12-2020

An Integrated Camera and Radar on-Robot System for Human Robot Collaboration

Anmol S. Modur
asm7148@rit.edu

An Integrated Camera and Radar on-Robot System for Human Robot Collaboration

by

Anmol S. Modur

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical and Microelectronic Engineering

Supervised by

Professor Dr. Ferat Sahin
Department of Electrical and Microelectronic Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
December 2020

Approved by:

Dr. Ferat Sahin, Professor
Thesis Advisor, Department Head, Department of Electrical and Microelectronic Engineering

---

Dr. Jamison Heard, Assistant Professor
Committee Member, Department of Electrical and Microelectronic Engineering

---

Prof. Carlos Barrios, Lecturer
Committee Member, Department of Electrical and Microelectronic Engineering

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

An Integrated Camera and Radar on-Robot System for Human Robot Collaboration

I, Anmol S. Modur, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Anmol S. Modur

Date

Dedication

I would like to dedicate this work to my Mom, the person who inspired me to pursue my passion ever since I was young and has supported me thoughout. Lastly I would like to dedicate this work the people in my life who have allowed me to learn from my mistakes and have let me grow as a person.

Acknowledgments

I would like to thank Dr. Ferat Sahin for giving me the wonderful opportunity to do what I love so much. Your guidance and generosity has been nothing but inspiring and motivational. At times it feels like the last several years working with you have gone by too fast. Thank you for everything!

Thank you Dr. Shitij Kumar, for introducing me to the idea that humans and robots can really work together. Without your research and amazing work, I would not have been able to find what I really enjoy doing.

Celal Savur, you are the best! Thanks for always keeping me on my toes and pushing me to be the creative person that I am. And many thanks for your unwavering hand, helping me get this done!

Sarthak and Eddie, thanks for helping me along the way, keeping me up to date with research, cool and interesting technologies in the field, and for keeping me on track!

Finally I would really like to thank D3 Engineering who were able to donate their amazing cameras and FPD-Link III Xavier interface card. And Texas Instruments for donating early radar modules for testing. Without this, my work would not have been able to progress.

For all the others not mentioned **thank you so much**.

Abstract

An Integrated Camera and Radar on-Robot System for Human Robot Collaboration

Anmol S. Modur

Supervising Professor: Dr. Ferat Sahin

The increased demand for collaborative tasks between humans and robots has caused an upsurge in newer sensor technologies to detect, locate, track, and monitor workers in a robot workspace. The challenge is to balance the accuracy, cost, and responsiveness of the system to maximize the safety of the worker. This work presents a sensor system that combines six 60GHz radar modules and six cameras to accurately track the location and speed of the workers in all 360 degrees around the robot. While the radar is tuned to identify moving targets, the cameras perform pose detection to evaluate the humans in the workspace and when fused, provide 4D pose estimates: 3D location and velocity. A custom PCB and enclosure is designed for it and it is mounted to the end-effector of a UR-10 robot. This system performs all of its computation on an Nvidia AGX Xavier for offline processing which allows it to be mounted to a mobile robot for outdoor use. Lastly, this system was evaluated for accuracy in human detection as well as accuracy in velocity measurements through numerous static and dynamic scenarios for the robot, the human, and both combined.

List of Contributions

- The creation of a radar-camera sensor system for use in human-robot collaboration. This sensor system can be implemented in industrial and commercial spaces, on mobile robots, robot arms or stand-alone.
- The creation of a image processing pipeline for easy access of 360 degree, spherical images and 360 degree radar information.
- A unique approach to calculate 3D human pose from cameras and radars for human robot collaboration.
- A dataset including data collected from cameras, radar, and a motion capture system for research in human robot collaboration

Publication Submission

- O. Adamides, **A. Modur**, S. Kumar, and F. Sahin, "A Time of Flight on-Robot Proximity Sensing System to Achieve Human Detection for Collaborative Robots," *2019 IEEE International Conference on Automation Science and Engineering*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Industry is now evolving, more than ever. With the introduction of smart and collaborative robots (cobots) in factories, problems that once were unable to be solved by robots alone, have a chance of being possible. The combination of the agile and efficient robot with the dextrous and intuitive nature of the human hand allows for productivity not seen in Industry. This is Industry 4.0. A revolutionized factory where humans and robots can work together. From small production environments to large assembly lines, worker safety is at the forefront of priorities. Maintaining the safety of the worker while placing them in close proximity to cobots and assembly lines is one of the critical challenges of evolving to Industry 4.0. However, if done correctly, assembly line based injuries will decrease saving much time and money.

The International Organization for Standardization (ISO) outlines several safety modalities between humans and robots in ISO/TS 15066:2016 [1]. One of these is speed and separation monitoring (SSM), which outlines the minimum protective separation distance (SP) between the operator and robot system, and their relative velocities to ensure safety in the workspace. These relative velocities create unchanging discrete safety-related zones around the robot which constantly move as the robot moves around. A more stringent

modality used in research is dynamic speed and separation monitoring (dSSM) which allows for S to be dynamically adjustable as the humans and robots move around the workspace. This changes the safety zones relative to the human and robot velocity and position of robot and human.

In real-world scenarios, the human is not a solid body moving throughout a workspace. Generally that worker is completing a task and is extended in a multitude of positions. Dynamically adjusting the minimum protective separation distance for dSSM must also take into consideration the location and velocities of the different parts of the human: hands, legs, head, chest and other joints. This will ensure that a dSSM implementation will keep the human as safe as possible while also allowing the least amount of separation between the robot and human for them to perform their tasks. This creates a need for a sensor technology designed for robot workspaces, that can track humans.

To track the human's position and velocity in 3D space, a variety of sensors are to be used. Cameras benefit in providing a plethora of data about the environment from scene classification, to object tracking and human detection. 3D Lidar (light detection and ranging) systems can quickly and accurately collect distance measurements and are excellent at localizing objects in 3D space. Radar (Radio detection and ranging) systems are superior at object detection and velocity estimation even with distant objects. All these sensors vary in complexity, cost, data bandwidth, size and power draw. An ideal sensor system for localizing a human fuses together several sensors that combine many of their features, maximizes performance but keeps the cost low.

In this work, an integrated camera and radar on-robot system is built for calculating the

3D position and velocity of a human in a robot's workspace for dSSM. The sensor combines six radar modules and six cameras to give a continuous 360-degree field of view for the radar and a pseudo-spherical view for the camera. This on-robot sensor system is easily mountable to any robotic arm and easily implementable to existing cobot solutions using ROS (robot operating system). This sensor system was tested for accuracy in measuring the human's position and each major joint in the human in both static and dynamic situations. Also, the precision of the sensor was tested as the robot moves throughout the workspace at different velocities. All of this was calculated and compared to OptiTrack, a highly accurate motion tracking system.

The rest of this work progresses from Chapter 2-6. Chapter 2 explores the existing work that has been done in the human-robot collaboration field as well as sensor systems that exist currently. Chapter 3 goes in-depth on the various aspects, both hardware and software, that make this sensor into reality. Chapter 4 discusses the testing that was done to understand the limits of the sensor system and Chapter 5 concludes this work. Lastly, Chapter 6 showcases future works that will advance this field of robotics.

Chapter 2

Background Literature

The current chapter will discuss current works in two areas of human robot collaboration. Section 2.1 will discuss a current modality for collaboration known as speed and separation monitoring (SSM) and Section 2.2 will discuss hardware used to implement SSM.

2.1 Understanding Speed and Separation Monitoring

Currently, there is a large amount of research happening in human-robot collaboration, especially using implementations of SSM, validating the effectiveness of SSM [2], measuring minimum distances [3], and dynamic trajectory planning using SSM [4]. There are three major types of SSM: Static, Tri-modal, and Dynamic.

Static SSM or just SSM uses a fixed minimum protective separation distance (SP) to create zones around the robot to which the robot will react if an object or human enters the zone. At the heart of the SSM, the algorithm is the condition that triggers a stop of robot motion outlined in ISO 13855 [5] that attempts to prevent any collision between human and robot. In it, a worst-case scenario is defined as 2000 mm/s which ultimately resolves an Sp of 500mm [6]. Current implementations of SSM use this value as the default which some suggest tending to be more conservative relative to the guidelines in ISO/TS 15066:2016

[1]. This static value does not take into consideration abrupt changes in the motion of the human, environmental changes [7, 4, 8] or which part of the body is within 500mm.

As SSM indicates a binary state for the robot, either stopped or normal, Tri-Modal implementations introduce a reduced mode where the velocity of the robot is decreased [8]. This additional state significantly decreases the acceleration spikes when the human enters the boundary of the stopped zone. Implementations using Tri-Modal SSM require much more precision and sensitivity as the reduced zone may not be large compared to the other two: stopped or normal. Some implementations optimize the Tri-Modal boundaries by taking into account the human or robot or both velocities. Results of these tests show that the combination of both the robot and humans' velocities are the safest [3].

