COMPUTER VISION CONTROL FOR PHASED ARRAY BEAM STEERING

An Undergraduate Research Scholars Thesis

by

JACOB A. FREKING

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Gregory Huff

May 2018

Major: Electrical and Computer Engineering

TABLE OF CONTENTS

ABSTRACT

Computer Vision Control for Phased Array Beam Steering

Jacob A. Freking
Department of Electrical and Computer Engineering
Texas A&M University

Research Advisor: Dr. Gregory Huff
Department of Electrical and Computer Engineering
Texas A&M University

This work proves a concept for a wireless access point that uses image identification and tracking algorithms to automate the electronic control of a phased antenna array. Phased arrays change the direction of their radiation electronically by adjusting the phase of the signal applied to the individual antenna elements of the array. This ability can improve a user's connectivity to a wireless network by directing radiation from an access point to a user, provided that the user's location is known. Open source image processing and machine learning libraries provided a basis for developing a Python program that determines the position of a target using a single camera. This program uses the position information acquired from the camera to calculate the phases required to steer the radiation of the array to the target. The Python program sends the required phases to another piece of software that controls the phases of the phased array. This software adjusts the phases of the antenna elements and steers the main beam. Experiments were conducted to evaluate the identification, tracking, and control capabilities of the system. Finally, a full system demonstration was performed to benchmark the wireless performance, study the trade-offs in performance for complexity, and compare the connectivity to the current standard in multi-antenna access points.

CHAPTER I

INTRODUCTION

The rapid adoption of wireless devices has created network traffic that requires an ever-increasing bandwidth. Previously the increases in usable bandwidth have been achieved through advanced digital communication techniques that reduce the loss of information over communication channels [1]. However, these techniques are limited by the synchronization ability of the hardware used in their application [2].

Antenna arrays offer a hardware solution that can increase the gain, directivity, and bandwidth efficiency of wireless systems, making them appealing to the future of wireless networks. Current array technology can operate at millimeter wavelength frequencies, which provide much greater bandwidths than current standard communication frequencies. The benefits of arrays have been proven theoretically and experimentally [2-5]. To be useful, however, the array must be able to determine in which direction to steer its beam.

This work explores a computer vision tracking system connected to a local network access point as a solution to this issue. Recently, computer vision and machine learning algorithms have become efficient enough for real-time applications [6-7]. The work presented focuses on the development of a computer vision system for identifying and tracking target objects to provide position information to a programmable phased array in real-time. The position information is used to determine the phases necessary to apply to the signal to steer the main beam of the array to the direction of the target object. The system was tested by measuring the received signal strength at a target location while the target was both stationary and in motion.

National Instruments (NI) provided the Huff Research Group with a millimeter wave (mmWave), software controlled, phased array system capable of steering its beam in directions determined by manual input [8]. A computer vision control system was integrated to this system and tested. This system actively searched for a target object and determined the direction to the target before returning the steering instructions to the phase-controlling software. The received signal strength was measured and recorded as the target moved. This was compared to the received signal strength from a transmitted beam in a constant direction.

CHAPTER II

BACKGROUND AND APPLICATIONS

Computer Vision

Computer vision gathers, processes, and analyzes digital images and videos in a way that allows computers to extract high-level information similar to what the human brain perceives. Modern computer vision capabilities include color detection and object recognition, which can accurately find and track objects of interest.

Color Detection

Color detection relies on image segmentation algorithms, which divide images into unique regions. Common image segmentation algorithms are histogram thresholding and edge detection. Histogram thresholding allows the computer to ignore pixels with color information outside of threshold values. After a threshold is applied, edge detection finds drastic changes in the remaining color information and returns the locations of the edges [9]. Combining these methods with the hue-saturation-value (HSV) color space, which is invariant to most lighting conditions, provides a reliable tool for locating objects of a single color [10]. The HSV color space has a well-documented, nonlinear transformation from the more common red-green-blue (RGB) color space, which is the standard color space used to store images on a computer [10]. OpenCV, an open source computer vision library, provides a computationally efficient algorithm for converting images from the RGB color space to the HSV color space [11]. The library also has tools for applying thresholds based on color space values to images. One can use this filtering capability to locate specific colors in an image. The filtered image can be searched by another OpenCV algorithm to find contours between the color of interest and the background of

the image [11]. This method of object tracking is useful if the object of interest has a unique color that is separable from the background. A more reliable method for tracking uses machine learning techniques to learn how to distinguish the object from the background regardless of shared colors.

Object Recognition

Computer vision image processing techniques can extract unique features from an image and identify patterns. The patterns in images can train a support vector machine (SVM). SVMs are supervised machine learning models used for classification [7]. The simplest form of an image classifier uses unique image features, marked by the programmer, to train an SVM. Object recognition applications attempt to locate those unique patterns in images using the trained SVM. There are many patterns that could be chosen to identify objects, one of which is the histogram of oriented gradients (HOG) [6].

HOG features are extracted by sectioning the pixels of an image into cells. The pixels in each cell are assigned a gradient vector based on the difference in color values between each pixel and its neighbors. The cells are then assigned their own gradient vector based on the sum of the vectors assigned to the pixels within the cells. HOG features are especially useful when searching for objects that can vary greatly with color. The main use of HOG features is facial recognition [6].

The open source library dlib provides an efficient algorithm for training a SVM using the HOG features of training images that contain the object of interest [12]. The algorithm provided by dlib achieves 99.3% accuracy with facial recognition, however, it is not as accurate with detecting other objects. After training the SVM, one can use image pyramids and the sliding window method to locate regions of the image with matching features. The sliding window

method searches a given image one region at a time. A window of fixed width and height scans the image from left-to-right and top-to-bottom [13]. Image pyramids allow matching features at any scale to be detected [14]. This scaling is performed by gradually increasing the size of the sliding window. The region of the image inside each window is resized to have the same pixel dimensions as the training images. The SVM is applied to each resized window to locate an object of interest. Once a match is found, the pixel coordinates of the match are output. By doing this, one can track a target in real-time in a computationally efficient way, simple enough to be done with central processing units (CPUs).

