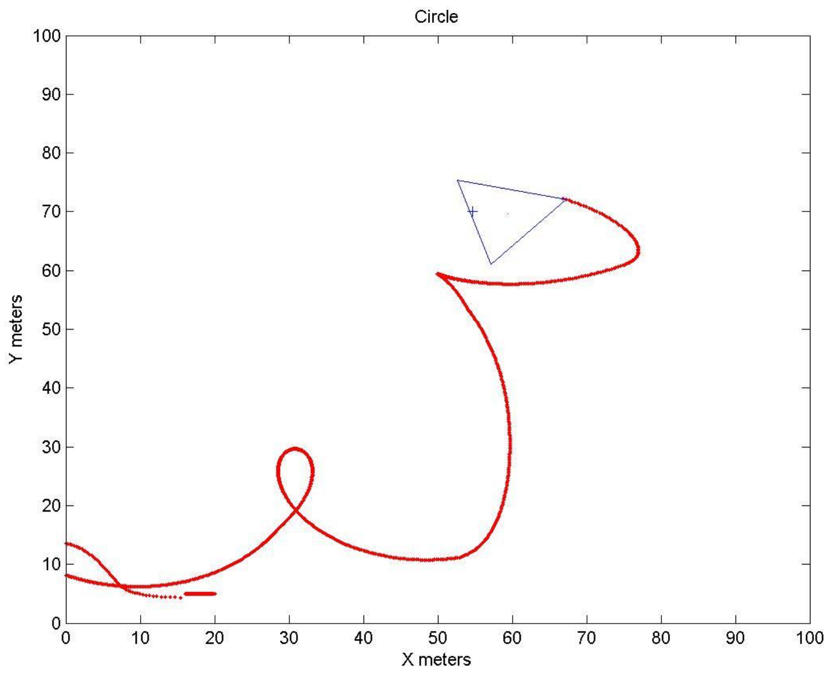


Figure 74: Circle at frame 1000

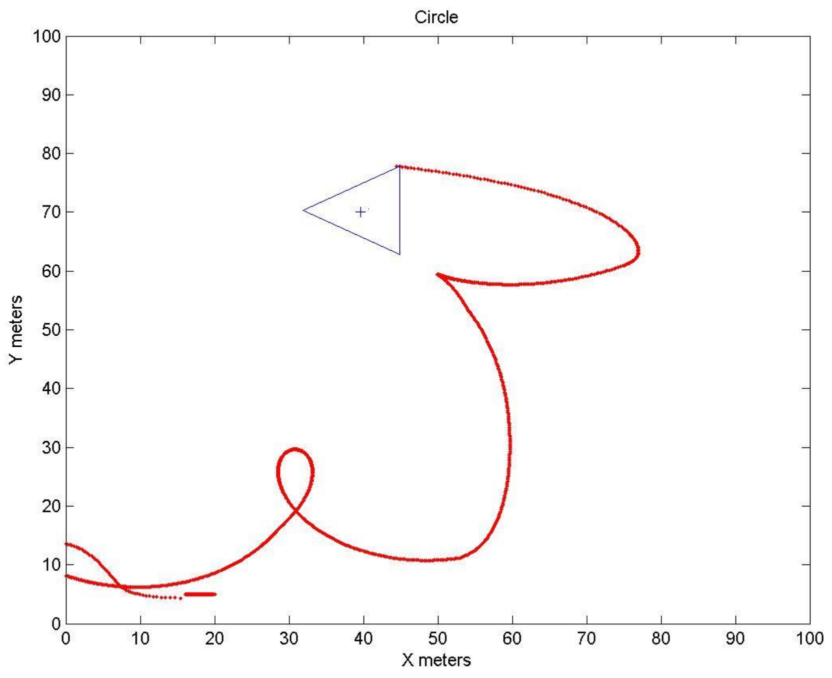


Figure 75: Circle at frame 1050

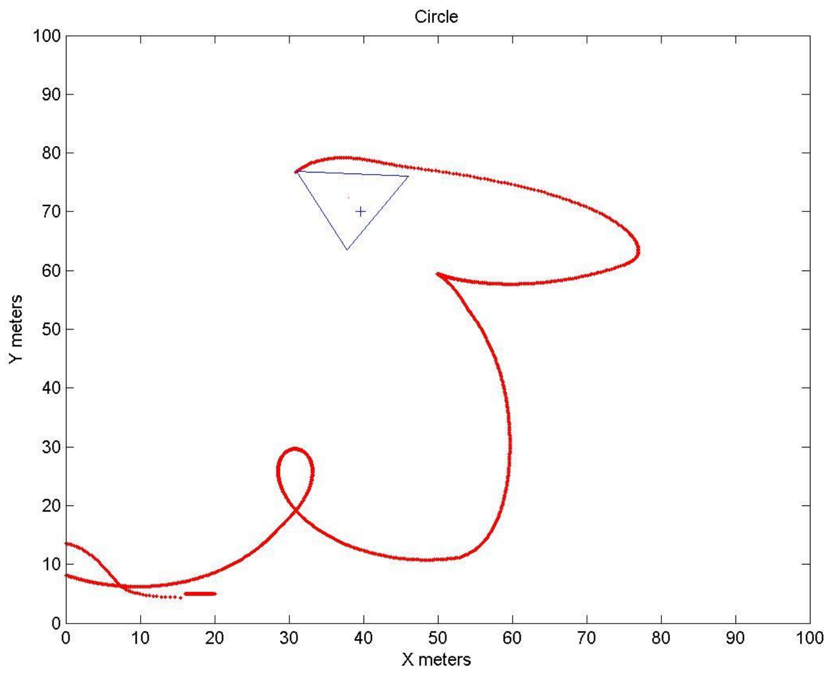


Figure 76: Circle at frame 1100

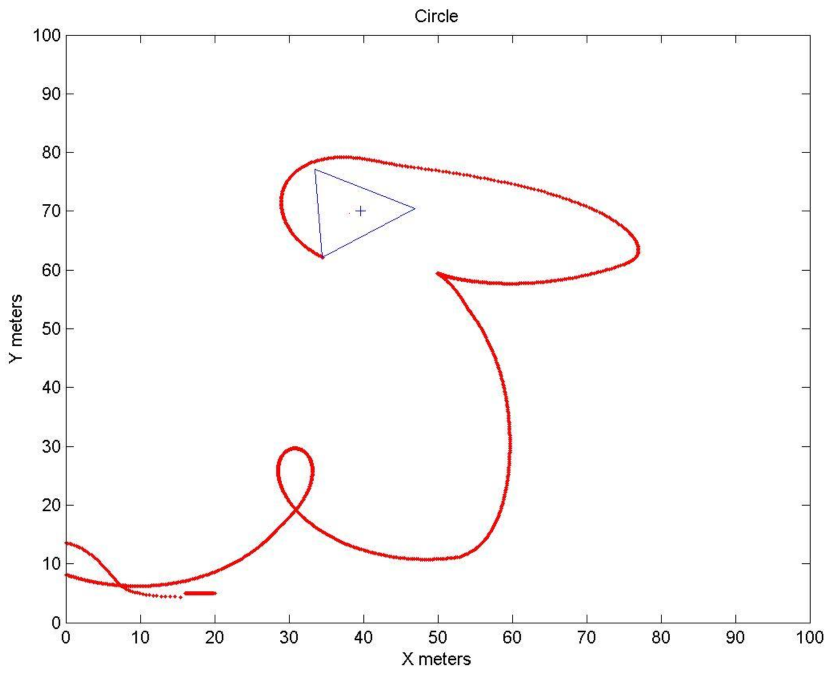


Figure 77: Circle at frame 1200

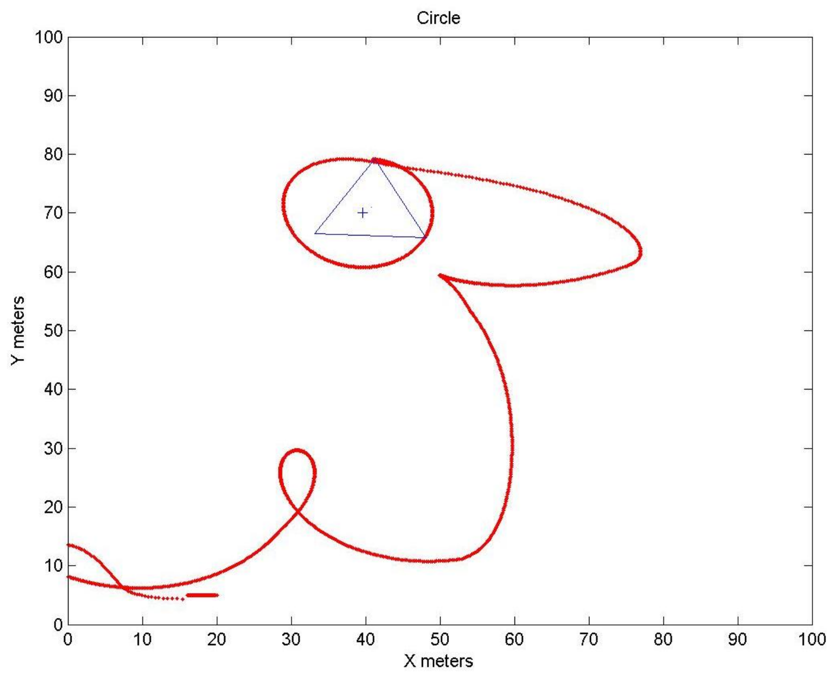


Figure 78: Circle at frame 1400

After the trajectory and tracking is rendered, the associated PID error response for X,Y, and Phi are illustrated below. X and Y responses are partially sinusoidal as expected. Phi contains a constant steady state error due to the method of inducing the rotation.

