Concept Design Report

18 Nov. 2019
Revision 1.0

Tracker for In-Flight Air Vehicle

Project Sponsor:
IMSAR

Capstone Team 15: RPS

Ira A. Fulton College of Engineering
Brigham Young University

Revision History

Revision	Date	Description
1.0	11 Nov. 2019	Initial draft, based on previous year’s template

Approval Signatures

The undersigned certify that they have read the Concept Design Report and that it is a true and accurate representation of the project.

Autumn Twitchell	Date
Daniel Sharp	Date
Garret Gang	Date
Jesse Krage	Date
Joe Hansen	Date
Nicholas Merriman	Date
Larkin Hastriter - Team Coach	Date
Mark Catanzaro - IMSAR	Date
Brian Jensen - Project Instructor	Date

Contents

Design Summary 1

1 Introduction 1

2 Selected Concepts and their Justification 1

Primary Artifacts
- CD-001 Concept Definition
- CS-001 Concept Selection Report
- RM-001 Requirements Matrix
- CT-001 Concept Testing Report

Supporting Artifacts
- CD-002 List of concepts considered
- CD-003 Concept Set Evaluation
- RV-002 Target Value Justification
- PD-001 Prototype Definition
- MD-001 Model Definition
- TP-001 Test Procedures
- CF-001 Computer Files Used
- CODE-001 Code for Tracking Simulation
- TODO Team Objectives and Development Overview

The first 2–4 pages of the Concept Design Report contain the Design Summary, whose main elements are described here. The Design Summary should not be more than 4 pages; you will have to work hard to get an excellent Design Summary, because there's not much space!

1 Introduction

IMSAR is a local company that specializes in making compact radar systems more affordable and accessible for use in small air vehicles. One of the stationary ground systems currently produced and sold by IMSAR is a computer-controlled tilt and pan positioner that maintains a communication link with radar units installed on air vehicles. The operational situation for these units is to be in a stationary position 2 to 20 miles away from the target in-flight vehicle. The current design employs a tilt and pan positioner that has become obsolete. Our team has been tasked with replacing the tilt and pan positioner, installing an onboard computer, and creating control software necessary to maintain a communication link.

Our team's objective is to design, prototype, and test a Radar Positioning System (RPS) that can track in-flight vehicles while maintaining visual contact by March 30, 2020 for under $1,500 and 1,500 man-hours. The overall performance of the system will be evaluated based on the following key success measures: time to train the user, percent increase in target acquisition time from minimum acquisition time, and the initial setup time. Ultimately the amount of time that the aircraft is within the field of view (i.e. the percent of time that a communication link is maintained) is the most important criteria. IMSAR's current system maintains a communication link 100% of the time under normal use conditions. Thus, we have determined that it is necessary for our end product to also maintain a communication link 100% of the time. Because this is a binary success or failure, it is not included in the key success measures, however, it is the most important criteria for the system. Full system requirements can be found in the requirements matrix (RM-001) attached to this document.

During the concept development stage three primary subsystems were defined i.e. the user interface, the system architecture, and the tracking method. Concepts for each of these subsystems were developed, and decisions on which concepts to pursue were made.

2 Selected Concepts and their Justification

User Interface Selected Concept

We have decided to develop a web-based user interface to setup the communication link. The user interface will allow the operator to input initialization parameters including the LLA and heading bias of the communication link, and the LLA of the aircraft or the relative position of the aircraft depending on which is known. The goal of the user interface is to be as intuitive as possible and require minimal training while still offering full functionality. See artifact XXXXX for user interface details.

Justification for User Interface Concept

Having a web-based user interface will allow the user to use their own device to setup the communication link. This will make the system easier to use and lead to a more desirable product.

An initial layout was made using PowerPoint, then taken to the user experience club at BYU for feedback. Updates to the layout were made based on the feedback received. After several iterations the layout was shown to Mark Catanzaro and Daniel Gunyan of IMSAR for more feedback. After meeting with them we were able to identify the critical features and make further updates. Based on their feedback we are confident that the user interface will be capable of meeting the requirements.