The state-of-the-art object detection algorithm is called "you only look once" (YOLO). This algorithm can process 45 frames per second and achieves an accuracy of 65.5%, which is the highest of the real-time object detection algorithms available [15]. The downside of this method is the cost of the equipment necessary to implement it. YOLO recognition requires each image to be sectorized and passed through a large neural net, requiring thousands of matrix multiplications. A graphics processing unit (GPU) can parallelize this procedure, which is necessary to run the algorithm in real-time. GPUs are expensive pieces of equipment that are not easily incorporated into existing computer architectures. This procedure takes about 20 seconds to perform on one frame using ordinary CPUs, making it unreasonable to use for this project.

Phased Arrays

Phased Antenna Arrays

The antenna structure of choice for modern communication and radar applications is the phased array [2]. The benefits of using phased arrays instead of single antennas include increased gain, reduced side lobe level, and the ability to control the direction of radiation [16]. Phased arrays consist of a group of individual antennas that are arranged and excited in a way that

synthesizes radiation patterns that cannot be achieved with single antenna elements. The array factors that govern the radiation pattern of an array distributed in three-dimensional Euclidean space are given in equations (1) - (4), where equation (4) gives the total radiation pattern of an array of isotropic radiators.

$$AF_x(\theta,\phi) = \sum_{m=1}^{M} e^{j(m-1)(kd_x \sin(\theta)\cos(\phi)+\beta_x)} \tag{1}$$

$$AF_y(\theta,\phi) = \sum_{n=1}^{N} e^{j(n-1)\left(kd_y \sin(\theta)\cos(\phi)+\beta_y\right)} \tag{2}$$

$$AF_z(\theta,\phi) = \sum_{p=1}^{P} e^{j(p-1)(kd_z \sin(\theta)\cos(\phi)+\beta_z)} \tag{3}$$

$$AF(\theta,\phi) = AF_x AF_y AF_z \tag{4}$$

In equations (1) – (3), the β terms represent the phase adjustments to a signal along the different axes. Phase requirements for steering the main beam of an array to the desired direction angles, θ and ϕ, can be derived from these equations. The maximum of (4) occurs at the main beam. One can obtain this maximum by equating the exponents of (1) - (3) to 0. Thus, the phase adjustments necessary for steering the beam in an arbitrary direction are given by equations (5) - (7).

$$\beta_x = -kd_x \sin(\theta)\cos(\phi) \tag{5}$$

$$\beta_y = -kd_y \sin(\theta)\cos(\phi) \tag{6}$$

$$\beta_z = -kd_z \cos(\theta) \tag{7}$$

These phase terms are dependent on the spacing between the antenna elements in each direction, *d*, the wavenumber, *k*, and the desired steering direction angles, θ and ϕ. In the case of a two-dimensional array, equation (3) goes to 1 and the total array factor only depends on the *x*

and *y* array factors. If the array is made of homogenous antenna elements, the total radiation pattern of the array is simply the product of the array factor and the radiation pattern of an individual element of the array.

A feed network controls the excitations applied to the elements of the array. This network consists of phase shifters and power dividers. The radiation pattern can be steered electronically to a desired direction by adjusting the amplitude and phase of the excitations applied to each element. These phase shifters can provide discrete, digital phase control or continuous, analog control. Digital phase shifting requires binary input for the state of the phase shifter. The spatial resolution of phased arrays employing digital phase shifters depends on the number of discrete states available. Analog phase shifters require an external, analog voltage to set their phase shifts. For example, the external voltage of 1.0 V could correspond to a 0.0-degree phase shift, whereas a voltage of 5.0 V could correspond to a 180.0 degree phase shift, and every voltage in between will create its own unique phase shift.

Previous work has attempted to integrate computer vision into a control structure for phased arrays [17]. Jensen et al. primarily focused on developing a system that could manipulate its radiation pattern given a specific geometry of antenna elements found by the computer vision system. This work seeks to extend the use of computer vision in phased array systems by providing a targeting application to control the direction of the radiation pattern autonomously.

Software Control

Software defined radio (SDR) provides a software implementation for the hardware typically found in phased array feed networks, which allows the feed network to be controlled directly by software. Software algorithms can provide sophisticated control of phase shifters. Equations (5) - (7) can be solved using software, and the phases can be updated either by

changing the discrete state of digital phase shifters or the analog voltage applied to an analog phase shifter. This makes tracking algorithms via software realizable if the location of the target is known. By locating the target, solving equations (5) - (7), and adjusting the phases of the phased array, the signal from the array can follow the target. This will allow network access points to direct their signal to users as they move through an area.

Applications

5G Communication

5G communications have been intensively researched recently and are predicted to become commercially available by 2020 [4]. The main improvement 5G will offer over 4G is an increase in the bandwidths of communication channels as a result of increasing the operating frequency. At mmWave frequencies, wireless signals experience more intense attenuation when they penetrate objects, making it necessary to have users in the line-of-sight of a base station. This property makes efficiently directing the radiation from base stations to users desirable. To make this possible, the base stations must find the location of users. Currently, algorithms like multiple signal correlation (MUSIC) provide a long-range solution for locating network users [18]; however, this algorithm requires measuring the phase of a received signal at each antenna element in an array before carrying out large matrix calculations to locate users. While this algorithm, or one of its variants, is the most feasible solution for locating users at long distances, there may be more efficient alternatives at short distances, especially inside buildings that the signals cannot penetrate.

CHAPTER III

DESIGN

System Hardware

The system utilizes high definition cameras for computer vision and software-controlled phased arrays. The integration of these systems will provide automated control for the phased array.