Lastly, the current safest approach to SSM is Dynamic SSM. Dynamic SSM fully takes into consideration the human's position and velocity throughout and around the workspace. This allows a zone-less characterization of the human's safety where the actual position and velocity of both the human and robot will determine where and how fast the robot moves [6]. An ideal implementation modifies the robot trajectory and dynamically plans around the safety requirements to prevent a collision [9]. Although Dynamic SSM is the most computationally intensive, its balance between safety and flexibility of where the human can be, makes it the ideal candidate for implementations in the industry.

2.2 Hardware designed for SSM

To implement SSM in a robotic workspace, sensors are placed either on the robot or around the workspace to monitor any changes. Sensors placed external to the robot have minimal blind spots and can observe both the robot and humans at the same time [10]. Usually multiple sensors are required and need to be calibrated to the workspace. A benefit to using external sensors is the processing of the sensors can be done using large computational nodes with direct-wired access to the sensors. On the other hand, on-robot sensors are sensor systems designed to be mounted on the robot and view the workspace from the robot's perspective. These sensor systems are subject to blind spots, the environment the robot is in, and a limit on power and weight [3]. A benefit is on-robot sensors can be placed on mobile robots and can perform either all the computation onboard the system or wirelessly transmit the data to a grounded system. On mobile applications, power constraints limit the types of sensors that can be used and on arm robots, weight constraints do the same [11].

Depth cameras are a common technology used in modern robotics. Depth cameras (or RGBd cameras) can provide a full resolution RGB image and a lesser resolution depth pointcloud that is overlaid on top of each other. This is done by having a camera, an infrared blaster, and an infrared depth sensor all in one package. The two main depth cameras used are the Microsoft Kinect and Intel Realsense [7, 12, 10]. Much of the implementations of depth cameras are external to the robot where both the human and the robot can be tracked [12, 13]. These range from understanding human gestures to control the robot to robot teaching and human-robot collaboration [14]. To process data from the depth cameras, much computational power is needed. To work around this FPGAs (Field Programmable

Gate Arrays) can be used however the development of such systems takes much time and is specific to the sensor type used [15]. Most of the time, the computation is accelerated using a GPU.

Lidar systems are another option when it comes to understanding the workspace around a robot. Unlike depth cameras, lidar data only output depth information in pointclouds and it does so much more accurately and quickly. Because of this, implementations that use Lidar systems also contain other sensors to give a reliable description of the environment [16]. Lidar systems can scan in a 2D plane or three dimensions. In most human-robot collaboration scenarios, 3D lidars are preferred as they produce a much higher data density per frame. 3D lidar systems have grown in popularity in mobile robots over the years as their robust nature and pseudo-imperviousness to changing environments allow them to perform much better than other systems. In mobile robot scenarios, human detection and tracking are of essence rather than collaboration [17, 11]. The biggest drawback to 3D lidar systems is the steep price. 3D lidar solutions defer to using depth cameras or 3D ToF cameras instead [18].

Radar systems are an uncommon but increasingly popular option for human-robot collaboration. Radar systems can estimate the location of the human as well as its velocity toward the sensor [19, 20]. With the introduction of small package millimeter-wave radar systems, they can now be placed in tight corners. Radar systems are also heavily tuned to the application at hand [21, 20]. The many parameters, from antenna design to signal to process, allows radar systems to be used for almost any task. Antennas can be designed to be flexible; capable of wrapping around a link of a robot. This allows the system to transmit

signals from different points around the robot and receive at different points, increasing the likelihood for a human to be detected in a workspace [22]. Because the signal processing for radar systems are complex, many use a separate sensor to calibrate the radar sensor such as a lidar [23] or depth camera [24].

For implementations in human-robot collaboration, few fuse different sensors into a single system to achieve good results [23, 24]. Few are able to perform dSSM as the technology is limited in field of view or is not able to process in real time [21]. Even fewer localize the joints of the human in the 3D workspace of the robot to measure minimum distances [25, 12]. The works that do, use a combination of on-robot sensing and external sensing to identify humans and observe the workspace [26, 27]. Because of this, there is a need for a completely on-robot system that can accurately track the position and velocity of the human.

Chapter 3

Proposed Method

Chapter 3 will discuss the proposed method for identifying and tracking the human's 3D pose, the process of selecting and tuning different hardware to achieve the desired results and detail the software and the process to compute the 3D pose.

3.1 Sensor Design

To perform dSSM, a sensor system optimized to detect and track, the human's position in 3D and velocity needed to be created. From previous work, the limitations of implementing an external sensor solution and calibrating techniques detract from the feasibility of use and use case scenarios. This leads to using an on-robot sensor system that is modular, easy to install, and highly effective. The robot to be used for testing this sensor system will be Universal Robots UR-10 which is a popular collaborative robot used in the industry. This 6DOF armed robot has many advantages including the ability to pick and place objects up to 10kg in weight. By adding an on-robot sensor, one of the design requirements must be to limit its weight as much as possible.

Unlike other approaches, this sensor design will need to cover a 360-degree field-of-view that will allow it to stay stationary on the robot but be able to sense the robot's entire

workspace. A perfect sensor to use is depth-cameras, however, the options for selection are limited. The first idea was to use six Intel Realsense cameras around the links of the UR-10 Robot or in a ring on the end-effector. This was inspired by Kumar et al. [3] as they placed time of flight lidar modules in a ring like manner. This would allow for easy context analysis of the objects in the workspace with the depth information to get accurate distances to the robot. However, one issue with the Realsense cameras is they have a 86 degree azimuth FoV and 57-degree elevation FoV. With this, at least five sensors would need to be used in a horizontal position and seven sensors would need to be used vertically. This means either there are more sensors on the robot or the sensors lack Vertical FoV. Also, the amount of bandwidth each Intel Realsense uses is very high. Only two depth cameras can be used on an entire USB 3.1 bus which requires the use of a very powerful computer capable of having extra USB 3.1 busses and a GPU able to process large amounts of data. This leaves the option of developing a custom sensor system that can observe the workspace and evaluate it at the same time.

A radar system is developed to detect, locate, and measure the velocity of the human. Radar systems are perfectly suitable for 3D tracking especially and are one of the few that directly measure velocity. Modern radar systems have significantly decreased in power and size and can be accurate to a few mm. Radar systems can discern easily whether the detected object is a human or not as they can be tuned to detecting humans. There are various radar modules available that have antennas that come in all different shapes and sizes. As a radar system can be used to evaluate the workspace, a camera can be used to observe the workspace.

Cameras have been used in Industrial applications to either detect the presence of humans or scan QR codes on pallets. Image processing has come a long way over the years and more powerful algorithms can be run extracting more information in each of the frames. For human tracking, images from the camera are processed to detect the pose of the human. A pose of the human is the combination of the orientation of all the human segments, from hands and arms to legs and feet, and detecting this will give information of what task the human is performing and where the human is in the frame of the camera. With the pose data of the human, along with the radar data, one can give accurate measurements of where each segment of the human body is in relationship to the radar-camera sensor system; thus a 4D pose can be created; three dimensions for the pose and one for the velocity of the pose. When used for human robot collaboration, this additional information will help algorithms like dSSM be more accurate and should lead to a more safe workspace for the human.

As important as it is to have the best sensors for the application, without proper placement, the sensors would be useless. Looking at previous works [24, 20] sensor systems that did not have a 360-degree FoV would have large blindspots and would be susceptible to making large tracking errors. Along with this, sensors mounted directly on the robot arm had difficulties in observing the entire workspace of the robot. For this, the designed radar-camera sensor system will be mounted on the end link, before the end effector. This was chosen as the ideal location to easily mount and hot-swap the sensor and for the accessibility to view much of the workspace. A 3D model to show how the six cameras and six radar modules were places, along with the custom designed enclosure for it can be seen in Figure 3.1. The proceeding sections will explore each sensor used, the hardware that puts

it together, and the software that produces valid results.