Tracking Method Selected Concept

The team has decided to use a reactionary method to track the aircraft. The aircraft will broadcast a health message including its location at a rate of about 2 Hz. When the communication link receives the LLA coordinate of the drone, the dish will move to point at that location.

Justification for Tracking Method

To justify using a reactionary method as opposed to a predictive method we wrote a Python script that simulates a vehicle in flight. For the simulation we had the drone flying a half mile away from the positioner at a speed of 60 mph. Note that this distance is closer than the nearest distance that will be used in real life. For the simulation we had the drone constantly moving with a box representing an 8°field of view updating its location at a rate of 2 Hz. This allowed us to visualize how quickly the aircraft would leave the field of view and if an update rate of 2 Hz could feasibly track it. In our testing the drone was easily maintained within the field of view. For more details on the testing see artifact XXXX.

It should also be noted that the current system produced and used by IMSAR also uses a reactionary control method. They are able to maintain a communication link using this method. Because this method has been shown to work and our own simulation supports this, we are confident that this approach will lead to a functioning and desirable end product.

System Architecture Selected Concept

We have decided to use a Raspberry Pi 3 to host a web server and run the necessary software to drive the positioner. This approach will allow the user to access the GUI from their own machine so long as they are on the same network as the Pi. Communication between the server and control software will be accomplished with a “single producer single consumer” queue. This will allow the server to write to this queue and the controller to read to the queue. This will maximize the rate of information throughput while still ensuring that the setup is thread safe. That is, the communication between the server and controller will not have any unintended interactions.

The latency for this communication setup is consistently less than 3ms, which is well under the necessary time threshold.

Justification for System Architecture

In our early meetings with IMSAR, we discussed the possibility of hosting a browser-based GUI on a Raspberry Pi. They were thrilled with this possibility. They liked the convenience of having an easily accessible GUI and the flexibility of the Raspberry Pi. Our testing showed that this design is not only plausible for accomplishing the design requirement, but exceptional. We developed a working version of the system in action to validate its credibility.

Primary Artifacts

CD-001 Concept Definition
CS-001 Concept Selection Report
RM-001 Requirements Matrix
CT-001 Concept Testing Report

Artifact ID: CT-001	Artifact Title: Concept Testing Report
Revision: 1.0	Revision Date: 13 NOV 2019
Prepared by: Autumn Twitchell	Checked by: checker
Purpose: This artifact is provided to summarize the testing of our concepts that led to our chosen concept.	

1. Revision History

Revision:	Revised by:	Checked by:	Date:
1.0	Autumn Twitchell	checker	13 NOV 2019

2. References

Artifact ID:	Revision:	Title:
N/A	N/A	N/A

3. Concept Testing

For our testing, we chose to split our system into three different components and from there we developed concepts and how to test whether they are viable concepts. Below are the results of our testing for each concept. More information on each concept can be found in the following artifacts: CD-001, …

Test Results:

- Reactive Tracking Concept: In our Python simulation, our test was successful in keeping the in-flight vehicle in the field of view at all times when we gave the system an update rate of 2 Hz. This proved to be exactly what we want to accomplish our goal of having the vehicle in the range of our antenna at all times.
- GUI Concept:
- Architecture Concept:

Supporting Artifacts

CD-002 List of concepts considered
CD-003 Concept Set Evaluation
RV-002 Target Value Justification
PD-001 Prototype Definition
MD-001 Model Definition
TP-001 Test Procedures
CF-001 Computer Files Used
CODE-001 Code for Tracking Simulation
TODO Team Objectives and Development Overview

Artifact ID: CODE-001	Artifact Title: Code for Tracking Simulation	
Revision: 1.0	Revision Date: 6 NOV 2019	
Prepared by: Autumn Twitchell		Checked by: Joe Hansen
Purpose: This code is used to simulate the tracking concept of our in-flight vehicle.		