Cameras

This work explored the use of two different cameras for the computer vision system. First, Microsoft's Kinect, a depth sensing camera, was used to gather information regarding the exact 3D position of targets. The accuracy of the Kinect was poor and calibrations failed to eliminate random errors in the position information returned in the point cloud. Next, a high definition, 2D camera was tested and provided more accurate azimuth and elevation information, however, no depth information was acquired. For this application, depth information is unnecessary for determining the direction of the array's beam. The target tracking subsystem uses the 2D camera and no depth sensor because of this.

The camera and phased array were placed near each other, so the effect of their relative positions could be assumed to be negligible. This configuration was used to determine if the steering instructions given by the computer vision system were accurate. With the camera mounted on the phased array, the system's spatial coverage was limited to the field-of-view (FoV) of the camera. A comparison of the measured FoVs of the array and Logitech camera is given in Table 1.

Table 1. Comparison of field-of-views for camera and array

Field-of-View Direction	Logitech C920 HD Webcam	NI mmWave Phased Array
Azimuth	60°	120°
Elevation	40°	120°

NI mmWave Transceiver

The NI mmWave transceiver system is a software defined radio (SDR) that offers the "first real time mmWave prototyping system" for mmWave communication systems [8]. The system's software interface allows designs to be implemented quickly as both hardware and software for mmWave systems. LabVIEW software controls high-performance FPGAs that facilitate the data processing of the transmitter and receiver systems. The transceiver also includes a phased array that operates at 60 GHz using digital phase shifters. Software controls the phase shifters, allowing the main beam to be controlled by software inputs.

The radiation pattern of the NI mmWave transceiver array was modeled as a 5x2 microstrip patch antenna array. The pattern was calculated using analog phase shifts and plotted for several main lobe directions. Figure 1 shows these patterns. The radiation patterns are very wide in the elevation plane but narrow in the azimuth plane. This characteristic makes it difficult to judge the accuracy of the results in the elevation tracking and steering mode. If the array had more elements, it could provide a more narrow, focused beam. A narrower beam would require more precise tracking methods to direct the main beam accurately.

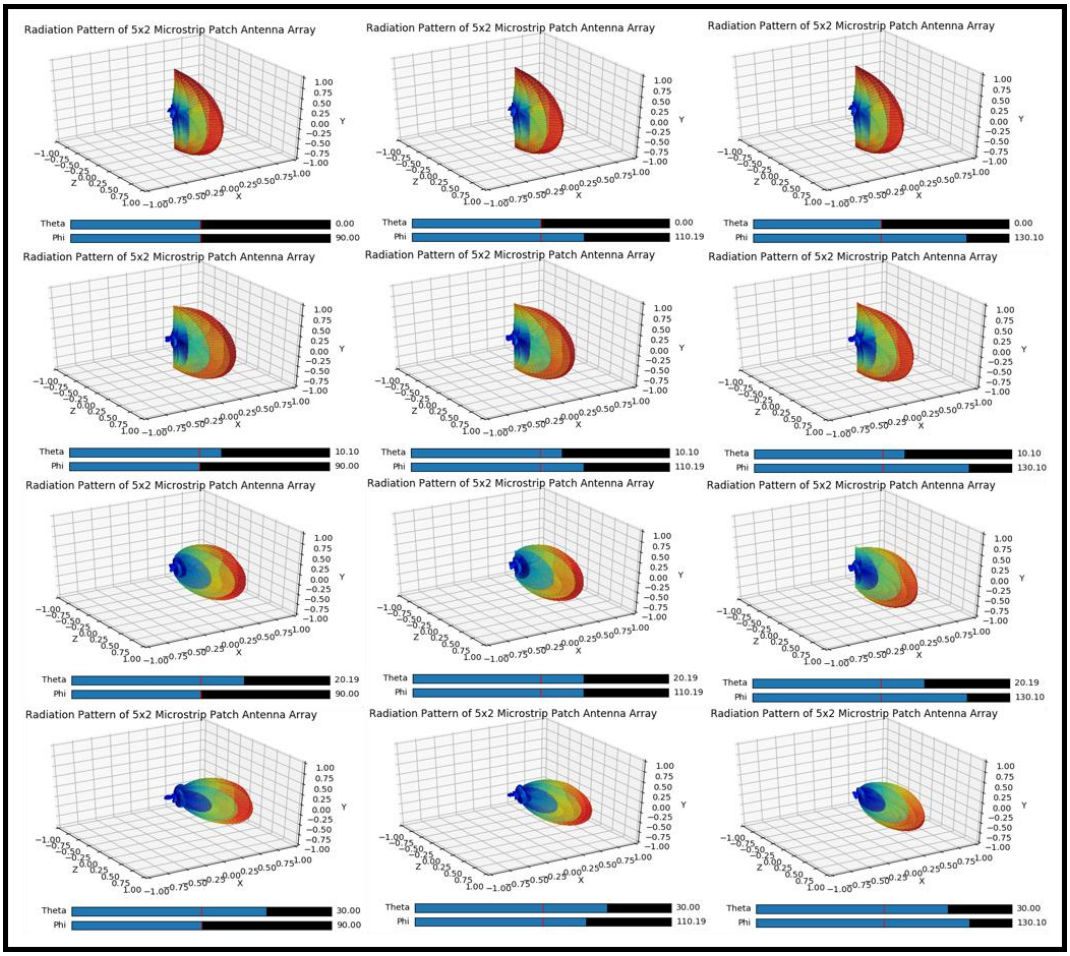

Figure 1. Approximate normalized radiation patterns of the NI mmWave at several main beam positions. The azimuth angle increases by 10° in each row, starting with 0° in the top row. The elevation angle increases by 20°, starting with 90° in the left column.

System Software

The computer vision system has two main tasks: accurately identifying target objects and providing the position of those objects. Identification was accomplished with color detection. Once a target was identified, a bounding box was drawn around it, and the location of the center pixel was returned. The center pixel was mapped to the measured FoV of the camera to extract the azimuth and elevation angles of the target.