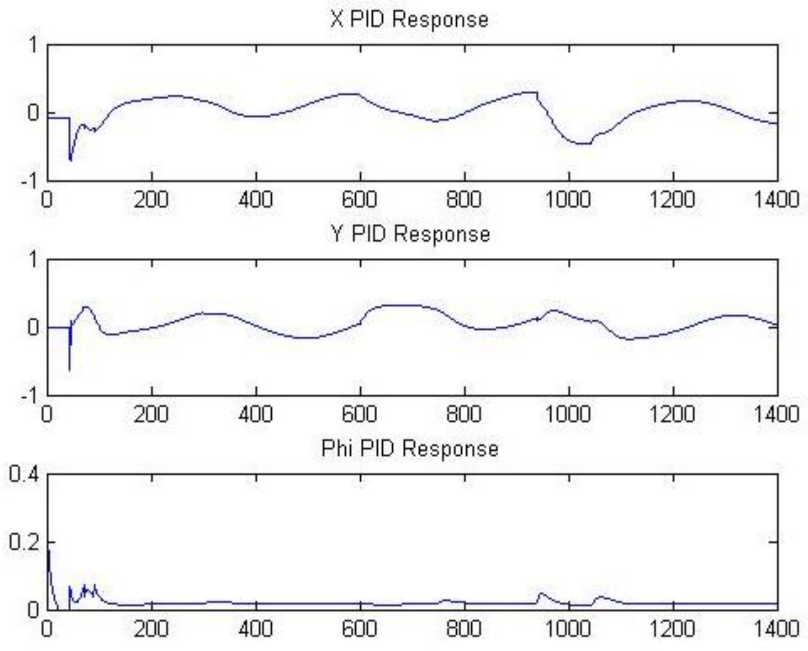


Figure 79: Circle Strategy Error Response

System Design and Implementation

Weight Based Airframe Design

MathCAD was used to find the target weight of the quadrotor. The manufacturer estimates each motor combined with a 8x4.3 propeller produce approximately 1000g of thrust. Four motors provide a total of 4000g of thrust to lift the vehicle. A torque curve was unavailable from the manufacturer, so it was chosen that the quadrotor should hover at 45% duty cycle. A calculated 1800g of thrust gives a target weight of approximately 4 lbs. The sum of component weights is estimated with Autodesk Inventor.

Data Flow

It is desired to keep the control of the vehicle and the intelligence algorithms separate. This allows for a failsafe in the event of a software problem to allow for manual control of the quadrotor. This creates the following diagram.

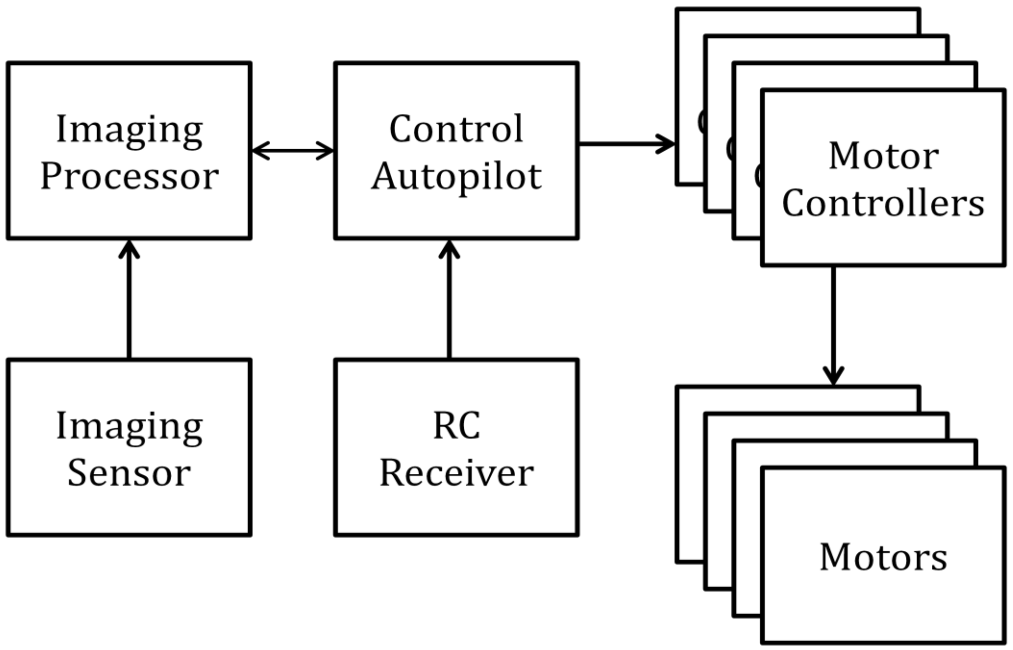


Figure 80: Data Flow Diagram

The image processor collects data from the imaging sensor. In this case, a Logitech C310 USB webcam is used as the sensor. The webcam is rigidly mounted under the quadrotor to give visual feedback.

The autopilot keeps track of the quadrotor's orientation. Notice the bidirectional flow between the image processor and the autopilot. This allows the image processor to access the sensor data from the autopilot when needed, while not having to calculate the orientation.

A software diagram showing the software implementation layout is shown below. Image detection software combined with a MAVlink communication protocol sends command and control instructions to the Arducopter software on the APM. The APM then gives feedback to the Pandaboard on the current state of the quadcopter.

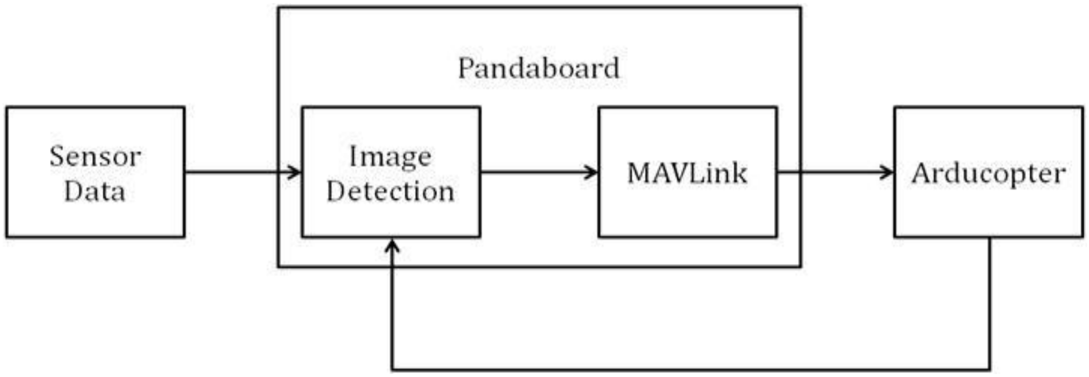


Figure 81: Software Block Diagram

Airframe

Design

The airframe is designed to be lightweight, agile, and able to carry the necessary payload. The design consists of several plates stacked upon each other with four booms supporting the motors. The primary CAD tool used was Autodesk Inventor. Manufacturing was accomplished on a Hurco VM1 CNC mill (Hurco CNC Machine Tools 2012).

A major factor in determining the agility of a quadrotor is its overall diameter (Michael, et al. 2010). When high agility is needed, a smaller diameter is used and overall stability is reduced. The result is higher positioning error and a higher propensity for horizontal drift. This is not a high agility application due to the need for stable camera images. The overall diameter of the quadrotor is 20.75 in., which provides a balance between agility and stability.