1. Revision History

Revision:	Revised by:	Checked by:	Date:
1.0	Autumn Twitchell	Joe Hansen	6 NOV 2019

2. References

Artifact ID:	Revision:	Title:
N/A	N/A	N/A

3. These files are found in box under the 2019-2020-Capstone-15/Concept Development/Concept Development Packet/Concept_Definitions/Tracking/Controls_Simulations folder.

4. ControlsMain.py

```
'''
    controls main
    - this code runs with the DrawSystem class and ControlsParameters.py
'''
import hwcounter as counter
import time
from DrawSystem import DrawSystem
import ControlsParameters as cp
import numpy as np
from matplotlib import pyplot as plt
import matplotlib.patches as mpatches

if __name__ == '__main__':

    ds = DrawSystem()

    # initialize target and home coordinates
    target = [cp.x[0],cp.y[0]]
    home = [cp.x[0],cp.y[0]]

    t = cp.t_begin
    clock_cycles = 8e10
    # draw initial plane and fov
    ds.draw_airplane(home)
    ds.draw_fov(target)
    ds.init_flag = False
    t_temp = 0
    start_time = counter.count()
    while t < cp.t_end :
        # Add the patch to the Axes
        plt.pause(.005)
        t = t + 1
        t_temp = t_temp + 1
        # update home coordinates (tracking plane)
        home[0] = cp.x[t] + 105
        home[1] = cp.y[t] - 15

        # update plane and fov with new coordinates
        ds.draw_airplane(home)
        #12.5 is about half a second
        if t_temp > 12.5:
            t_temp = 0
            # update target coordinates (tracking target)
            target[0] = cp.x[t]
            target[1] = cp.y[t]
            ds.draw_fov(target)

    elapsed_time = counter.count_end()-start_time
    print(elapsed_time)
    plt.show()
```

5. ControlsParameters.py

```
'''
Controls Parameters

'''
import numpy as np

# values that can be changed
linspace_size = 5000    # largest value for linspace
x_center = 1395 # x-coordinate for center of airplane's path
y_center = 960  # y-coordinate for center of airplane's path
radius = 500    # radius of our current path for the airplane
t_begin = 0     # start time for the while loop in main function
t_end = 5000    # stop time for the while loop in main function
plot_width = 2790
plot_height = 1920
# parameters for shape of airplane
long_l = 100
med_l = 80
small_l = 20

# values that cannot be changed
fov_len = 127   # this is the width of the square for the field of view (fov)
center_of_rect = int(fov_len/2) # this allows us to know where the center of the fov is

# values that help to create the airplane's path
N = np.linspace(0, linspace_size, 5001) # number of points
theta = (2*N*np.pi)/linspace_size

# path of airplane
x = 2*radius*np.cos(theta) + x_center
```

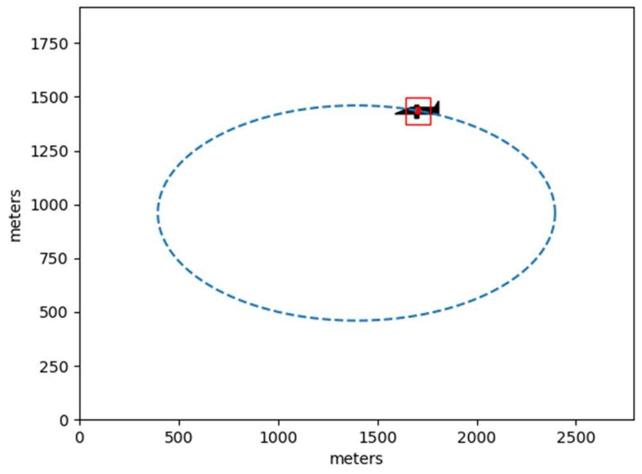
```
y = radius*np.sin(theta) + y_center
```