To integrate this subsystem with the NI transceiver, a communication link was necessary. The transmitter and receiver systems already utilized TCP/IP connections through local access networks (LANs) to communicate the received signal strength (RSS) from the receiver to the transmitter. It made sense that the computer vision system could also make use of a LAN to communicate the phase information to the transmitter. A TCP/IP host was created within the Python computer vision framework. A TCP/IP client was created in the transceiver's LabVIEW interface. The final computer vision subsystem is shown in figure 2. Verifying the accuracy of this subsystem will be discussed in the next section.

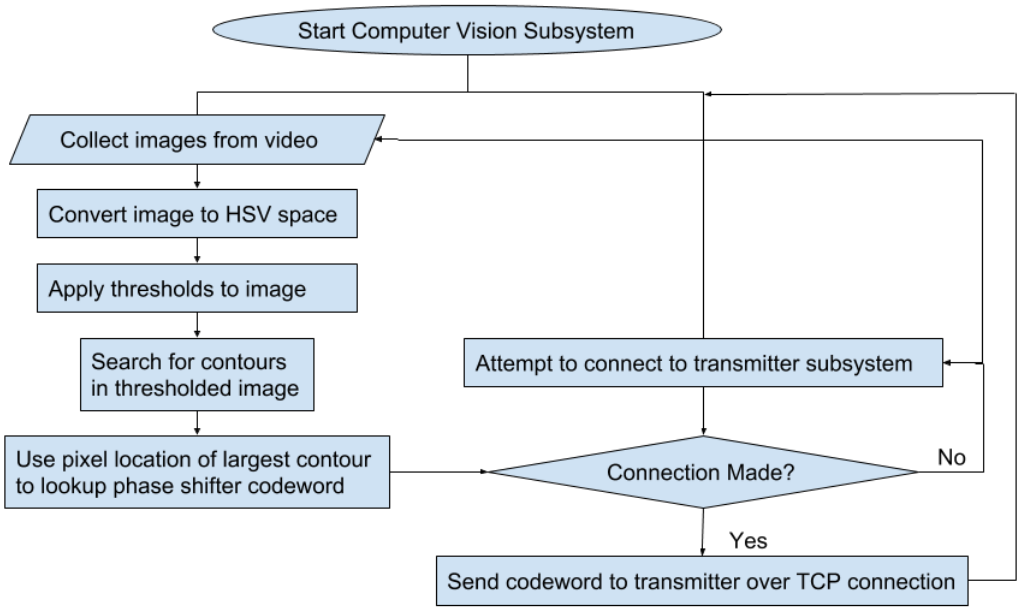

Figure 2. Flowchart for computer vision subsystem

CHAPTER IV

VALIDATION AND INTEGRATION

Subsystem Performance Testing

The accuracy of the pixel mapping algorithm was tested by placing targets in known positions and comparing the measured position to that output from the computer vision system. The computer vision measurements in the azimuth direction erred -1.4° from the physical measurements with a standard deviation of 2.1°. The azimuth error increases near the edge of the camera's FoV. In the elevation direction, the error was 0.893°, with a standard deviation of 0.596°. These errors in angle translate to average errors of -2.4cm in azimuth position and 1.6cm in elevation position at a radial distance of 1m from the camera. The position errors observed are acceptable for close-range, wide beam systems. Because the NI system has only 12 elements, it provides a relatively wide beam, so for experimental purposes, this computer vision system is sufficient. Implementing this system with phased arrays that have more antenna elements and more narrow beams should utilize more accurate computer vision systems or LIDAR (light detection and ranging) sensors to reduce this error.

A TCP/IP connection was established between the computer vision system and the NI transceiver. Python and LabVIEW documentation helped tremendously with this implementation. The NI transceiver system acts as the client and the computer vision system the host. The computer vision system continuously updates the position of the target and relays the phase information to the transceiver. The connection between host and client was tested by sending hard-coded strings over the connection and verifying that the output matched the input. The connection was verified once strings of any length could be sent and received consistently.

System Integration

Once the TCP/IP connection was verified, the integrated system performance was tested. The flowchart for the integrated system is shown in figure 3. This shows the startup procedure for the transceiver systems and how they interface with the computer vision subsystem.