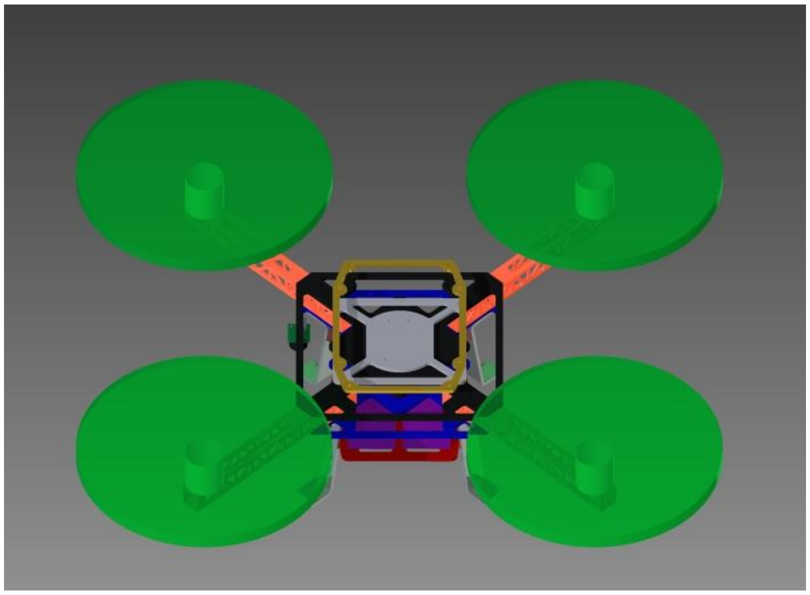


Figure 82: CAD Model Assembly

The electronics were designed to be a cradle for easy access and removal. The image processor sits on top and connects to the top frame plate. The autopilot and sensor package sits below the image processor and does not connect to the frame directly. This design isolates the gyro and accelerometer from motor vibrations.

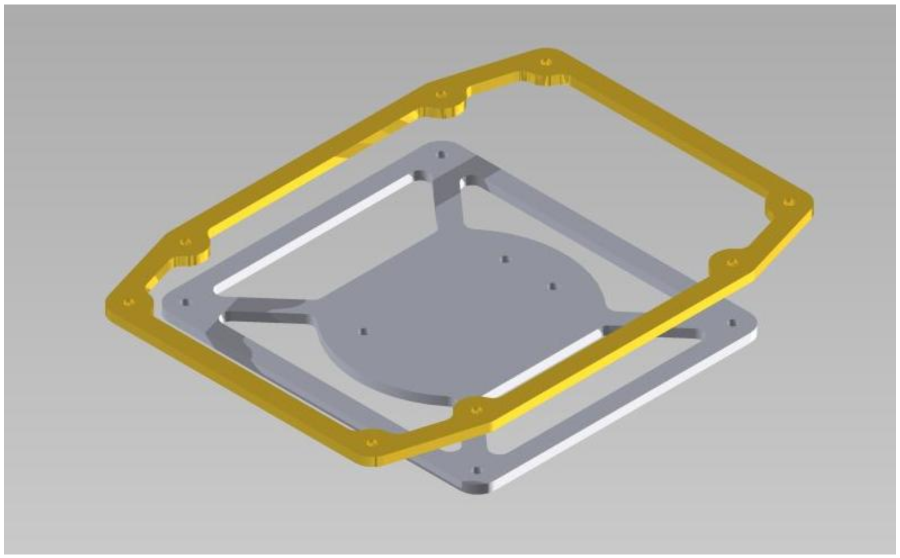


Figure 83: Electronics Cradle

Two 5Ah batteries sit below the frame plates to keep the CG below the motors for stability. A lower plate is mounted below the bottom frame plate to support the batteries and carries the mount for the camera.

Four Turnigy Parkfly 480 brushless motors are used. The motors mount directly to the square aluminum booms. The motors are fastened with screwed and secured with Loctite. The booms are lightened to reduce weight while still maintaining their desired strength to support the payload and weight of the quadrotor. Rivets fasten the booms to the frame plates.

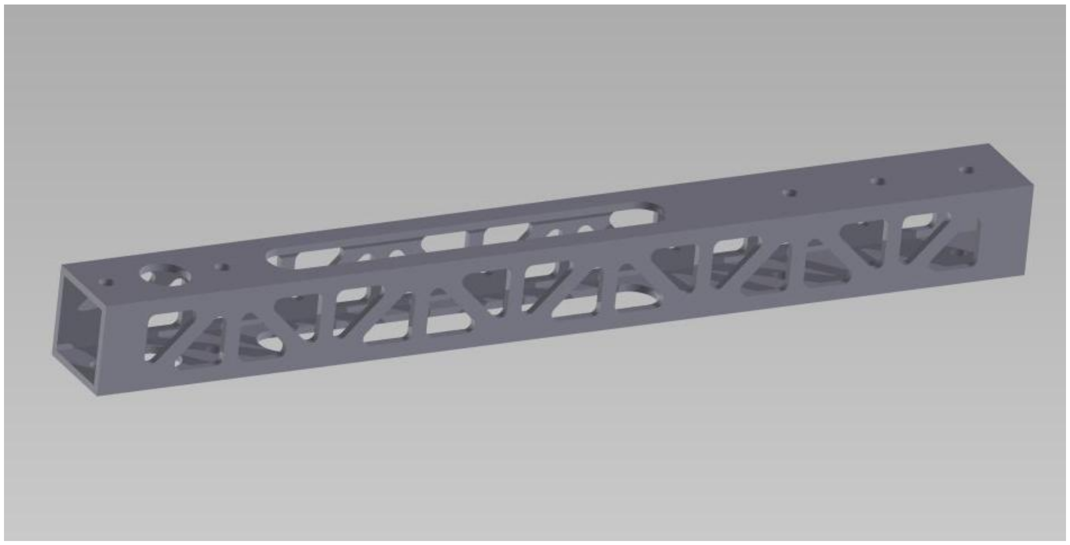


Figure 84: Square Boom

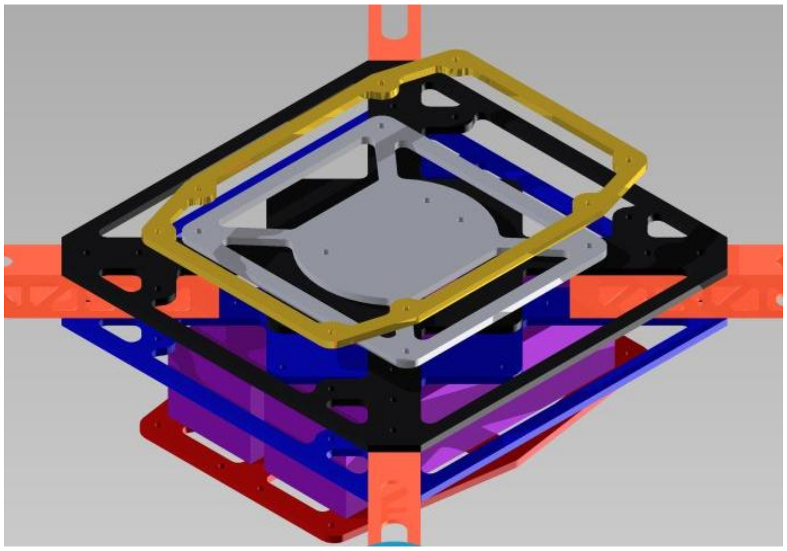


Figure 85: Frame plates and Electronics Cradle

The primary fasteners used are blind rivets and #4 button head screws. The booms are riveted between two frame plates to create a strong frame. Hex standoffs are used to mount the additional plates to accommodate the batteries and the electronics cradle.

Manufacturing

3D CAD models are exported from Inventor and used to create the realization of the model. The major materials used are 1/8" thick ABS plastic for the plates and 1/16 in. thick by 3/4 in. square aluminum tubing for the booms. Plastic is used because of its low cost when purchased in sheets and its high machinability compared to composite materials. The total system weighs 1.9kg.