6. DrawSystem.py

```
'''
    DrawSystem class
    - draws airplane, fov square, and point in the middle of the fov square
'''

from matplotlib import pyplot as plt
import numpy as np
import matplotlib.patches as mpatches
import ControlsParameters as cp

class DrawSystem :

    def __init__(self):
        self.init_flag = True
        self.airplane = 0
        self.rect = 0
        self.circle = 0
        self.fig, self.ax = plt.subplots() # PLOT

        plt.axis([0, cp.plot_width, 0, cp.plot_height])
        plt.plot(cp.x,cp.y,'--')

    def draw_airplane(self, home):
        pts =np.matrix([
            [home[0],home[1]],
            [home[0]-cp.long_l,home[1]],
            [home[0]-cp.long_l,home[1]-cp.small_l],
            [home[0]-cp.long_l-cp.small_l,home[1]-cp.small_l],
            [home[0]-cp.long_l-cp.small_l,home[1]],
            [home[0]-cp.long_l-cp.small_l-cp.long_l,home[1]],
            [home[0]-cp.long_l-cp.med_l+cp.small_l*2,home[1]+(cp.small_l*1.5)],
            [home[0]-cp.long_l-cp.small_l,home[1]+(cp.small_l*1.5)],
            [home[0]-cp.long_l-cp.small_l,home[1]+(cp.small_l*2)],
            [home[0]-cp.long_l,home[1]+(cp.small_l*2)],
            [home[0]-cp.long_l,home[1]+(cp.small_l*1.5)],
            [home[0]-(cp.small_l),home[1]+(cp.small_l*1.5)],
            [home[0],home[1]+(cp.small_l*3)]]).T

        xy = np.array(pts.T)

        if self.init_flag :
            self.airplane = mpatches.Polygon(xy, facecolor = 'black', edgecolor = 'black')
            self.ax.add_patch(self.airplane)
        else :
            self.airplane.set_xy(xy)

    def draw_fov(self, target) :
        rectangle_start_point = [target[0] - cp.center_of_rect, target[1] - cp.center_of_rect]

        if self.init_flag :
            self.rect = mpatches.Rectangle(rectangle_start_point, cp.fov_len,
cp.fov_len,linewidth=1,edgecolor='r',facecolor='none')
            self.circle = mpatches.CirclePolygon(target,radius=15, color='r')
            self.ax.add_patch(self.circle)
            self.ax.add_patch(self.rect)
        else :
            self.rect.set_xy(rectangle_start_point)
            self.circle._xy = target
```

7. Simulation Figure

The figure below is an example of the output when we run ControlsMain.py. We are simulating the movement of the plane at a distance of 1/2 a mile away, flying at 64 miles per hour. We then have the red square and red point acting as the field of view for the radar positioning system, which we update at a rate of 2 Hz.



Team Objective and Development Overview

15 Nov 2019
Revision 2.0

Tracker for In-Flight Air Vehicle

Project Sponsor:
IMSAR

Capstone Team 15: RPS

Ira A. Fulton College of Engineering
Brigham Young University

Revision History

Revision	Date	Description
1.0	11 Sept 2019	Initial Release
1.1	18 Sept 2019	First Draft for Review
1.2	30 Sept 2019	Second Draft, Updated based on feedback from Design Review
1.3	1 Oct 2019	Updated based on feedback from Brian Jensen
1.4	2 Oct 2019	Fixed Typos, and made major updates to requirements matrix, rewrote key success measures based on feedback
1.5	3 OCT 2019	Rewrote Key Success Measures based on a phone meeting with Mark
2.0	15 NOV 2019	Draft for Concept Development; changed date of Concept Development in Project Approval Matrix; updated Project Background to reflect feedback from OD stage; updated Key Success Measures

Approval Signatures

The undersigned certify that they have read the stage approval package and approve of the requirements and key success measures contained in it.