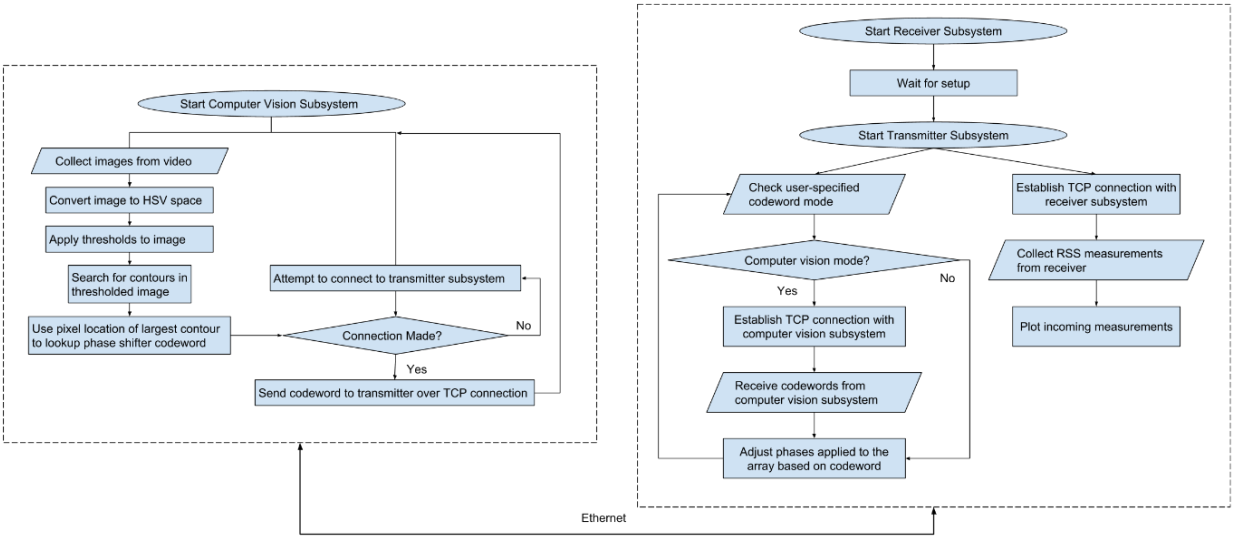

Figure 3. Flowchart for integrated system

The integrated system is shown in figures 4 and 5. Figure 4 shows the transmitting and computer vision systems. Figure 5 shows the receiver system with the target outlined by the computer vision system. The receiver was placed on a remote-controlled rover to speed up measurements. To test the system performance, the receiver system was moved and the RSS and target position were recorded at each point.

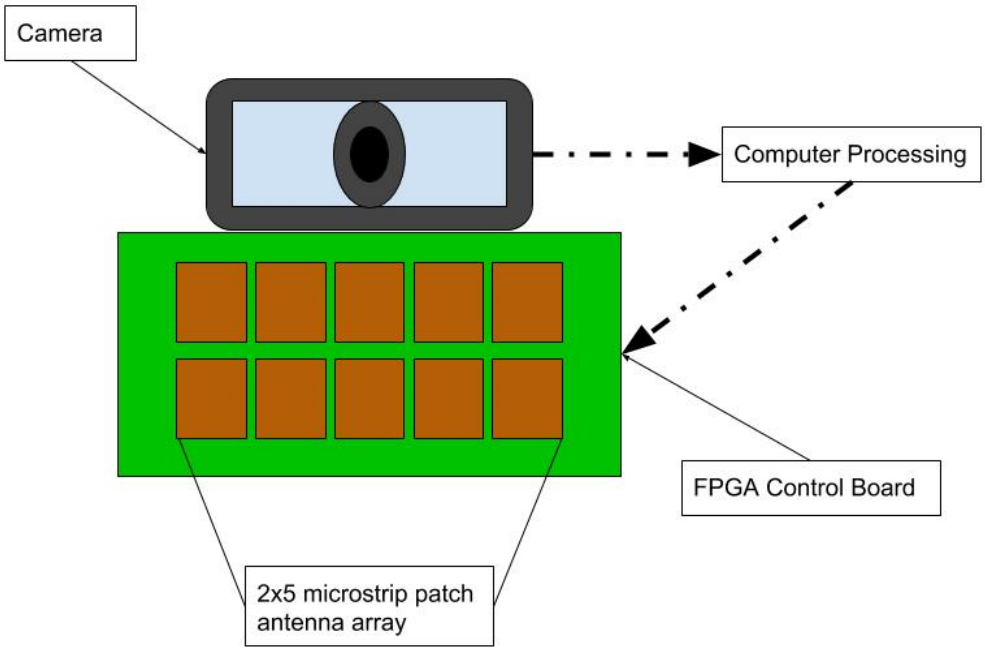

Figure 4. Transmitter and camera systems

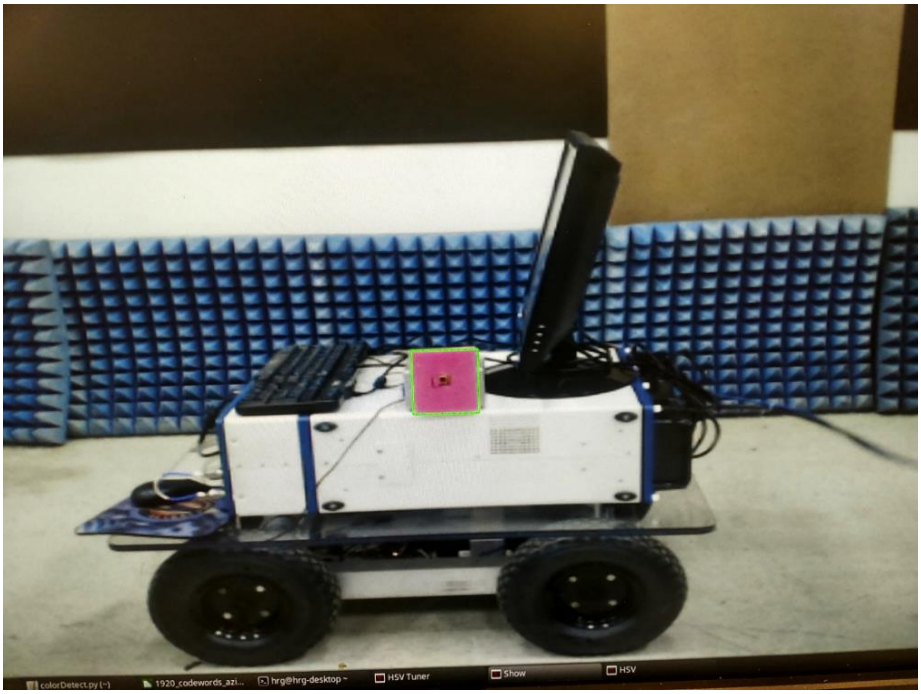

Figure 5. Receiver system and measurement platform as seen by the computer vision system

CHAPTER V

RESULTS

Azimuth Tracking

The first test verified that the system tracked a target in the azimuth plane and steered the beam with the target as it moved. The receiver system on the rover was moved through the camera's FoV, maintaining a constant distance away from the transmitting array. At several points in the FoV, the RSS and position were recorded. In addition to the computer vision-controlled beam, a beam centered at 0°, and a wide beam (equal power at all points) were measured for comparison. This test was done both with and without realigning the receiving horn. When the receiving horn was not aligned, it was kept facing parallel to the transmitter, as shown in figure 6. Figures 7 and 8 show the results of the tests with and without realignment respectively. As expected the RSS of the computer vision tracking mode in the realigned test remained nearly constant at all points in the camera's FoV. The test without realignment shows a drop in RSS due to the directivity of the antennas. In both tests, the computer vision tracking provided significantly higher signal powers than the two control cases.