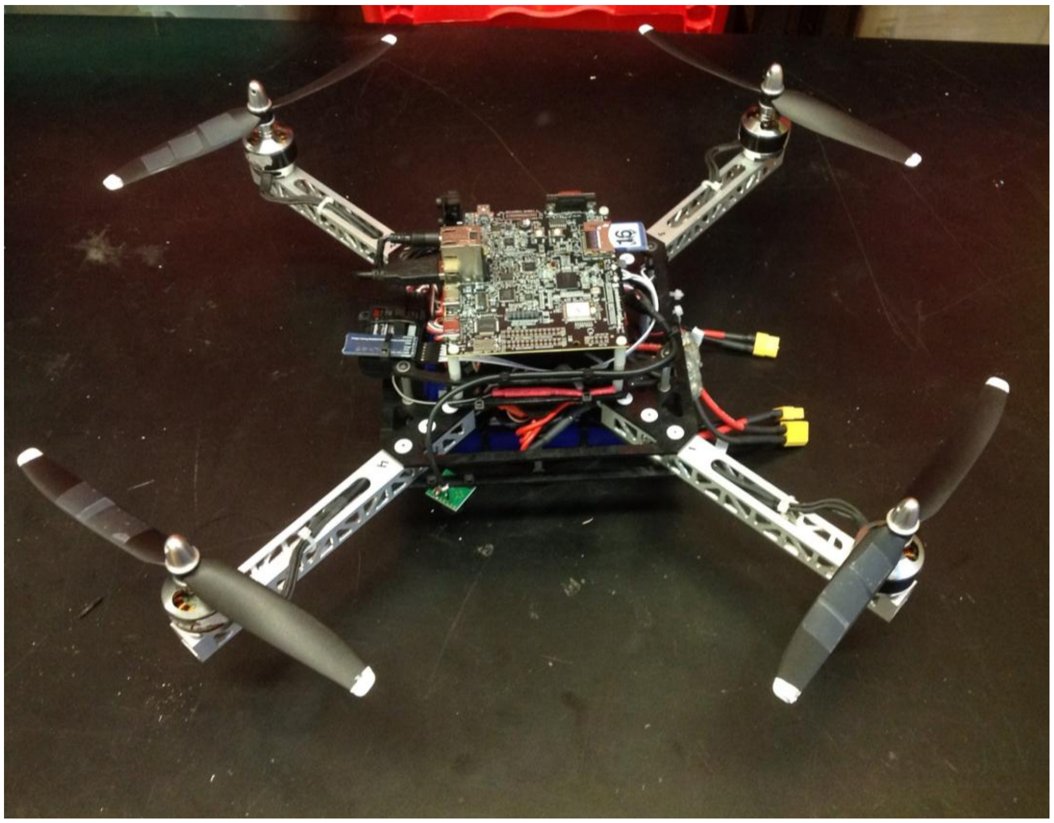


Figure 86: Quadrotor Airframe and Electronics

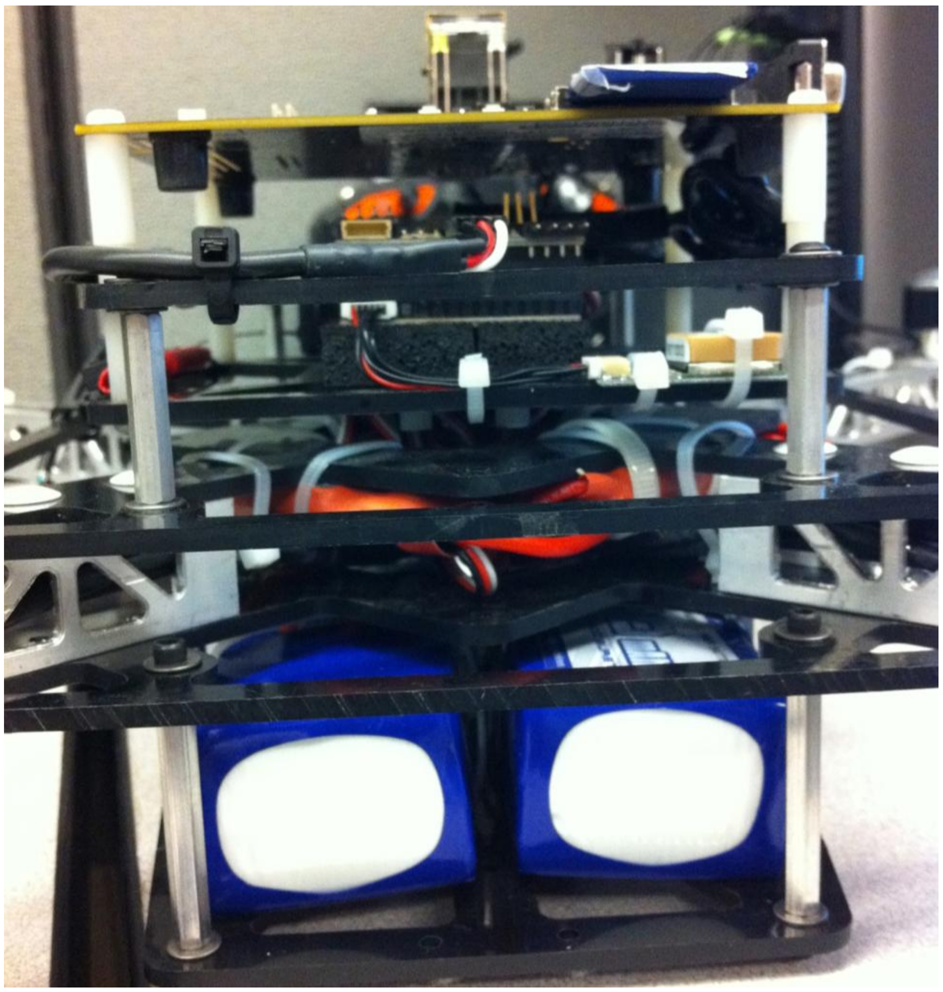


Figure 87: Quadrotor Electronics Side View

A webcam mount was 3D printed from ABS plastic to keep the camera pointed down 30 degrees and aligned with the center axis of the quadrotor.

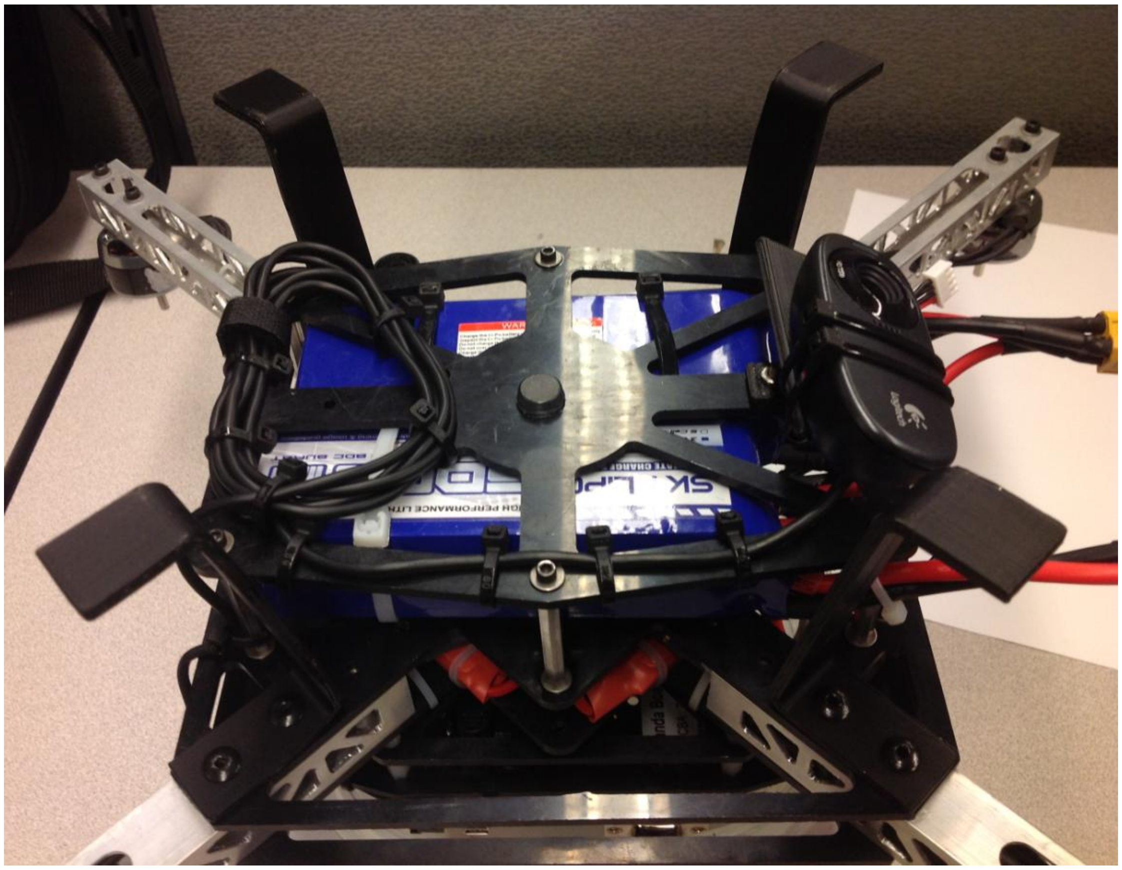


Figure 88: Landing Gear and Webcam Mount

Landing struts were also 3D printed in order to provide support under landing. It was learned through flight tests that there is commonly a horizontal velocity associated when landing. Landing struts with high friction caught debris when moving close to the ground and could cause the quadrotor to flip over. The more robust current landing gear have an inherit spring factor which causes a rebound when impacted with the ground to prevent inversions.

Image Processor

A core focus of this project is to keep all data processing onboard, so an RF data link is not required for unprocessed data. This is a divergence from previous autonomous research (Michael, et al. 2010). This on board processing focus will allow for greater expansion of capability later.

A Pandaboard ES is used to process images from a camera, analyze and generate a trajectory if necessary, and communicate with the autopilot. It is used as a platform for mobile software development and contains a Dual-core ARM Cortex A9 processor operating at 1.2 GHz. It was chosen for its low power consumption, ability to run Linux, and its small size. Measuring only 4 in. by 4.5 in. makes this ideally suited for the small size of the quad rotor.