Autumn Twitchell	Date
Daniel Sharp	Date
Garret Gang	Date
Jesse Krage	Date
Joe Hansen	Date
Nicholas Merriman	Date
Larkin Hastriter - Team Coach	Date
Mark Catanzaro - IMSAR	Date
Brian Jensen - Project Instructor	Date

Contents

1 Contact Information 1

2 Project Background 2

3 Project Objective Statement 3

4 Project Approval Matrix 3

5 Key Success Measures 4

6 Summary of Requirement Validation 4

7 Change Management Procedure 5

1 Contact Information

Name	Title	Cell Phone	Email
Autumn Twitchell	EE Student	801-388-5666	atwitch23@gmail.com
Daniel Sharp	EE Student	801-888-6053	danielsharp.a@gmail.com
Garret Gang	CE Student	520-333-0168	garretgang@gmail.com
Jesse Krage	EE Student	619-878-4245	jessekrage5@gmail.com
Joe Hansen	ME Student	208-850-7781	1joehansen@gmail.com
Nicholas Merriman	ME Student	385-240-3759	nicholas.merriman95@gmail.com
Larkin Hastriter	Team Coach	937-979-3481	larkinhastriter@yahoo.com

IMSAR Contact Information

Name	Title	Office Phone	Email
Daniel Gunyan	VP of Engineering	801-798-8440	dgunyan@imsar.com
Mark Catanzaro	Verification Engineer	801-798-8440	markc@imsar.com

2 Project Background

IMSAR is a local company based out of Springville, UT, that specializes in making compact radar systems more affordable and accessible for use in small air vehicles. Since their conception in 2004, they have fulfilled multiple contracts with the Department of Defense and have developed radar systems with applications ranging from fighting fires to detecting enemy troop movements.

One of the radar positioning systems (RPS) currently produced and sold by IMSAR is a computer-controlled tilt and pan positioner that maintains a communication link with radar units installed on air vehicles. The operational situation for these units is to be in a stationary position 2 to 20 miles away from the target in-flight vehicle. The current design employs a tilt and pan positioner that has become obsolete. The goal of this project is to replace the tilt and pan positioner, install an onboard computer, and to improve upon the current design. The current design requires that the data processing be done remotely on an external machine and has a barely passable user interface. Our final design will eliminate the need for external data processing. Challenges include integrating a new onboard control computer and proper handshaking between subsystems. The new RPS will be tested against IMSAR's drones flying in the vicinity. We will validate system functionality by tracking the drone with a camera mounted to the RPS. IMSAR will provide the positioning gimbal, and as such designing a positioner is not within the scope of our project.

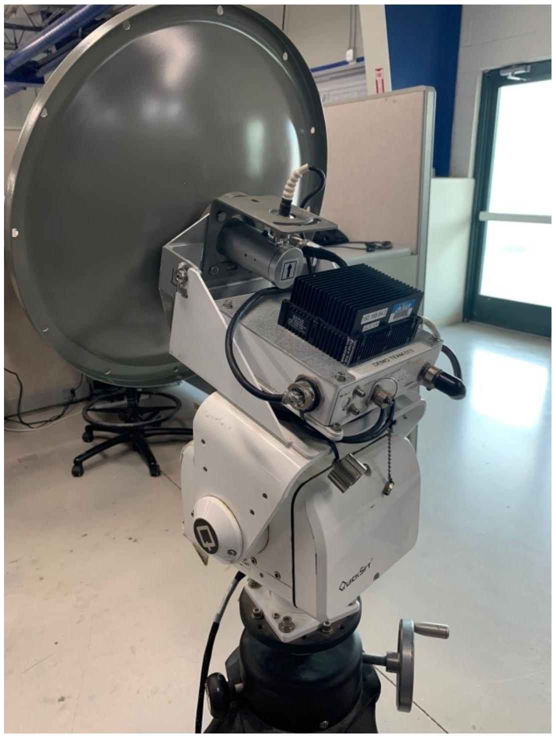


Figure 1: Current IMSAR Positioning System

3 Project Objective Statement

The team will design, prototype, and test a Radar Positioning System (RPS) that can track in-flight vehicles while maintaining visual contact by March 30th 2020 for under $1,500 USD and 1,500 man hours.