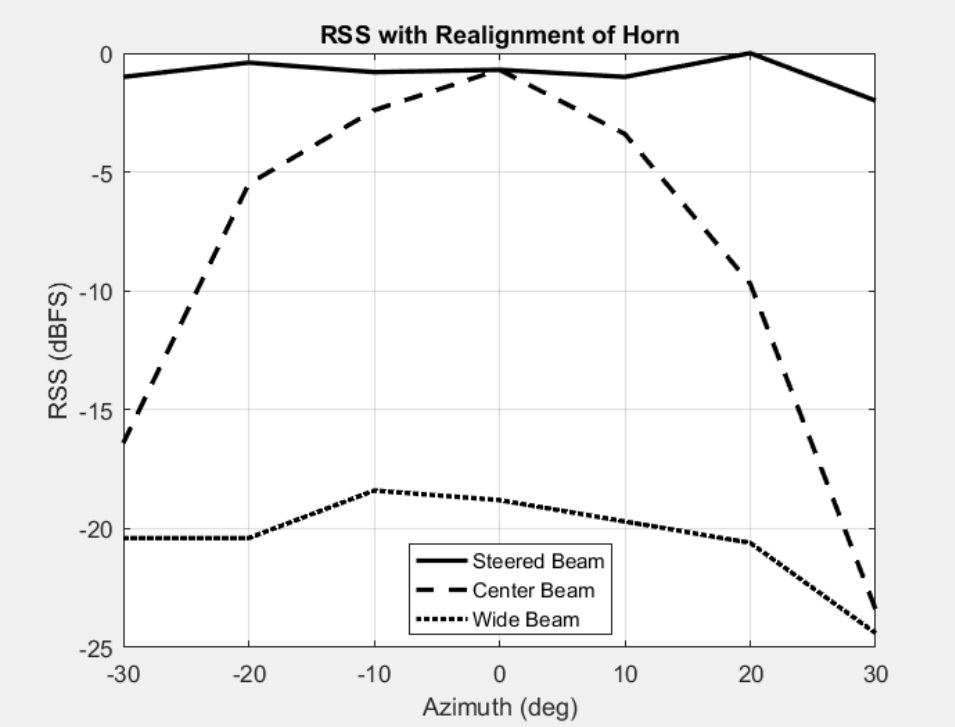

Figure 6. Normalized RSS measurements from NI system with realignment of the receiving horn

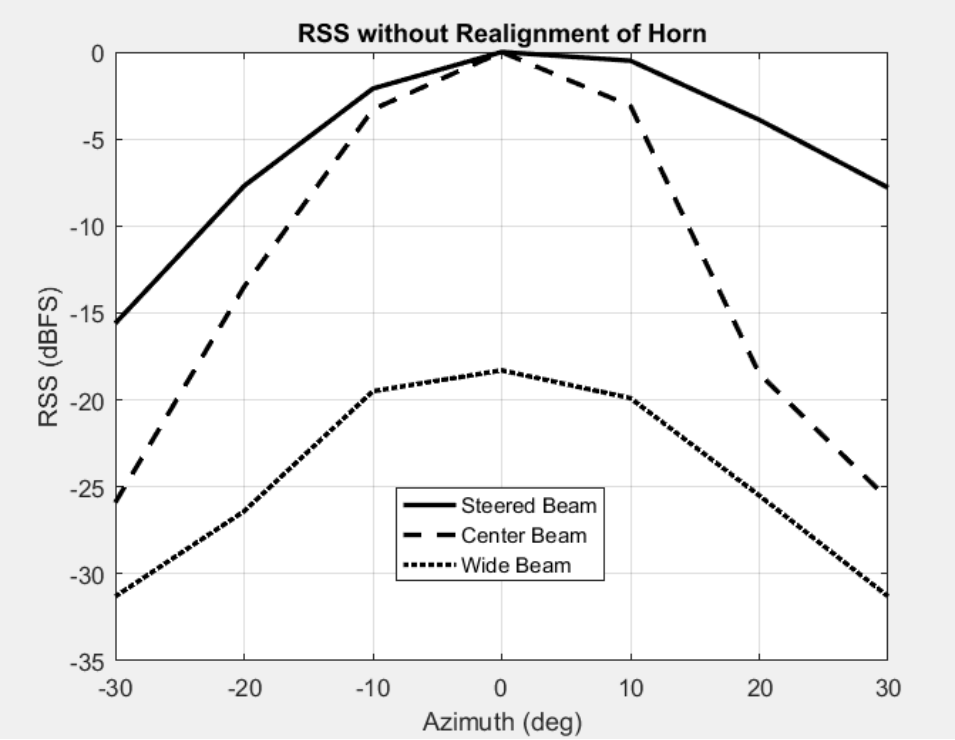

Figure 7. Normalized RSS measurements from the NI system without realigning the receiving horn

Elevation Tracking

Another test was conducted to evaluate the performance of the tracking software and the beamforming of the NI array. The measured RSS in both azimuth and elevation planes is shown in figure 9. The RSS in the tracking mode is more dependent on the azimuth direction than the elevation direction. The computer vision tracking mode consistently replicates the RSS in figure 7 for each elevation plane. RSS from the center beam is also shown for comparison. The measurements from the center beam also follow the pattern observed in figure 7.