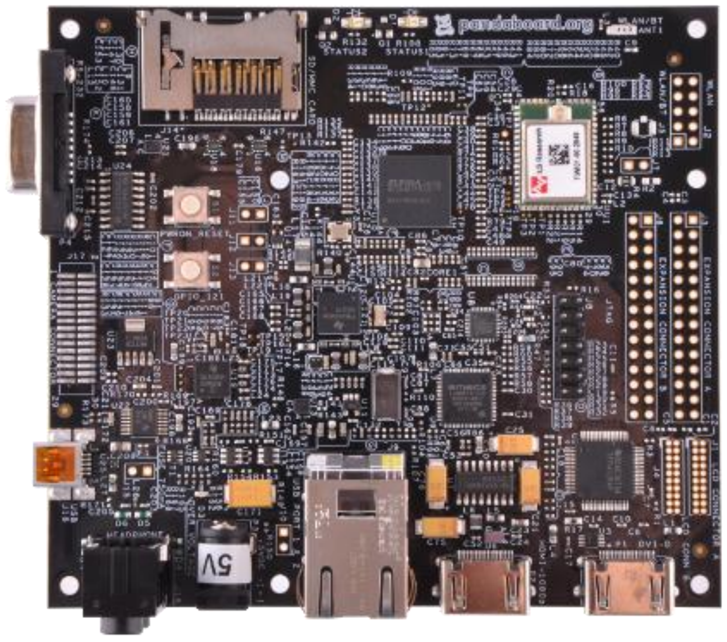


Figure 89: Pandaboard ES

Autopilot

The Ardupilot Mega (APM) is a full-featured IMU autopilot suited for a variety of unmanned aircraft applications (DIYdrones 2012). It interfaces to the motor controllers to stabilize the airframe in flight and execute maneuvers such as takeoff and landing. Onboard sensors include a gyroscope, an accelerometer, magnetometer, and a barometer. GPS is used for location reference when needed. An external ultrasonic sensor is used to obtain altitude readings because it is more accurate than a barometer at low altitudes.

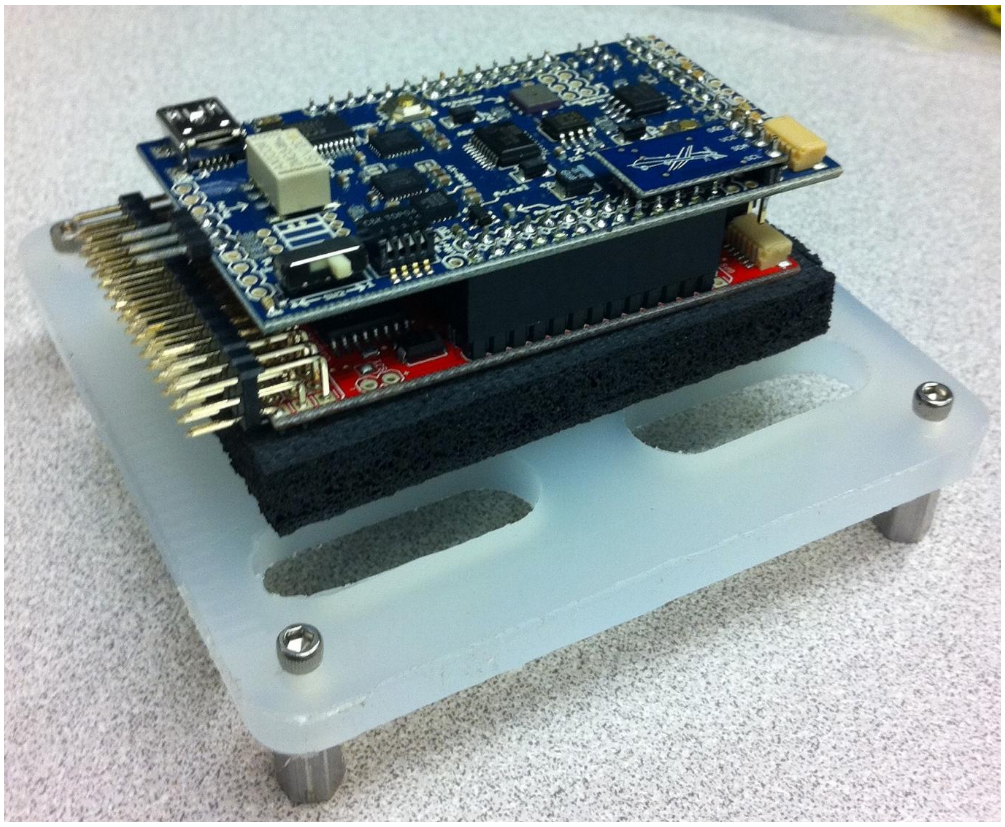


Figure 90: ArduPilot Mega 2560

ArduCopter

The software set used on this autopilot is an open source project called ArduCopter (ArduCopter 2012). ArduCopter aims to create an easy to setup and fly platform for multi-rotor UAVs. The project provides software to control the copter in flight. An interface to an RC receiver allows for manual control and override. For debug purposes, a Bluetooth link is used as a wireless serial data link to a ground station.

ArduCopter is a popular project and receives regular updates from its developers. When the APM was purchased, the version 2.0.48 was current. As of May 2012, version 2.5.5 is used.

Mission Planner

Mission Planner is another open source project that is leveraged (The Mission Planner Utility 2012). Written by Michael Oborne, it provides a GUI interface to configure the APM, monitor flight status, input waypoints, analyze log files, and change the flight modes. This software is used for functionally testing the APM and loading firmware changes. It is normally run on a Windows desktop. While there have been attempts to run the planner under other operating systems using Mono (ArdupilotMegaPlanner runs natively on ubuntu linux with the program mono 2011), the Mission Planner was unable to run on the Pandaboard under Linux.

Power Distribution

The power distribution system for this application must provide enough current for all flight critical components to achieve a successful flight. Main power comes from two 12V Lithium polymer batteries. Each battery has a capacity of 5000mAh and is rated for 200A continuous discharge. The major loads on the systems are the four motors. The maximum current draw per motor is 28A when stalled. Four stalled motors produce a load of 112A that is easily handled by a single battery.

Power from the batteries is fed through a power switch to four Electronic Speed Controllers (ESCs). Each ESC can supply one motor with up to 30A at 12V continuously. Every ESC also provides 2A at 5V, which will power the autopilot and the image processor. The four ESCs together create a 5V rail capable of 8A.

The autopilot and the image processor are minor loads and consume <3% of the total power. The autopilot receives 5V power from the ESCs. The image processor connects to the 5V rail at the autopilot. While the ESCs filter out much of the high frequency noise from the motors, additional protection is added for the image processor. An LC filter between the autopilot and the image processor filters out any additional noise and stabilizes the current.

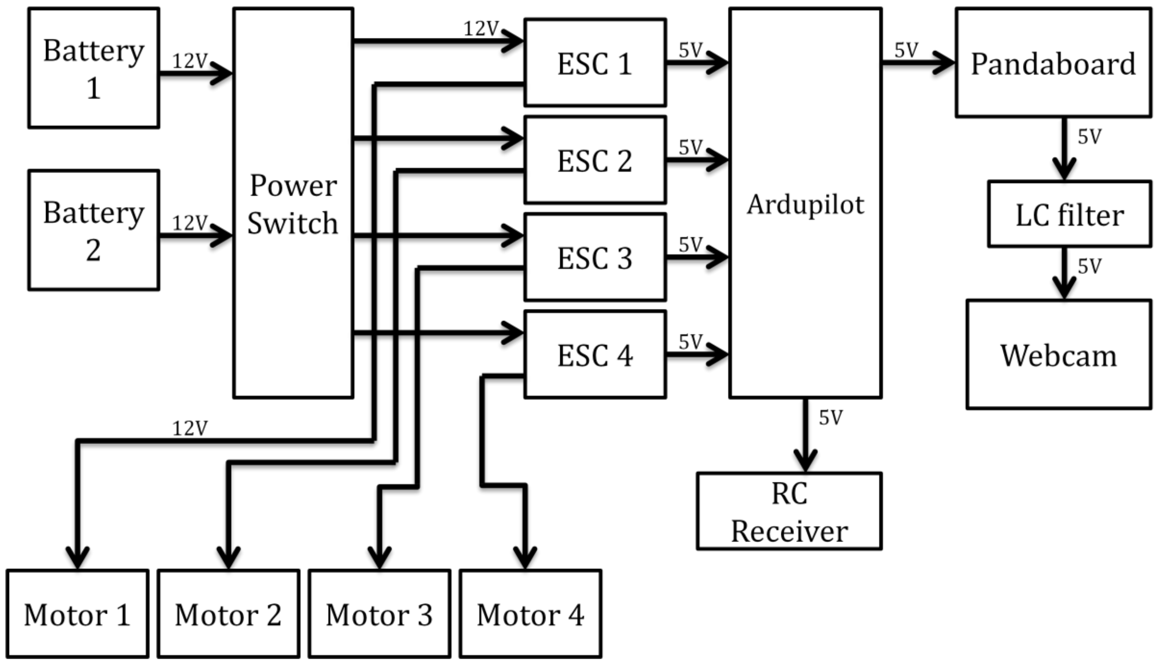


Figure 91: Power Distribution Block Diagram

Imaging Sensor

The imaging sensor is an integral part of the system. It delivers imaging feedback to the Pandaboard image processor. A variety of imaging sensors were considered, but only video sensors were purchased due to cost.