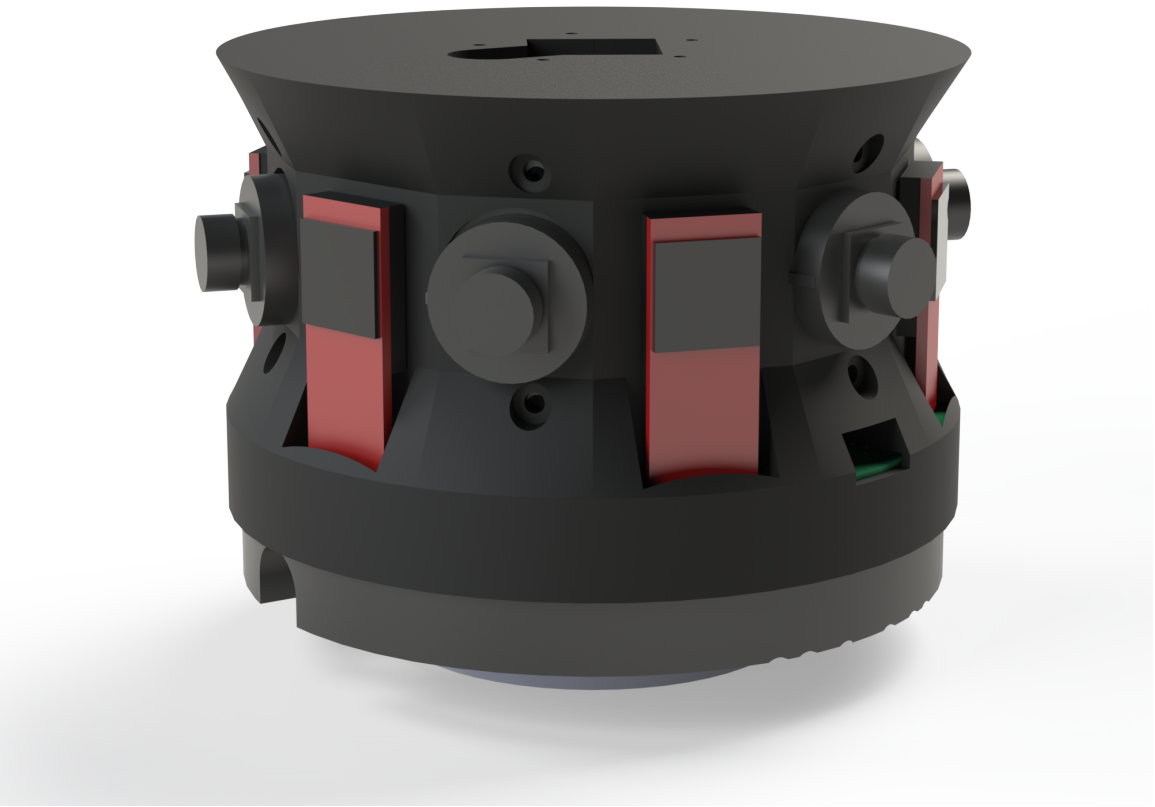


Figure 3.1: The proposed solution

3.2 Radar

Radar systems have been in use for decades and their technology is used widely in defense and military applications. Generally, radar systems are noted to contain large antenna arrays and are mounted on large vehicles or ships. However, with the introduction of radar systems in medical fields, autonomous driving applications and virtual reality, their size, and power draw have significantly decreased. Now, radar modules can be present inside everyday phones, such as the Google Pixel 4, making their use widespread . Although the modules in phones and the systems on ships have many significant differences in how they perform, their intended use is still the same: to detect and identify objects.

For radar applications in human-robot collaboration, a small device is required with high resolution and comparatively short range. The most suited radar modules that would be able to perform in an industrial setting are the off-the-shelf radar modules designed for Advanced Driver Assistance Systems (ADAS). These modules are designed for high localized movements, for example, cars on a highway system moving in similar and opposite directions, constant change in environment, ADAS must work in rainy, windy, and sunny conditions, and should be able to detect a multitude of objects, such as cyclists, motorcycles, traffic cones, and humans. Radar systems optimized for the road should also draw as little power as possible, have a detection range within 50 meters, should have a wide field of view, and should be compact to fit in a car.

The Texas Instruments IWR6843AOP, shown in Figure 3.2, was chosen as the radar module for human-robot collaboration. This module is a radar module intended for the

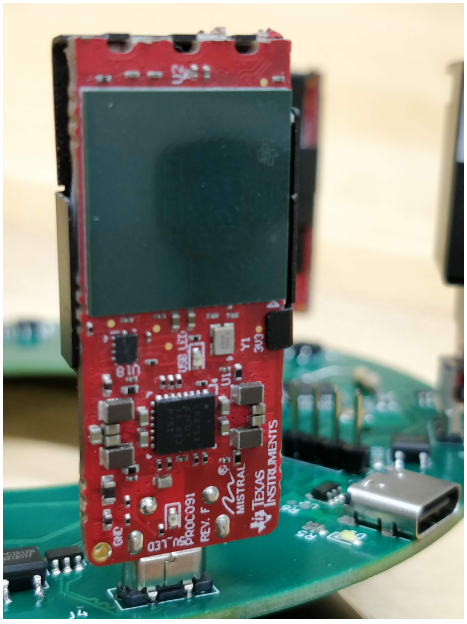


Figure 3.2: The Texas Instruments IWR6843AOP Rev F. with the debug portion broken off

automotive industry, however, adapted for the industrial setting. There is no weather protection for the module or rugged enclosure. This radar module comes in a much more compact package compared to other radar modules as it has its radar antenna on the radar module itself. Although this limits the number of antennas the module can have, it allows for a predetermined layout of the antenna arrays, which minimizes the time needed to implement the radar. This module has a field of view (FoV) of 130 degrees azimuth and 130 degrees elevation. This wide FoV minimizes the number of radar modules needed to cover a 360-degree field of view and minimizes blind spots in detection.

3.2.1 Design of the Radar System

To get the full 360-degree coverage, only three modules are needed. However, to ensure objects can be successfully detected at the edges of their operating FoV, and if any issues

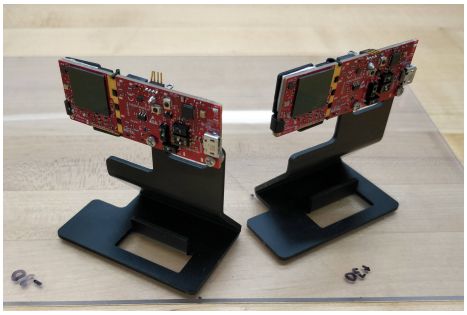


Figure 3.3: Two radar modules mounted to a acrylic holder for testing

arise that require the shutting down of one of the modules, the system needs to be able to continue to operate and safely bring the robot to a stop. For initial testing, a laser-cut piece of acrylic was used to hold two radar modules in place using their stands, shown in Figure 3.3. Each sensor was placed at a radius of 3 inches giving enough room for a camera module to be placed and allowing heat to be dissipated from their heat sink easily. One reason as to their placement was the physical limitations in size for testing. Each sensor module has two circuits, one that contains the IWR6843AOP radar chip and the other for access to debugging ports and extra communication ports. The debugging side can be detached and the sensor becomes much smaller and compact. During initial tests, however, everything was kept intact to ensure the radar modules worked correctly. This system was connected to ROS to view the initial pointclouds and modification of the configuration of the radars was done on this platform.

During the development of the radar modules system, Texas Instruments discontinued the IWR6843AOP evaluation board as they found an issue with the radar module. Later they updated it with a silicon revision and now it is referred to as the IWR6843AOP Rev F. This new module had a different evaluation board which altered their placement and a new

testing mount was created. As the USB port is centered on the evaluation board, it allows for a tighter pattern and needs less support to hold the sensor, shown in Figure 3.4. Because the previous software was able to be used with the new AOP module, the new modules were detached from their debugging circuitry by cutting the PCB along its perforations. They were then directly mounted to the ring holder.

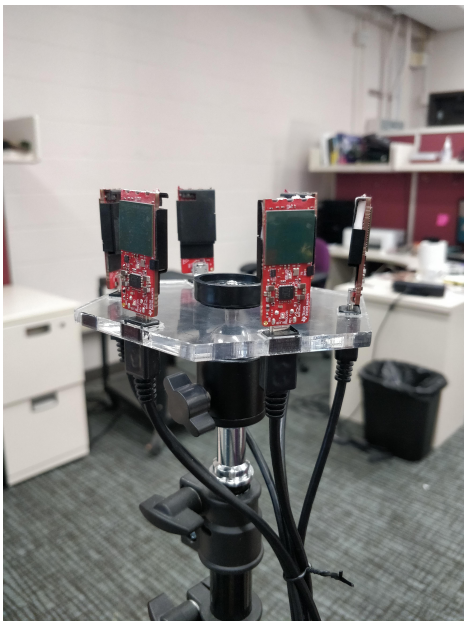


Figure 3.4: The six radar modules mounted to a acrylic holder to test all modules at the same time

3.2.2 Radar Processing

One of the benefits of using the AOP radar module was its versatility and customization. The IWR6843AOP contains a 32 bit ARM cortex-R4 processor running at 200Mhz and a DSP co-processor solely for radar data processing. The ARM processor can be programmed to provide simultaneous computation alongside the DSP core and can be interfaced through LVDS, SPI, Quad SPI, I2C, and UART protocols. On the IWR6843AOP evaluation board, the UART port is connected to an FTDI chip and allows for easy access

through a micro-USB port. The rest of the protocols are accessible over a 60 pin Q-Strip connector (commonly used for MIPI).