4 Project Approval Matrix

Development Stage	**Expected Completion Date**	**Artifacts Required for Approval**	**Budget**
Opportunity Development	19 Sept 2019	Team Objective and Development Overview, System Requirement Matrix with sections A-D completed, Requirement Validation	$5
Concept Development	22 Nov 2019	Written and Visual Definition of Concept, Concept Selection Report, Requirements Matrix with Target Values, Revised TODO, Concept Testing Reports	$590
Architecture Review	10 Jan 2020	System Architecture Document, System Requirements Matrix, Architecture Justification Document	$5
Subsystem Engineering	14 Feb 2020	System Design Package, System Requirements Matrix, Measured and Predicted Performance Summary	$400
System Refinement	27 Mar 2020	Written and Visual Description of Design, Performance Summary, System Requirements Matrix	$200
Final Reporting	2 Apr 2020	Fully Transferable Design, Functioning Prototype, Final Capstone Report	$300

5 Key Success Measures

Measure	Stretch Goal	Excellent Performance	Good Performance	Fair Performance	Lower Limit	Ideal	Upper Limit
Time to Train User	N/A	5 minutes	10 minutes	20 minutes	N/A	5 minutes	30 minutes
Percent Increase in Target Aquistion Time from Minimum Aquistion Time	10%	20%	30%	40%	N/A	0%	50%
Initial Setup Time	2 minutes	5 minutes	7 minutes	9 minutes	N/A	2 minutes	10 minutes

The key success measures shown above were chosen to help determine the desirability of the radar posititioning system (RPS). They will distinguish our design from a basic, functioning design. By achieving excellence in our key success measures we expect to exceed the customer's expectations. The team feels that these goals define our highest quality of work. Our key success measures account for the major flaws in IMSAR's current system. The user interface and setup time have caused the most issues for IMSAR and as such they drive our key succes measures. Reasoning for our defined measurements is given below.

The interface is currently barely usable and by running a survey and testing the interface with market representatives we intend to deliver an interface that does not require extensive training. It also takes a long time (10 minutes) to setup the RPS on site, due to the complexity of entering the data to control the RPS. We aim to reduce this by making it easier to enter the information and by improving the usage of non-volatile memory. The usage cases of IMSAR's radar units specify that every second matters when reacquiring the communication link. Mark was supportive of these key success measures and he participated in a team call in which we decided these key success measures (see NOTE-003).

6 Summary of Requirement Validation

The requirements matrix (see REQ-001 artifact) is a result of direct feedback from IMSAR's VP of Engineering, Daniel Gunyan, and Project Engineer Lead, Mark Catanzaro. In our first meeting with Daniel, we learned more about the scope of the project, and generated a rough outline of market requirements and some performance measures. After that meeting, we drafted the first iteration of our requirements matrix that we presented in person to Mark the following week. As part of this meeting with Mark, we were able to see the current system in use, and address a variety of questions (see NOTE-001). After our discussion, Mark made minor changes to our performance measures, and gave us approval for the matrix via email (see REQ-002).

Since gaining Mark's initial approval, we have made changes to both the market requirements and the performance measures for improved clarity and measurability. We have stayed in contact with Mark via phone calls and emails during the revision process. In our most recent discussion with Mark, we went over our key success measures and he approved of them (see NOTE-003).

7 Change Management Procedure

When it is determined that the TODO requires a revision, an engineering change order must be filled out (See ECO-000). The proposer will be the ECO owner. The proposer should include a description and justification of the proposed changes.

A completed copy of the ECO and TODO will be presented to the members of the team. When team feedback is implemented a completed copy will be sent to contacts at IMSAR for approval, after IMSAR's approval we will send the revision to Dr. Jensen for approval. For the proposed changes to be approved and implemented three signatures are required: The capstone team leader, a pod instructor, and a representative from IMSAR. Once the ECO is approved the proposed changes will be made to the TODO, and the revision history table filled out. If the ECO is rejected, no changes will be made.