A GoPro Hero HD was tested on the first and second prototype (GoPro HERO Cameras 2012). The camera provides excellent video quality. However, the high weight of 167 grams and the inability to convert the output video stream in real-time to the linux OS caused the camera to be replaced.

A Logitech C310 Webcam was chosen for its low profile, low weight, and compatibility with linux on the pandaboard (C310 Technical Specifications 2012).

MAVLink

MAVLink is a message marshalling library for micro air vehicles (MAVLink Micro Air Vehicle Communication Protocol 2012). This library is used to provide a bidirectional interface between the image processor and the autopilot. The protocol is geared towards transmission speed and safety as it is normally used in a Ground Control Station (GCS) to MAV architecture. Python scripting is used to implement the protocol.

The packet structure is shown below. The overhead used is 8 bytes with a maximum payload of 255 bytes. The USB – serial interface provides a bandwidth of 115200 bps or approximately 56 MAVLink Packets per second.

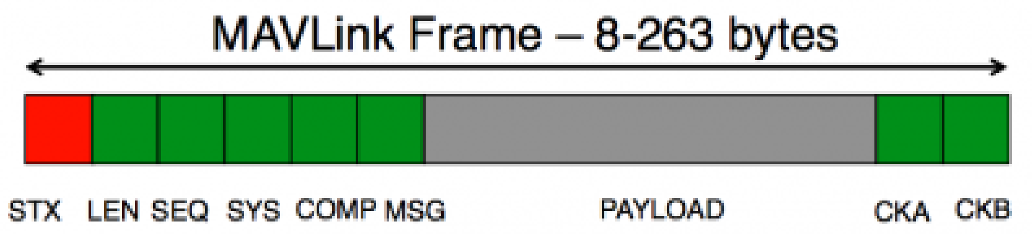


Figure 92: MAVLink Packet Structure (MAVLink Micro Air Vehicle Communication Protocol 2012)

In this application, the Pandaboard is used to send MAVLink messages to the APM all on board the quadrotor. The message headers are generated with unique message IDs.

To facilitate autonomous functions, a fork of MAVProxy was created (MAVProxy 2012). Functions for overriding RC channels were available. This allows for the development of more autonomous procedures. Functions to “arm” and “disarm” motors were written. Various functions to take off to a given altitude and then land were written to demonstrate autonomy.

While a framework has been setup, further development is needed to create a fully autonomous agent. Input for the image processing software will be processed and outputted using the presented MAVLink code.

Object Detection

Imaging data taken from the camera is processed to detect the presence of a target and its attributes. The image analysis identifies the location of the target relative to the quadrotor and creates the setpoint needed for the PID movement controllers.

OpenCV

OpenCV is library of functions for real-time computer vision (OpenCV 2012). It contains optimized algorithms used for general image processing. All code is written in C++ with OpenCV version 2.3.

Processing Operations

The basic target used for detection was a tennis ball due to its distinct shape and color. The tennis ball would be attached to a mobile target such as small robot. The software to detect and analysis this object is run under a Linux operating system. Multiple iterations of the image detection software set were made.

After opening the video stream the image is resized to 320 pixels by 240 pixels for performance. The image is masked with an upper and lower bound to isolate the object. The mask is filtered using a combination of erosion and dilation to clean up the mask. The object is then successfully isolated in an image. A sample mask is shown below where the object is placed .5 meters away from the camera.

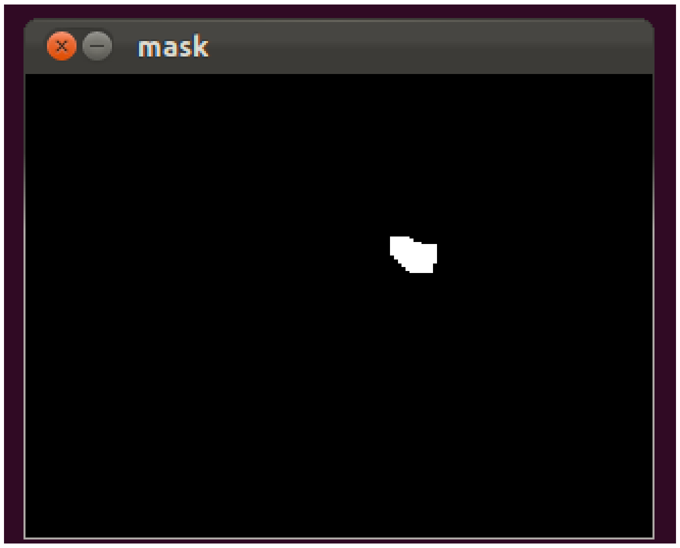


Figure 93: Mask at .5 Meters

The first motion analysis method used masking, sensing circular contours, and recording the locations in a circular buffer. The buffer is then used to calculate target movement. The block diagram is shown below with an example output.

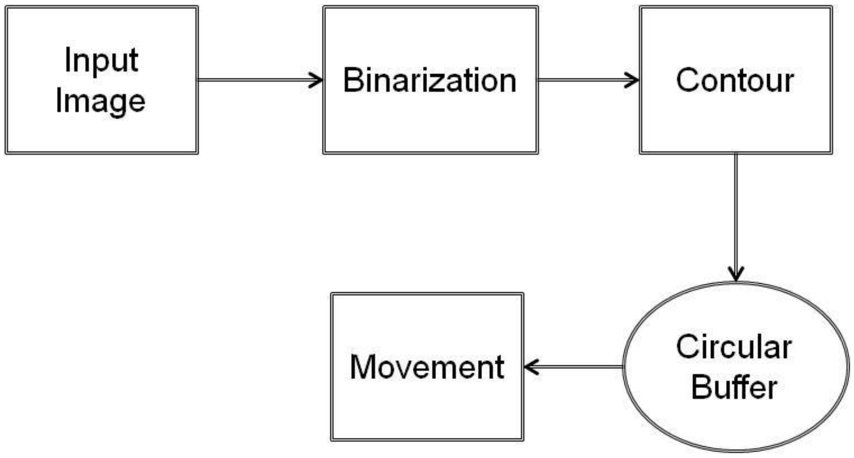
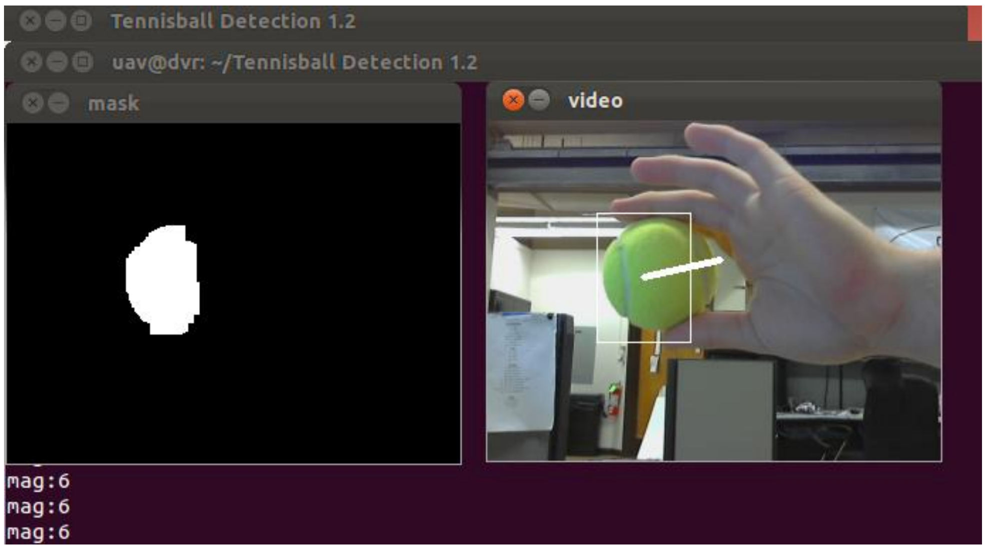


Figure 94: First Image Detection Block Diagram



```
mag:6
mag:6
mag:6
```

Figure 95: First Image Detection Output

This method suffered from low frame rates and another method was researched. To further refine the motion detection of the object, an optical flow algorithm is used. The Lucas-Kanade method is used with a sparse feature set (Motion Analysis and Object Tracking 2010).