The DSP core on the IWR6843AOP is set up to process the raw ADC data and output pointclouds through four stages. Each of these stages has to be configured upon boot for the radar to start outputting data. The first stage takes the raw analog values and bins the data into radar batches for each pulse. For each range, this stage performs a Range FFT to set up for analysis of the target's distance as the range is proportional to the frequency. The same radar batch is processed in the second stage which performs a Doppler FFT and saves the output into L3 memory. The third stage performs CFAR (Constant False Alarm Rate) on both the Range and the Doppler data separately. CFAR constantly estimates and adjusts a threshold that divides the noise from the signal. For a starting point, the CFAR thresholding values are configured during the initial configuration phase. Lastly, the detections from the CFAR pass into the last stage which performs an angle of attack (AoA) analysis and estimates the azimuth and elevation angles by performing a 3D FFT. These angle estimates along with range and doppler information get encoded into a pointcloud and get transmitted for each frame of radar transmission.

As the output of the pointcloud is significantly less data than the output of the ADC, it was possible to transmit the pointcloud information over UART alone. To make this happen, the board was flashed with TI's latest firmware that enables access to the debugging features and pointcloud data through UART. Using the demo ROS package also provided by TI, through the mmWave SDK, it was modified to receive the raw TLV (Type, Length, Value) messages, extract the x,y,z, and velocity estimations, and publish ROS pointclouds.

These pointclouds can be accessed for further processing by subscribing to the ROS topic for each radar.

3.2.3 Configuring the Radar

One of the difficult tasks is setting up the IWR6843AOP correctly for the task at hand. In general, radar systems are very sensitive to environmental changes and need to be calibrated for each environment. For example, the radar system in the industrial environment will not have to deal with rain or clutter coming from the real world but will have to deal with heavily populated spaces with a lot of metal and moving objects. The IWR6843AOP has an easily configurable DSP core that performs all of the radar data processing, allowing the radar module to be used in a variety of scenarios when tuned to the task at hand. This makes the difference between accurately detecting the intended target and missing it altogether. Also as this application is concerning the safety of humans, thus the distance and velocity measurements need to be as precise as possible.

The IWR6843AOP is a radar whose antennas are tuned to 60-64GHz. Because the ranging of detection is crucial in the dSSM implementations, the maximum range and the range resolution need to be calculated to ensure the radar module is a viable sensor. The maximum range is calculated using equation 3.1. To do this, the parameters were sourced from TI's documentation on the IWR6843AOP and it determines the module can range up to 50m.

For a radar, typically the range resolution of the radar would be inversely proportional to the pulse width transmitted. However, to get a high range resolution, the pulse length

$$R_{max}^4 = \frac{P_t G_t^2 \lambda^2 \sigma}{(4\pi)^2 S_{min}} \tag{3.1}$$

3.1: Where R_{max} is maximum range, P_t is transmit power, G_t is antenna gain, λ is wavelength, σ is the radar cross section of target, and S_{min} is the power of minimum detectable signal

would have to be extremely short, and amplifying a short pulse requires a lot of power. To mitigate this problem, pulse compression techniques were developed that allow one to get the range resolution while still having a long pulse width. One that can be set up in the IWR6843AOP is linear frequency modulated waveform (LFMW) which sweeps the frequency of the pulse at a known rate. For an LFMW the bandwidth of the signal is the difference between the initial and final frequencies. As the IWR6843AOP can transmit between 60-64GHz signals, the range resolution is 37.5mm as calculated using equation 3.2.

$$\Delta r = \frac{c}{2B} \tag{3.2}$$

3.2: Where Δr is the transmit power, c is the speed of light, and B is the bandwidth of signal

$$B_{LFMW} = f_2 - f_1 = 64GHz - 60GHz = 4GHz \tag{3.3}$$

$$\Delta r = 37.5 \tag{3.4}$$

A range resolution of 37.5mm, as calculated by equation 3.2 3.3 and 3.4, is viable in this industrial scenario as it is just enough to detect the arm of a human.

As velocity is also a part of the dSSM calculation, velocity characteristics need to be assessed as well. Radar systems, unlike lidar and camera systems, can directly measure velocity without estimating it, however, the velocity measured is in the radial axis of the radar. This is done through a pulse doppler technique which during a single frame of radar

transmission, many pulses equally spaced out in time are emitted. Many techniques use a combination of object tracking and the doppler velocity measurements to derive lateral velocity, but it is not done in this research as dSSM looks at velocity to and from the robot. The doppler resolution is proportional to the transmission frequency and depends on the sampling frequency of the ADC.

$$f_d = \frac{2V}{\lambda} \tag{3.5}$$

3.3: Where f_d is the doppler frequency, V is the velocity of Detection, and λ is wavelength

$$f_d = (0.4mm/s)/Hz \tag{3.6}$$

This means that after the Doppler FFT is performed, knowing the difference in frequency between two samples can tell the estimated doppler velocity difference. To increase the Doppler resolutions, one can set the ADC's sampling frequency higher as it will be able to resolve velocities at a granular resolution. Another is to transmit many pulses in a single radar frame. This will require much more processing and memory to store, however, it will ensure targets at longer ranges will have the same velocity resolution as closer targets. Applying the Doppler FFT for each receive channel can help with accurately identifying the velocity and even more so if the pulses are staggered upon transmitting among the transmit channels. Having a high-velocity resolution is independent of having high range resolution however both parameters will detract from having short frame times and high-frequency pointcloud output.

When connecting the Radar module over USB to interface the UART ports, two UART

ports appear with independent configurations. The first is a console port able to send configuration parameters upon boot as well as starting and stopping the sensor. Each command along with its associated parameters are listed on the user guide for the IWR6843AOP. To set up the modules for HRC, doppler resolution was placed as a higher priority to max detectable range. The goal is to be able to detect a human within a range of two times the workspace size which will give the robot ample time to react. To reiterate, the max range resolution is fixed; it depends on the bandwidth of the transmitted signal (max 4GHz). Using TI's mmwave demo visualizer allows one to configure the exact parameters necessary for configuration while ensuring the parameters are within the bounds for the particular sensor.

Table 3.1 The configuration parameters for the IWR6843AOP

Parameter	**Value**
ADC Resolution	16 Bit (Max)
ADC Samples	496
ADC Sampling freq	2.583 MS/s
Chirps/frame (multiple of 4)	384
Frame Duration	100ms (10Hz)
Start Frequency	60 Ghz
Frequency slope	20Mhz/us
Ramp end	200us

High Pass Filter 1	175kHz (min)
High Pass Filter 2	350kHz (min)
Range CFAR initial gain	15dB
Doppler CFAR initial gain	15dB

The parameters used to configure the radar module are described in Table 3.1. Parameters such as the ADC Resolution, ADC Samples per Chirp, and ADC Sampling frequency are set to maximum. This is dependent on the ADC characteristics and the onboard high pass filter sizes. The maximum chirps/frame for a 100ms frame duration came to 384 chirps. This allows for the transmit antenna to successfully power down after the previous chirp, is a multiple of four as the window used for the Doppler CFAR requires it. To maximize the bandwidth of the transmit, the frequency slope and duration of the frequency ramp is set as such. Lastly, the initial CFAR gain was recommended by TI. After configuring the radar module, it is ready to output pointcloud frames.

3.3 PCB

To bring all the radar modules together in a compact manner, a PCB was designed to easily connect and power up the radar modules, as shown in Figure 3.5. The biggest issue with the IWR6843AOP modules is how they communicate to their host computer. Each sensor, on boot, is given a communication port to attach to and send data to. This is attributed on

a first-come first-served basis. Because every radar sensor is identical, they have the same vendor ID and device ID and the host machine has no way to individually identify specific radar modules other than that there are multiple connected. As all the six radar modules boot, it creates confusion to the user as to which sensor has which serial port. To resolve this, they must be booted individually with a delay, which ensures the correct module gets a predetermined port. If any issues arise, the modules can easily be rebooted and the process can be restarted.

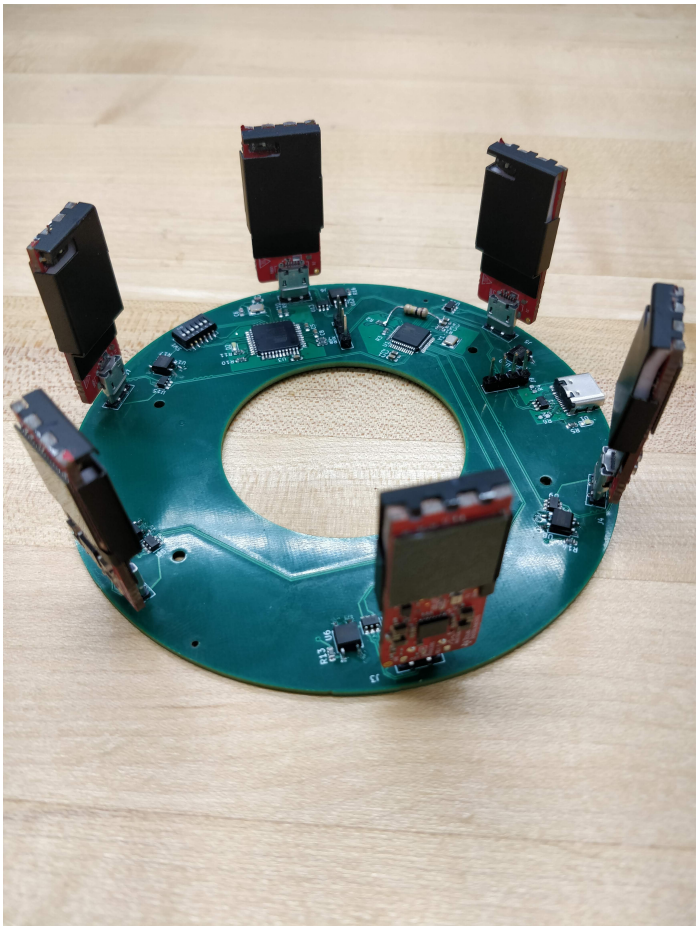


Figure 3.5: Radar modules on the custom designed PCB

Another purpose of the PCB is to distribute power to each of the radar systems. When

all IWR6843AOP modules are running, the maximum power draw can reach up to 3A of current at 5V. Having a single PCB can allow each module to tap into a single regulated power plane without needing individual power from each of the USB controllers.