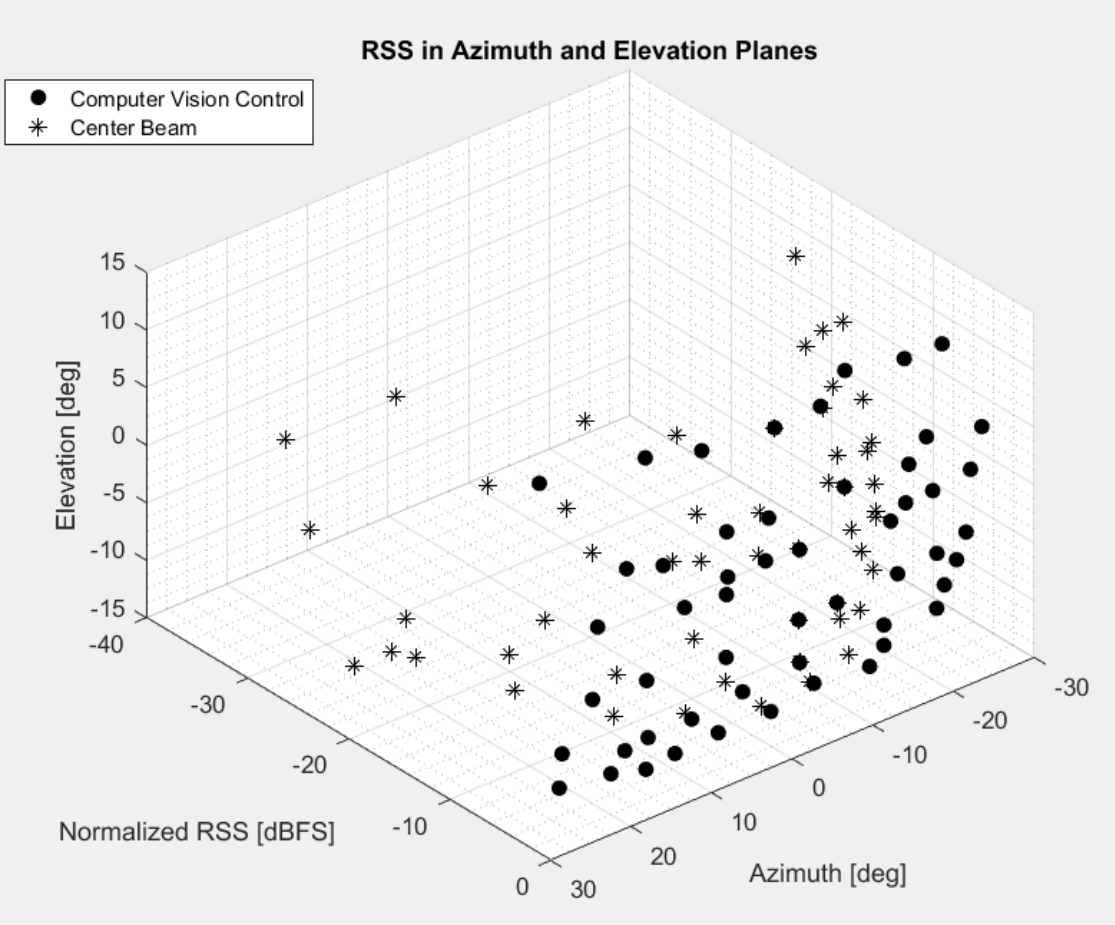

Figure 8. Normalized RSS measurements from NI system in with elevation

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

This control system uses computer vision to update a target's location and steer the main beam of a phased array to that location. This work proves the feasibility of using computer vision in the network access points of future communication systems. The functionality of the system is constrained by the FoV of the camera, which may be improved with the use of a stereo camera system. Another solution may be to place the camera(s) away from the array itself and perform geometric translations to provide relative position information to the array. In addition to improving the FoV, the object recognition system may also be improved. Current state-of-the-art object recognition algorithms, such as YOLO, require expensive GPUs to operate in real-time applications. Implementing this system with YOLO would provide a means to locate more difficult-to-detect objects, like mobile devices. Other future work also involves implementing the system with multiple arrays as access points. Such a system would allow experiments to determine how and when one access point should be used to connect to a network rather than another based on the line-of-sight to the receiver on the target. Expanding on this work could result in a working physical layer of a localized 5G network.

REFERENCES

[1] J. Rahhal, Y. Wang, and G. E. Atkin, “PSK-based constellation expansion for fading multipath channels,” in Computers and Communications, 1997. Proceedings., Second IEEE Symposium on, 1997, pp. 685–689.

[2] D. Jenn, Y. Loke, T. C. H. Matthew, Y. E. Choon, O. C. Siang, and Y. S. Yam, “Distributed Phased Arrays and Wireless Beamforming Networks,” International Journal of Distributed Sensor Networks, vol. 5, no. 4, pp. 283–302, Jul. 2009.

[3] J. Kim and A. F. Molisch, “Fast millimeter-wave beam training with receive beamforming,” Journal of Communications and Networks, vol. 16, no. 5, pp. 512–522, Oct. 2014.

[4] W. Roh et al., “Millimeter-wave beamforming as an enabling technology for 5G cellular communications: Theoretical feasibility and prototype results,” IEEE communications magazine, vol. 52, no. 2, pp. 106–113, 2014.

[5] M. H. Habebi, M. Janat, and M. R. Islam, “PHASED ARRAY ANTENNA DESIGN FOR 5G MOBILE NETWORKS,” 2006.

[6] A. Geitgey, “Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning,” Medium, 24-Jul-2016.

[7] D. K. Srivastava and L. Bhambhu, “Data Classification Using Support Vector Machine,” JATIT, vol. 12, no. 1.

[8] “Introduction to the NI mmWave Transceiver System Hardware.” National Instruments, 05-Jul-2017.

[9] P. W.-M. Tsang and W. H. Tsang, “Edge detection on object color,” in Image Processing, 1996. Proceedings., International Conference on, 1996, vol. 3, pp. 1049–1052.

[10] H. D. Cheng, X. H. Jiang, Y. Sun, and J. Wang, “Color image segmentation: advances and prospects,” Pattern Recognition, vol. 34, no. 12, pp. 2259–2281, Dec. 2001.