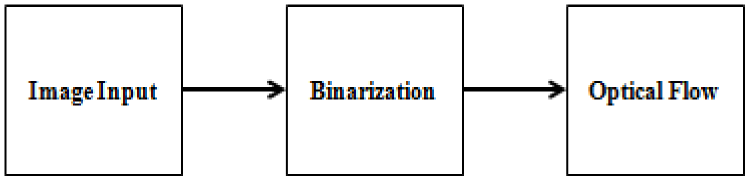
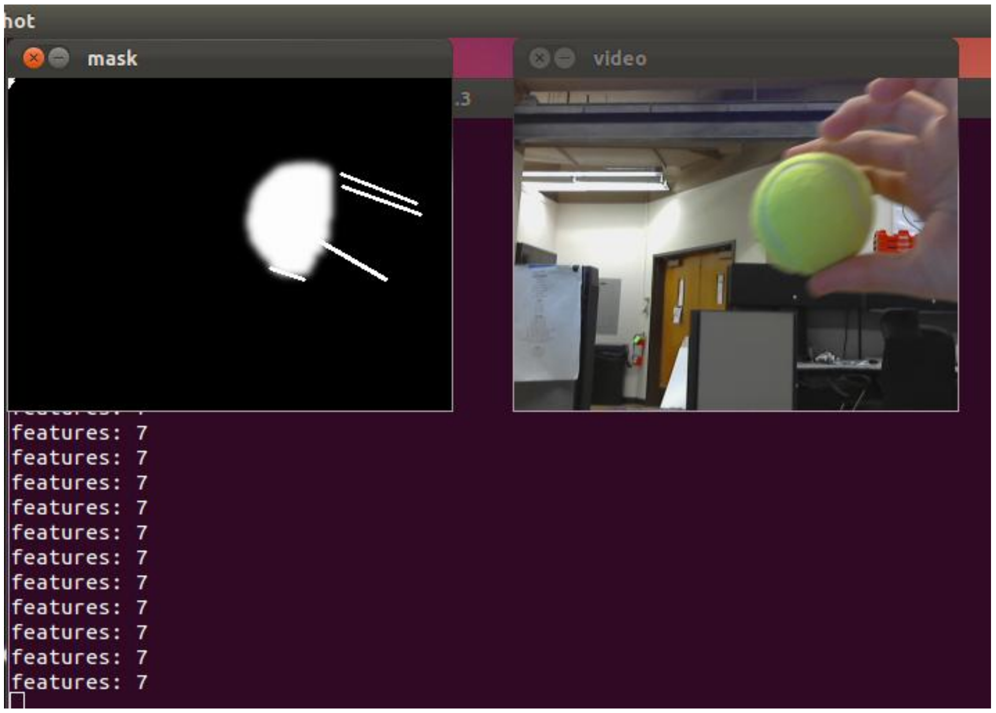


Figure 96: Optical Flow Image Analysis Block Diagram



```
features: 7
features: 7
features: 7
features: 7
features: 7
features: 7
features: 7
features: 7
features: 7
features: 7
features: 7
```

Figure 97: Image Detection with Optical Flow

Currently, little is done with the output of the image detection software. The software currently suffers from low reliability. Further analysis is needed to determine the object's movement relative to the quadrotors movement in order to create a desired setpoint for vision-PID control.

Testing

Safety

An external RC transmitter can be used to override autonomous control in the event that the quadrotor comes into close proximity of an object or person.

Flight Time

To test the flight time of the quadrotor, and tether and a weight were used to keep it close to the ground. Then under manual control, the unit was brought into a hover with all systems running to give maximum load. The quadrotor stayed aloft for 27 minutes and 40 seconds.

Manual Flight

The current prototype, as well as previous prototypes, has been flown under manual control using visual feedback to characterize the flight characteristics.

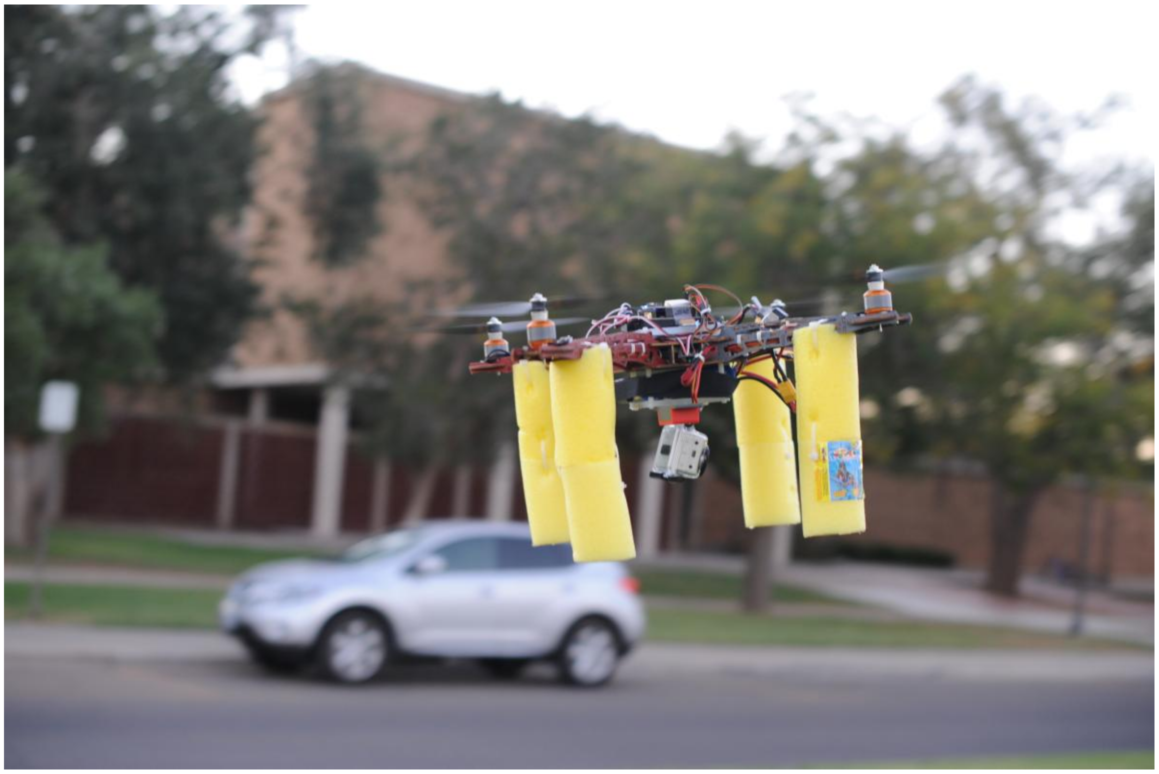


Figure 98: First Prototype Outdoor Flight 1

Figure 99: First Prototype Outdoor Flight 2

When outside, all prototypes have exhibited vulnerability to wind. Even low wind gusts have an effect upon horizontal positioning. Because of this, the platforms are not flown outdoors in high winds.

Reliability

Reliability has been an ongoing problem. The components exhibiting problems have typically been motors and propellers. There is no redundancy built into the platform, so either failure causes catastrophic failure.

Motor failures have been traced back to connectors that fail under high vibration causing intermittent signal integrity problems. These connector failures cause motor RPM to drastically decrease, causing a crash. This issue was addressed by changing connectors.

Motor shaft retaining clips have failed, causing the armature to disconnect from the motor housing. This issue was exhibited by only one model of motor and has been replaced by a higher quality motor without a shaft retaining clip. A photo of one such motor is shown below.

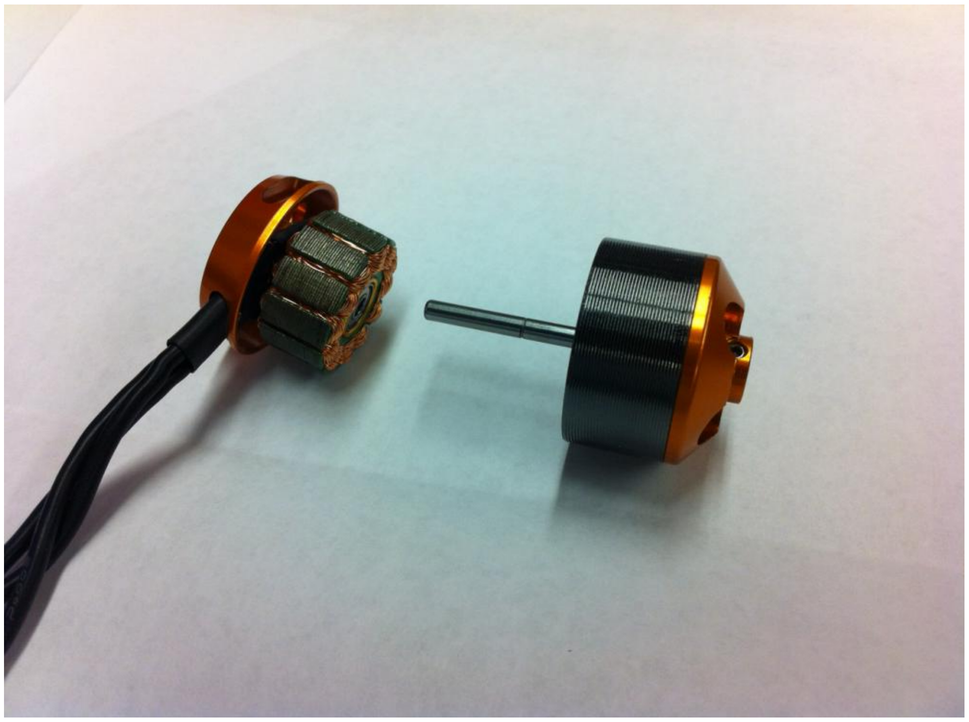


Figure 100: Failed Motor

Results and Current Status

As of now, the current model for autonomous search and tracking has been modeled and simulated successfully. The simulation has reached a level of maturity that accurately reflects the intent of the action language model. The model and simulation together provide an insight which allows development of autonomous agents of various applications.