For controlling the boot of each radar, an Atmel ATMega32u4 microcontroller was used. It is a 32-bit microcontroller found in the Arduino Leonardo platform. This controller has direct USB access without needing an FTDI chip and it runs directly on the 5V supplied by USB. To power up the radar, a solid-state optocoupler is used to connect power to the device. This allows for the microcontroller to be isolated from the radar sensor, protecting it from any current spikes. For manual access to the power control of the radar modules, each GPIO line is routed through a DIP switch.

To minimize the number of connections to the radar sensor board, a single USB C cable was used to deliver power as well as all the data lines. As the USB C specification allows, some USB C cables can handle over 100W of power delivery topping at 5A. As the radar modules draw a maximum of 15W altogether, the radar board is easily supplied through the cable. For passing data through the USB C connector, a USB Hub chip was used to merge the data lines. In the USB2.0 specification, up to a maximum of seven devices can operate on a single hub chip, which is perfect as there are six radar modules and one microcontroller. The hub chosen was the MaxLinear XR22417.

The creation of the board shape was difficult as the placement of the radar modules were critical. Using the prototype made of acrylic earlier, a circular PCB was made with vertical USB-micro connectors to mount the radar directly to the PCB, but the final size was unknown until the entire camera-radar system was designed in CAD. The final shape was

a 110mm diameter ring with an inner diameter of 50mm. The ring allowed for cabling and other mounting hardware to be passed through the center of the PCB without needing to use the PCB as a structural element. This minimizes strain and flex on the PCB and lessens the chance of damaged or delaminated traces which did happen on an early revision of the PCB. Each Micro USB vertical header was placed 60 degrees offset from each other with a radius of 50mm and the USB C connector was placed in between two radar connectors. All these measurements were necessary to build a model in CAD and to be used in the visualization of radar data in ROS.

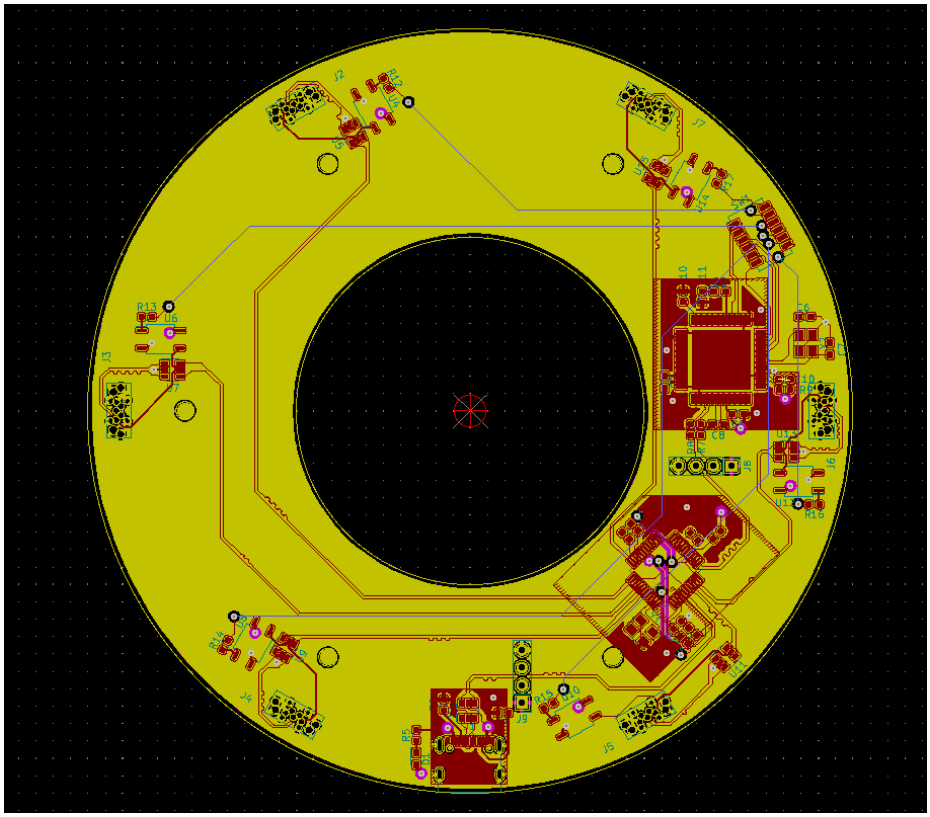


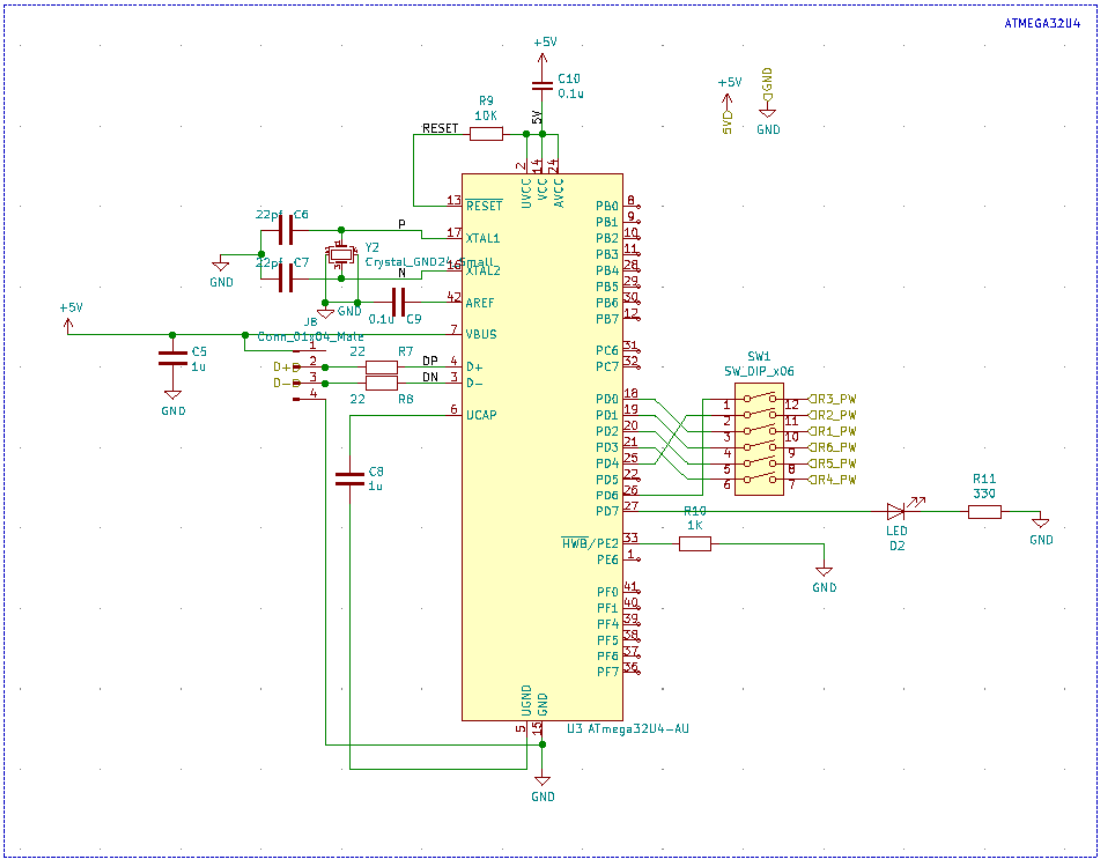
Figure 3.6: Layout of PCB

The PCB stackup has four layers, two signal planes, and a ground and power plane. As this PCB had USB2.0 traces, the ground plane significantly decreased the EM noise

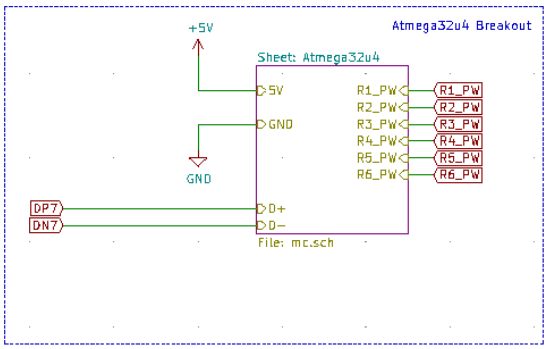
generated by the differential lines. The power and ground planes made it much easier to route many components as no power or ground lines needed to be routed. This also made it easy for current to the radar to easily pass through the power and ground planes. When routing the USB lines, a trace width of 0.2mm with the dimensions of the PCB stackup given by the manufacturer was used to create a 90Ohm impedance line, which is in the specification of USB 2.0, and was matched to within 0.1mm or less in length. Each USB line also had ESD protection through an STM USBLC6-2SC6; which were placed near each port. ESD protection was necessary as any stray voltage spikes could easily damage any components on the same bus.