[11] “OpenCV 3.0.0-dev documentation,” 10-Nov-2014.

[12] D. King, “dlib Documentation.” [Online]. Available: http://dlib.net/python/. [Accessed: 23-Oct-2017].

[13] A. Rosebrock, “Sliding Windows for Object Detection with Python and OpenCV,” pyimagesearch, 23-Mar-2015.

[14] A. Rosebrock, “Image Pyramids with Python and OpenCV,” pyimagesearch, 16-Mar-2015.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016, pp. 779–788.

[16] C. A. Balanis, Antenna Theory Analysis and Design, 4th ed. John Wiley & Sons, Inc., 2016.

[17] J. S. Jensen, K. Buchanan, J.-F. Chamberland, and G. H. Huff, “A Computer Vision-Based Framework for the Synthesis and Analysis of Beamforming Behavior in Swarming Intelligent Systems,” in Radar Conference (RadarConf), 2017 IEEE, 2017, pp. 0118–0122.

[18] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, 1st ed. John Wiley & Sons, Inc., 1996.

[19] J. Ruff and G. Huff, “Update on SiBeam Collaboration with TAMU and National Instruments: A Mobile Characterization System for Millimeter-Wave Phased Arrays,” 14-Feb-2018.

APPENDIX

COMPUTER VISION CODE

```
import numpy as np
import cv2
from socket import *
import socket
import os
import sys
import pandas as pd
import netifaces as ni

# Get codewords from Excel file
codebook = pd.read_excel('/home/jfreking/Desktop/192x108_codewords.xlsx',header=0)

txCodes = codebook['Tx Codes']
#print txCodes

# Get IP address (check the available links with cmd: ifconfig -- connection may
be 'eth0' instead of 'enp3s0')
ni.ifaddresses('enp3s0')
ip = ni.ifaddresses('enp3s0')[ni.AF_INET][0]['addr']
print ip

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

# Bind the socket to the port
server_address = (ip, 8085) #server name, port number
print >>sys.stderr, 'starting up on %s port %s' % server_address
try:
    sock.bind(server_address)
except socket.error as msg:
    print 'Bind failed.Error Code: ' + str(msg[0]) + ' Message: ' + msg[1]
    sys.exit()

print 'Socket bind complete'

# If no connection is available in 2 seconds of trying to send data, raise error
sock.setblocking(1)
sock.settimeout(0.1)

# Listen for incoming connections -- accept waits for an incoming connection
sock.listen(10)
print 'Socket listening'

cap = cv2.VideoCapture(1)

def nothing(x):
    pass

# Uncomment to tune tracker to a different color
```

```
cv2.namedWindow('HSV Tuner')

cv2.createTrackbar('Hmin', 'HSV Tuner', 0, 180, nothing)
cv2.createTrackbar('Hmax', 'HSV Tuner', 0, 180, nothing)
cv2.createTrackbar('Smin', 'HSV Tuner', 0, 255, nothing)
cv2.createTrackbar('Smax', 'HSV Tuner', 0, 255, nothing)
cv2.createTrackbar('Vmin', 'HSV Tuner', 0, 255, nothing)
cv2.createTrackbar('Vmax', 'HSV Tuner', 0, 255, nothing)


while True:
    # Uncomment to tune tracker to a different color

    # Get slider positions
    hMin = cv2.getTrackbarPos('Hmin', 'HSV Tuner')
    hMax = cv2.getTrackbarPos('Hmax', 'HSV Tuner')
    sMin = cv2.getTrackbarPos('Smin', 'HSV Tuner')
    sMax = cv2.getTrackbarPos('Smax', 'HSV Tuner')
    vMin = cv2.getTrackbarPos('Vmin', 'HSV Tuner')
    vMax = cv2.getTrackbarPos('Vmax', 'HSV Tuner')


    # Set HSV thresholds
    # Uncomment to tune tracker to a different color
    lw_range = np.array([hMin,sMin,vMin])
    up_range = np.array([hMax,sMax,vMax])

    # Get frame from camera
    ret, frame = cap.read()

    # Logitech C920 has a resolution of 1920x1080
    frame = cv2.resize(frame, (1920,1080))

    # Convert frame to HSV
    hsv_img = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)

    # Define frame threshold with HSV thresholds
    frame_threshold = cv2.inRange(hsv_img, lw_range, up_range)

    # Find contours
    ret,thresh = cv2.threshold(frame_threshold, 127, 255, 0)
    _, contours, heirarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    # Find center of largest contour and produce a codeword based on position
    #   Note: codeword MUST be a string for TCP/IP communication
    if contours != []:
        areas = [cv2.contourArea(c) for c in contours]
        maxIndex = np.argmax(areas)
        cnt = contours[maxIndex]

        x,y,w,h = cv2.boundingRect(cnt)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,255,0), 2)

        cv2.rectangle(frame_threshold, (x,y), (x+w, y+h), (180,255,255), 2)

        X = x+w/2
        Y = y+h/2

        #resolution: 1920x1080
```

```
        codeword = str(txCodes[np.ceil(X/10)*np.ceil(Y/10)])

        #DEGBUGGING
        """
        print("x: {}".format(X))
        print("y: {}".format(Y))
        print(" ")
        print 'codeword: ' + codeword
        print(" ")
        """

    # Wait for a connection
    print >>sys.stderr, 'waiting for a connection'
    # Try to accept a client
    try:
        conn, client_address = sock.accept() #returns open connection btwn server
and client and the client address

        # Send codeword if there is a connection, if no connection, print error
and continue
        try:
            conn.sendall(codeword)
        except socket.error as msg:
            print 'No connection available. Error Code: ' + str(msg[0]) + ' Error
Msg: ', msg[1]
            continue

    except timeout:
        print 'caught a timeout'



    cv2.imshow("Show", frame)
    cv2.imshow("HSV", frame_threshold)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

conn.close()
cap.release()
cv2.destroyAllWindows()
```