Three prototypes were constructed and revised upon. The current prototype is capable of stable flight and provides sufficient battery life and agility to further development. It is capable of basic autonomous maneuvers from the APM and Pandaboard. Local autonomous navigation is currently not available in a GPS denied environment. While image processing software has reached some milestones such as object identification, further work is needed.

Several attempts were made to gather long term funding from various sources, but were unsuccessful. Fortunately, the low cost of the quadrotor and no proprietary software allows for development without major expenses.

Awareness of the project was generated by posting a video of some in flight footage around the department. This has spurred interest in the project and has caused several students to build a quadrotor of their own.

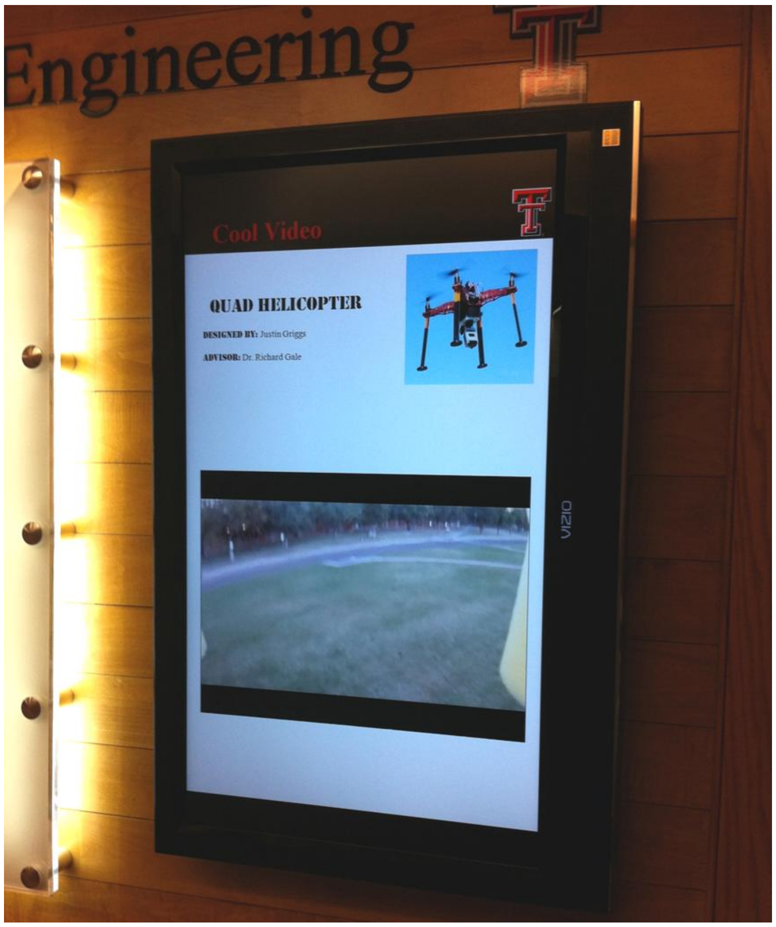


Figure 101: Department Awareness Video.

Future Work

This project represents the beginning of a large collaboration between many disciplines. There are many future possibilities and additional work to be done.

Local Navigation

One of the chief technical difficulties for multicopters is local navigation is GPS-denied environments. Additional sensor input such as an optical flow sensor can be combined with the imaging and gyro data and processed to create a stable local navigation system.

Image Processing Software

The vision based PID controller has not been fully implemented due to complexity. It is desired that additional manpower be tasked to implement the needed software sets.

Test Bed Development

Due to problems with reliability and the time spent on repairs, a controllable environment where the quadrotor can be tested is desirable. The test bed would provide hard limits to its movement and stop the quadrotor in a failure without damaging it. This would allow for repeatable tests without a lot of time spent on repairs in a crash.

Platform Agnosticism and Teamwork

It would be advantageous if the methods described were able to function independent of what platform it was implemented on. When multiple platforms are used, strategies can be tailored to take advantage of the quantity of UAVs available, known as *Swarm*.

Competition Team

Competition can push teams of people to create their best work. IARC competition with the current platform is possible and can foster teamwork and technical skills (International Aerial Robotics Competition 2012). The competition's goals are compatible with the aim and intent of this project.

Conclusion

In this paper, a model for controlling and directing actions of an autonomous quadrotor was presented and simulated. An action language model of the autonomous agent was presented. Various tracking strategies were simulated and shown through illustrations. The design of the physical system including hardware, airframe, and software has been discussed. The manufacturing of a quadrotor airframe and the associated hardware has been shown. The quadrotor's flight characteristics have been described. Problem areas have been discussed as well as available future work.

Works Cited

Aggressive Quadrotor Part II. 9 15, 2010. http://www.youtube.com/watch?v=geqip_0Vjec.

ArduCopter. 2012. http://code.google.com/p/arducopter/wiki/ArduCopter (accessed 2012).

ArdupilotMegaPlanner runs natively on ubuntu linux with the program mono. 5 16, 2011. http://diydrones.com/profiles/blogs/ardupilotmegaplanner-runs (accessed 2011).

Attaching the props to the motors. 9 8, 2011. http://code.google.com/p/arducopter/wiki/AC2_Props.

Autodesk Octo-Copters Scouring for Kenyan Trails. June 21, 2012. http://www.pddnet.com/video-autodesk-octo-copters-scouring-for-kenyan-trails-062112/.

C310 Technical Specifications. 5 15, 2012. http://logitech-en-amr.custhelp.com/app/answers/detail/a_id/17181/~/c310-technical-specifications.

Dempsey, Martin E. "Eyes of the Army - U.S. Army Roadmap for unmanned Aircraft Systems 2010-2035." *United States Army.* April 9, 2010. http://www-rucker.army.mil/usaace/uas/US%20Army%20UAS%20RoadMap%202010%202035.pdf (accessed 5 27, 2012).

DIYdrones. 2012. https://store.diydrones.com/default.asp.

Drone journalism Lab. 2011. http://www.dronejournalismlab.org/about.

Gelfond, Michael, and Yulia Kahl. "Knowledge Representation, Rasoning, and the Design of Intelligent Agents." *Michael Gelfond.* 2012. http://www.cs.ttu.edu/~mgelfond/FALL02/book.pdf.

GoPro HERO Cameras. 2012. http://gopro.com/hd-hero-cameras/.

Hoffmann, Gabriel M, Haomiao Huang, Steven L Waslander, and Claire J Tomlin. "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment." 2007. http://www.stanford.edu/~haomiao/papers/GNC07_Quadrotor.pdf.

Hurco CNC Machine Tools. 2012. http://www.hurco.com/.

International Aerial Robotics Competition. 2012. http://iarc.angel-strike.com/.

L-3 Communications - WESCAM. *L-3 Wescam Products.* 2012. http://www2.l-3com.com/wescam/products/products_services_1.asp.

Leishman, J.G. "The Breguet-Richet Quad-Rotor Helicopter of 1907." http://www.enae.umd.edu/AGRC/Aero/Breguet.pdf.

MAVLink Micro Air Vehicle Communication Protocol. 2012. http://qgroundcontrol.org/mavlink/start.

MAVProxy. 2012
https://github.com/tridge/MAVProxy

Michael, Nathan, Mellinger, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. "The GRASP Multiple Micro-UAV Test Bed Experimental Evaluation of Multirobot Aerial Control Algorithms." *IEEE ROBOTICS & AUTOMATION MAGAZINE*, 2010: 56-65.

Motion Analysis and Object Tracking. 2010. http://opencv.willowgarage.com/documentation/c/video_motion_analysis_and_object_tracking.html.

OpenCV. 2012. http://opencv.willowgarage.com/wiki/.

Pandaboard. 2012. http://pandaboard.org/.

Tacocopter. 2012. http://tacocopter.com/.

The Mission Planner Utility. 2012.
http://code.google.com/p/arducopter/wiki/AC2_Mission (accessed 2012).

Ungerleider, Neal. *Unmanned Drones Go From Afghanistan To Hollywood.* 2012 йил 15-2. http://www.fastcompany.com/1816578/unmanned-drones-go-from-afghanistan-to-hollywood.