Laying out the Atmega32u4 microcontroller was straightforward as common layouts were available open source with proven success; as shown in Figure 3.7 The most difficult part was understanding the requirements for the external oscillator circuits for both the microcontroller and USB hub. A 16MHz 12pF crystal oscillator was used with 22pF load capacitances for the Atmega32u4 and for the XR22417 hub, its load capacitances were internal to the chip and just a 12MHz 20pF crystal oscillator, as recommended by the datasheet, was used. These oscillators are used for keeping the internal clocks in time and ensures proper operation in digital circuitry.

This PCB went through two revisions as the hub chip was incorrectly laid out on the first revision. The first revision used a 64 pin LQPF package and allowed for current sense lines to prevent over current draw. As the radar modules were not being powered by the host machine, the need for current protection became unnecessary. The biggest issue with the XR22417 chip and similar varieties such as the more popular Terminus FE2.1 is the lack of



(a)



(b)

Figure 3.7: (a) The Atmega32u4 schematic (b) The breakout symbol used on the main page showing the USB connections and power lines

documentation for the layout guidelines. One that was missed was the onboard 3.3V and 1.8V regulators needed to be externally routed to other components on the same chip. In the second revision, this was fixed and the hub was functional. Additionally USB debug headers were placed as well as several test points and indicator LEDs for debugging if any issues arose. The eventual schematic for the XR22417 chip is in Figure 3.8.

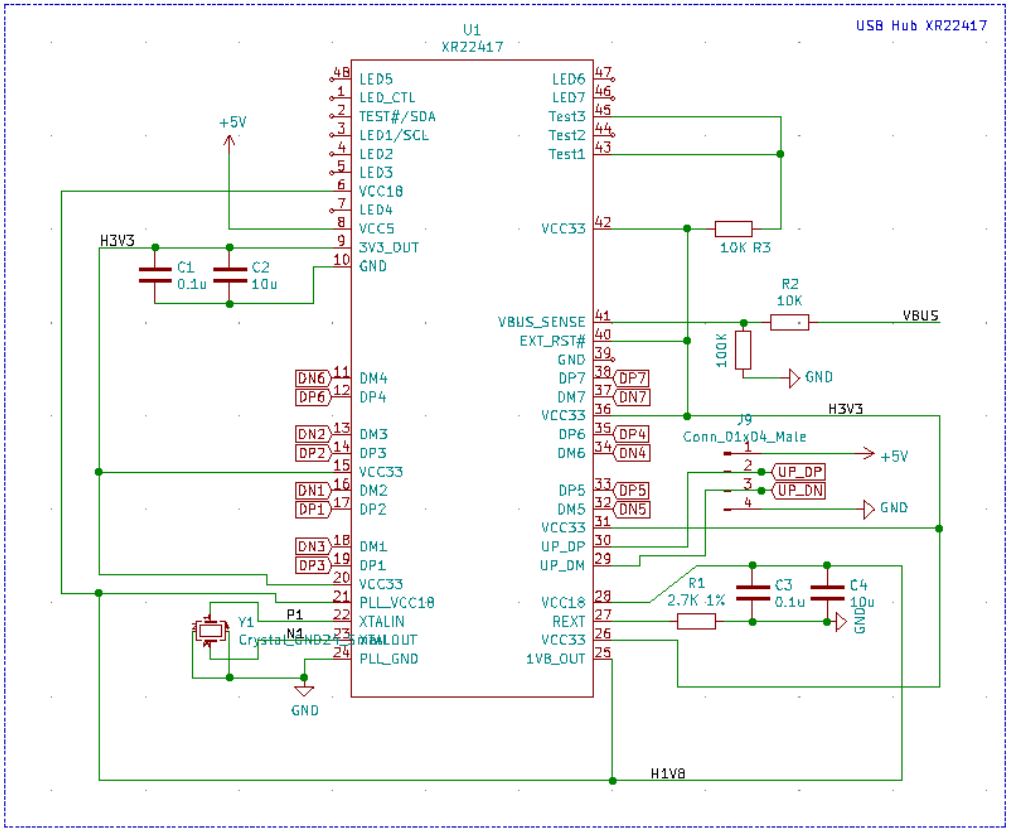
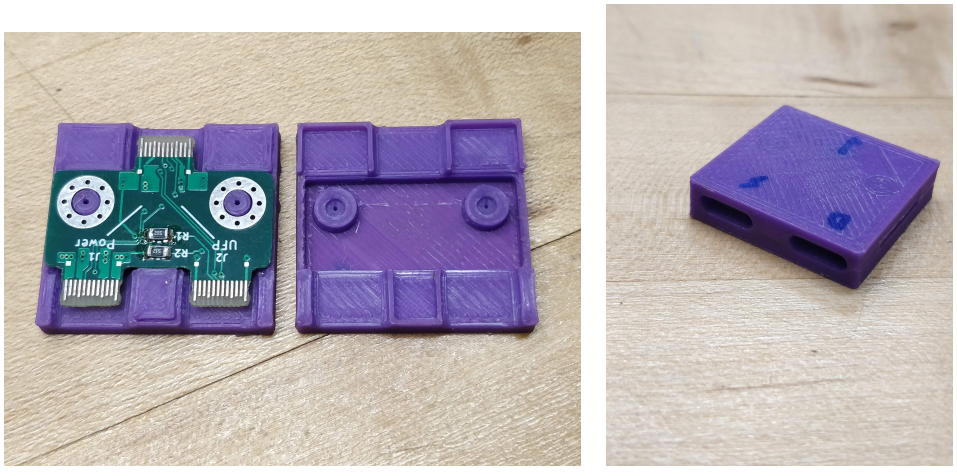


Figure 3.8: The breakout of the Maxlinear XR22417 USB Hub chip after the second revision

An issue with the current design was that there was no USB C port on a computer that was able to output 4A of current. To safely power the system, the data lines from the USB C port needed to be split for power and data. This was done with a custom PCB that was inspired by Clara Hobbs's PD Buddy Wye, and modified to take in a single USB

C cable, shown in Figure 3.9. It was designed to take in a power line over USB C, and a data line through USB C and merge them into one for the radar sensor system. This simple board did not need any ports other than two resistors for letting the host know the Current Capability (CC) of the system. No connectors were used as the PCB integrated the connections directly. This PCB had to be created at a height of 0.6mm so it could fit inside the USB C female connector and connect to each pin within.



(a) (b)

Figure 3.9: (a) The custom USB C breakout for power delivery and data transfer seperated (b) The 3D printed enclosure with labels: Lightning for Power, D for Data and Arrow for Radar

3.4 Nvidia AGX Xavier

For processing the radar and camera data, the Nvidia AGX Xavier was chosen, shown in Figure 3.10. It is an embedded GPU platform that is very capable in its performance for its size. It runs Ubuntu 18.04 with Nvidia's Jetpack 4.3.1, a custom kernel with drivers and device-specific libraries. The Xavier has a custom GPU as well as an 8-core ARM processor which is enough to run ROS, embedded image processing, and machine learning algorithms in real-time. Also as this is made by Nvidia, the libraries are optimized to make

full use of the GPU and its tensor compute cores. It also contains a PCI-e 4.0 x8 slot for a secondary peripheral such as a GPU or network card, and direct access for cameras though its 16 CSI-2 lanes.

Figure 3.10: The breakout Nvidia AGX Xavier with D3 Engineering's FPD-Link III Breakout board and a 3D printed holder for the full assembly.

3.5 Cameras

Cameras were used to understand where the human is around and within the workspace of the robot. Cameras are the perfect technology for this as they operate at high frame rates and each frame contains much data about the environment around. With modern image processing algorithms, one can detect many different objects in many lighting conditions and locate them in the frame. To choose the ideal camera sensor for the job, one needs to understand the output resolution of each camera, the pixel size, the dynamic range, and the available lenses.

For the implementation of human tracking in a robot's workspace, the camera frames will be processed through a pose detection algorithm which will identify where the joints of the human that are in the frame of the image. The human can be anywhere from several meters away to several centimeters away. Also, as this is running in an industrial scenario, lighting conditions can be assumed as optimal where there are no major shadows or dark spots in the workspace. The camera sensor will be outputting a large amount of data per second and enough processing power must be available to perform the pose detection. A high-resolution camera may be beneficial to capture the detail in the human but may significantly slow down the processing pipeline, increasing the latency to receive the pose information. Lastly, as the camera will be mounted to the robot, each will need to be able to capture an image without much blur or dynamic distortion: distortion as the camera moves around.

The initial camera chosen was the Flir Flea3 with a 1.3Mp e2v EV76C560 sensor, shown in Figure 3.11(a). This camera is a global shutter camera which means there will not be any rolling shutter or distortion as it moves around. The 5.3um pixel size allows it to capture a large amount of light and lower the sensitivity of the CMOS sensor outputting sharp, noise-free frames. The one downside to this camera is that it requires the use of a large lens which increases the size and weight of the camera. The lens used for testing was a Fujinon 6mm f1.2 lens. Because of the large focal length to Flea3's sensor size ratio, the field of view is 73.7 degrees, which will not resolve the full height of the human when close up. Lastly, the interface to the Flea3 is over USB3. Although the USB3 protocol's bandwidth is enough to support a 1.3Mp image, processing an image from USB3, depending on the platform, isn't

usually optimized and requires more processing power. Also having multiple cameras over USB3 makes this even tougher.

(a) (b)

Figure 3.11: Cameras used for prototyping and testing: (a) Flir Flea 3 (b) D3 Engineering IMX390 rugged camera module

Knowing the Nvidia AGX Xavier can concurrently connect to 16 different cameras at the same time using its CSI-2 lanes, cameras were searched for that could use this interface. D3 Engineering has an interface card that is designed to mount directly to the Nvidia AGX Xavier that interfaces cameras over FPD-Link III and converts it to CSI-2. The interface card attached to the Xavier is shown in Figure 3.10. FPD-Link III is a serialized protocol that can efficiently transmit high data throughput over long distances. By doing this, it will allow the cameras to be placed at a distance from the Xavier but having very little loss in receiving the images.

The cameras selected, to connect to the interface card, were D3 Engineering IMX390 sensor modules as seen in Figure 3.11(b). D3 provides drivers and the core ISP pipeline to correctly retrieve, correct for lens imperfections, auto white balance, filter, and encode images directly from the sensor. They have a built-in FPD-Link Serializer to transmit up to 24-bit raw images at 1920x1080 at 60fps. This sensor performs well in low light conditions

and the sensitivity characteristics can be optimized continuously for the scenario it is in. One of the lenses this camera comes with is a 1.83mm fisheye lens which gives 192-degree horizontal FoV and 120-degree vertical FoV. When using this camera rotated 90-degrees in a portrait manner, which will give a high vertical FoV, it will be able to observe the position of the full human body from very close to the lens. This is ideal for human tracking as it will allow for continuous tracking and virtually no blind spots. The minimum tested distance from the lens to view the entire human is 10cm. To introduce redundancy such that the system can operate when cameras fail, six of these cameras were used 60 degrees apart to be able to view the human anywhere around the robot.

Lastly, custom cables were assembled for use on the UR-10 Robot. D3's Cameras and Xavier FPD-Link III card connect over a 50Ohm RG174 Coax cable. This cable required a solid copper center conductor with a layer of shielding and housing as per the FPD-Link requirement. RG174 was chosen over RG58 as it has a better signal to noise performance. To connect to the Cameras, D3 Engineering chose to use Fakra connectors code z which resembles the universally keyed connector. This means the code z connector will be compatible with all other code connectors. Because the camera and Xavier card have male connectors, the cable must use female connectors. Because the UR-10 can span around two meters and is approximately 1 meter off the ground, a three-meter cable would be required. To design this sensor with accessibility and modularity in mind, an additional cable was chosen as a short extension so that one would not need to connect the cables to the camera and would connect to the extensions instead. This short extension was a 90-degree female Fakra connector and a regular straight male Fakra connector on the other end. This

extension cable allowed the sensor system to be as compact as possible.

3.6 Enclosure

A CAD model was used to design the enclosure for the radar modules and camera sensors, shown in Figure 3.12. Each part of the system was modeled including the radar modules, the PCB, the cameras, and the camera connectors. The goal for the enclosure was to protect the sensors, make a compact and easy to mount system, and to keep the inertia as low as possible. Knowing this system is to be mounted on a UR10 robot's end link, the sensor system would need to be able to hold mounting options for end effectors on top of the sensor. Because of this, the core of the sensor enclosure is designed out of aluminum that will be machined to bolt onto the robot's end link.

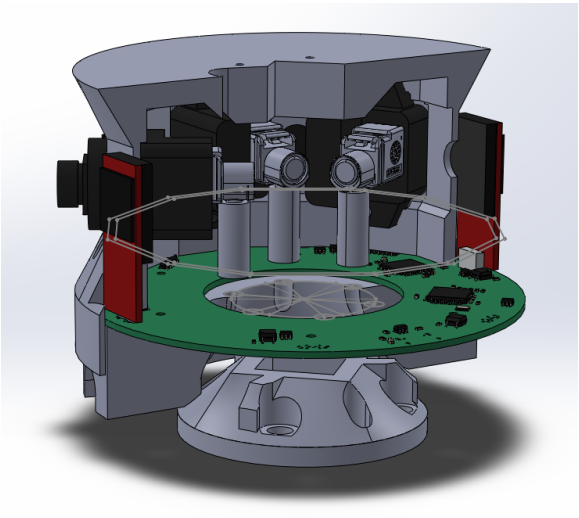


Figure 3.12: The CAD model of the entire assembly of the radar-camera system

The enclosure is 3D printed as it allows for unique mounting options and is lightweight but strong. The outside was printed with a 50% infill density and 0.15mm layer height

to ensure maximum rigidity. The material used was PLA as its ease of print, however, ABS would have been a better material for its thermal characteristics and its increased strength compared to PLA. Unfortunately at the time of printing, ABS was not available. The enclosure is designed to mount to the PCB and provides easy access to the DIP Switch and USB type C port. The CAD model was loaded on to Cura, a slicing software designed to make G-Code for 3D printers from 3D model, shown in Figure 3.13. The model was printed on an Ultimaker 3.

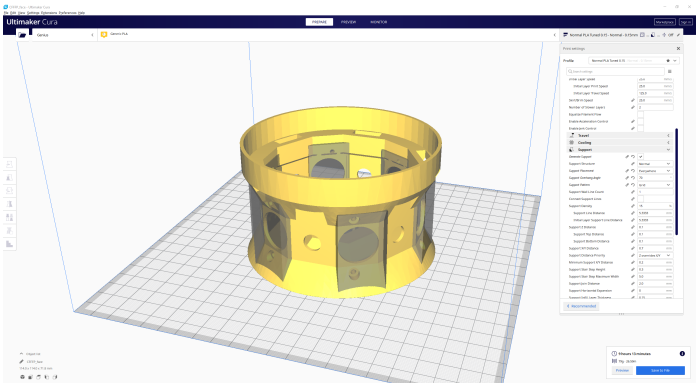


Figure 3.13: The CAD model loaded in Cura for 3D printing

Another 3D printed piece serves to protect the AGX Xavier and D3 Engineering's FPD-Link III breakout card and from anything shorting connections on the raw PCB, shown in Figure 3.10. After noticing how the temperature of the radar systems gets hot, the surfaces on the enclosure that are close to the radar modules were lined with Kapton tape, a heat resistant tape, to prevent melting and deformation of the enclosure, see Figure 3.14.

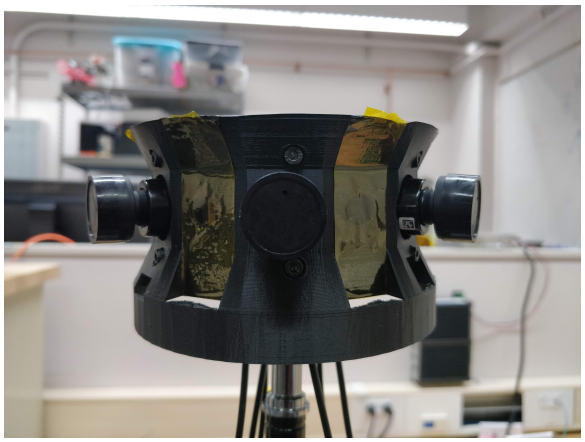


Figure 3.14: The radar enclosure wrapped in Kapton Tape where the radar modules sit.

3.7 Human Detection

To identify and understand the human's presence in the robot workspace, machine learning is used to process the camera feeds which will output the location of the human. For implementation in dSSM, not only is the location of the humans is needed but also a characterization of the human's position is also critical. As the human walks around the robot workspace, different segments of the human may be closer or further than the body to the robot. This additional step, to identify and locate the positions of the human body will help increase the safety of the human and will allow the human to ultimately work closer and for the robot to accurately move around the human. To do this, pose detection or skeleton tracking algorithms are used which use machine learning to output the locations of the identified human joints. Pose detection algorithms output 2d points as they are detected on camera frames, however, they can also output 3d depth points when using 3d sensors. In this implementation, the pose algorithm will output 2d points and the third dimension and velocity will be given by the radar modules.

To identify humans using cameras, several pose detection algorithms were compared. OpenPose, a popular open-source algorithm and widely used as its state-of-the-art two-branch multistage CNN promises high accuracy with multiple subjects. OpenPose was trained on the Body_25 and the COCO (common objects in context) dataset which offer many different performances. OpenPose is able to run on a multitude of operating systems, with and without GPU acceleration, and on embedded platforms as well. Another option is using Intel's Cubemos Skeleton tracking. This is a paid application that boasts high performance with large images while running only on the CPU. One benefit is that the Cubemos architecture is able to run efficiently on the most resource-constrained embedded devices. Although a license is required for use of the skeleton tracking SDK, the 30-day trial license was used for all testing. This has no impact on the performance of the algorithm. A downside to cubemos is that it is limited to run only on Intel processors. Another pose detection algorithm by Intel that is open source is the OpenVino. This also runs on Intel based CPUs but also Intel GPUs and their neural compute stick. This algorithm is based on OpenPose, and uses a similar architecture, but uses a tuned MobileNet v1 as a feature extractor where OpenPose uses VGG-19 [28]. The last pose algorithm tested was one by Nvidia called Trt-Pose. This pose algorithm makes use of the TensorRT cores onboard Nvidia GPUs and optimizes models designed in all libraries to make full use of the hardware. The base model for this network is a resnet-18 pre-trained on COCO in PyTorch. When the network is run on an Nvidia embedded platform, it initializes the network by optimizing the weights at half-precision on the device to use TensorRT.

To test these pose detection algorithms they were tested using a Flir Flea3 with a native resolution of 1280x1024 and resized to the input size for the network. The goal is to collect the pose information directly on the Nvidia AGX Xavier either through running the pose detection algorithms onboard the Xavier or processing them on a more powerful machine and sending the results back. For Cubemos and OpenVino, they were run on an Intel based desktop running Ubuntu 18.04. To ensure there was no bottlenecking of the network, both the desktop and Xavier were set up on the same local network with a gigabit switch dedicated for the two machines. To verify this, the link speed was tested and the latency came out to 1000Mb/s with an average 0.32ms latency. Lastly, to ensure the pose detection algorithm ran as efficiently as possible using every resource available to them, all machine learning libraries required were manually built using the Nvidia CUDA and cuDNN libraries over OpenCL when possible. These libraries were Caffe, OpenCV, OpenPose, Tensorflow, TensorRT, PyTorch, and Keras.

Specs of the Desktop:

Intel i7 5820k @ 3.30GHz (6 cores - 12 Logical Processors)

32GB 2133MHz Ram

Nvidia Quadro P4000 with 8GB VRam

480 GB Nvme SSD

Test 1:

The first test was to run OpenPose on a single thread capturing each frame and processing it

in a blocking manner. This is the simplest approach and each thread running will consume the maximum resources.

Test 2:

The second test was to run OpenPose in a multi-threaded framework. This allowed each frame to be pre-processed and buffered so the OpenPose algorithm would process frames consecutively. Each frame was resized to 600x480 which should improve the framerate. These tasks were split on their own thread.

Test 3:

The third test was to run OpenPose on the desktop that would buffer and process each frame. To make this happen as seamlessly as possible, each frame that was captured was encoded in *.jpg and sent over the network using ImageZMQ. OpenPose running on the Desktop would process the compressed frame and send the pose information over ZMQ back to Xavier. The round trip time was measured.

Test 4:

The fourth test was to run Cubemos on the desktop and similar to Test 3, transmit images over ImageZMQ and retrieve the pose information on ZMQ.

Test 5:

The fifth test was to run OpenVino on the desktop and similar to Test 3, transmit images

over ImageZMQ and retrieve the pose information on ZMQ.

Test 6:

The last test was to run Trt-Pose on the Xavier in a single thread capturing each frame and directly processing it.

After averaging the values of 50 different frames, the results are clearly favorable to Trt-Pose, as seen in Figure 3.15. The biggest contributor to OpenPose's performance was its inability to efficiently use the resources available. According to the OpenPose Github, 2GB of memory is needed for the COCO model which is a lot of resources Xavier does not have especially when scaling to six camera feeds. The expected boost in performance between the single-threaded test and the multi-threaded test because of the downsampling of the image didn't end up speeding the algorithm up much. This is to be estimated because of limited resources on the Xavier. Unsurprisingly, running Openpose, Cubemos, and OpenVino on a much more powerful machine showed its performance compared to when running on the Xavier, and this is expected. However, it was noticed that of that time, the image compression took around seven milliseconds. The most surprising result was Trt-Pose performing at, an average of over 50 frames, 165 frames/s. Visually, as seen in Figure 3.16, this framerate allows capture of a moving, stationary and jumping human. This shows how important optimizing networks for the hardware being run on and making use of Tensor cores when possible dramatically increases the performance of the model.

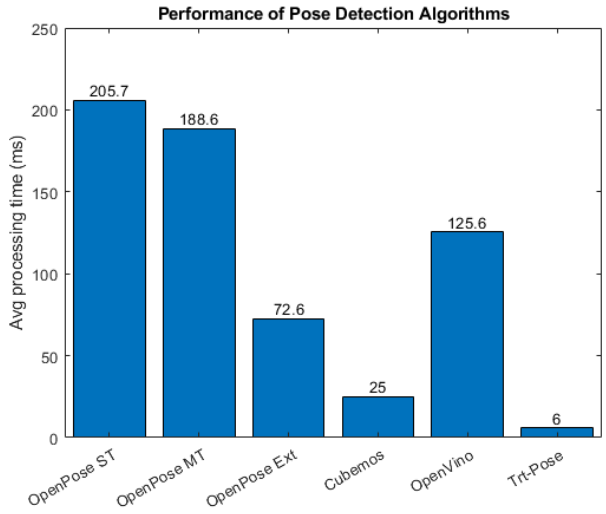
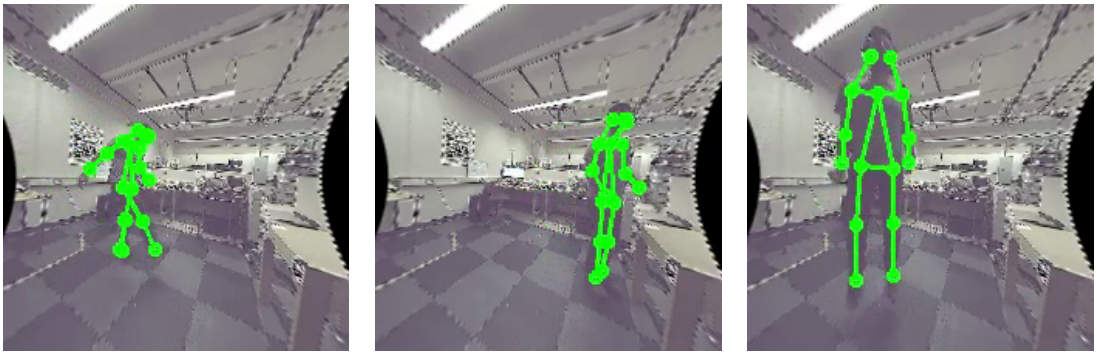


Figure 3.15: The Pose Detection Algorithms and their average processing per frame time in ms. (from left to right) OpenPose Single Threaded, OpenPose Multi Threaded, OpenPose over the network, Cubemos, OpenVino and Trt-Pose



(a) (b) (c)

Figure 3.16: Sample images from the output of TrtPose (a) when walking (b) when standing and (c) when jumping. Each image is 224x224 px

3.8 Software

The software to calcuate the 3D pose estimate was all done on the Nvidia AGX Xavier in Python and C++. The flowchart of how it was calculated is is shown in Figures 3.17 and 3.18.

The two main data transportation layers implemented were ROS and ZMQ. ROS was used as much of the industry uses it with support for a plethora of sensors, applications, simulation tools, and complicated algorithms. Wherever this radar-camera sensor system is mounted in the future, whether on an armed robot or a mobile robot, developers can easily access the raw data and develop their own approaches to solving problems. ZMQ is a socket-based embeddable network library that allows for extremely quick distribution of data. ZMQ is even faster than ROS as the overhead is much less. In certain aspects, when developing the software in sections of the processing pipeline that didn't require ROS, it was simpler and required less processing power to transmit data over ZMQ than to implement ROS and access the data through it. ZMQ was implemented using a publisher/subscriber messaging pattern. Another benefit of ZMQ is that as this data is broadcasted over sockets; it can also be accessed for further processing from other computers on the same network.

The main objective of the software design for the radar-camera sensor system is to calculate the 4D pose of the human as accurately and precisely with as high of a sample rate as possible. To get the 4D pose, the radar sensors, giving out pointcloud data, and the cameras, giving frames of data, need to be merged and then processed together.

Getting data from the radar sensors is straight forward; each of the six radar modules

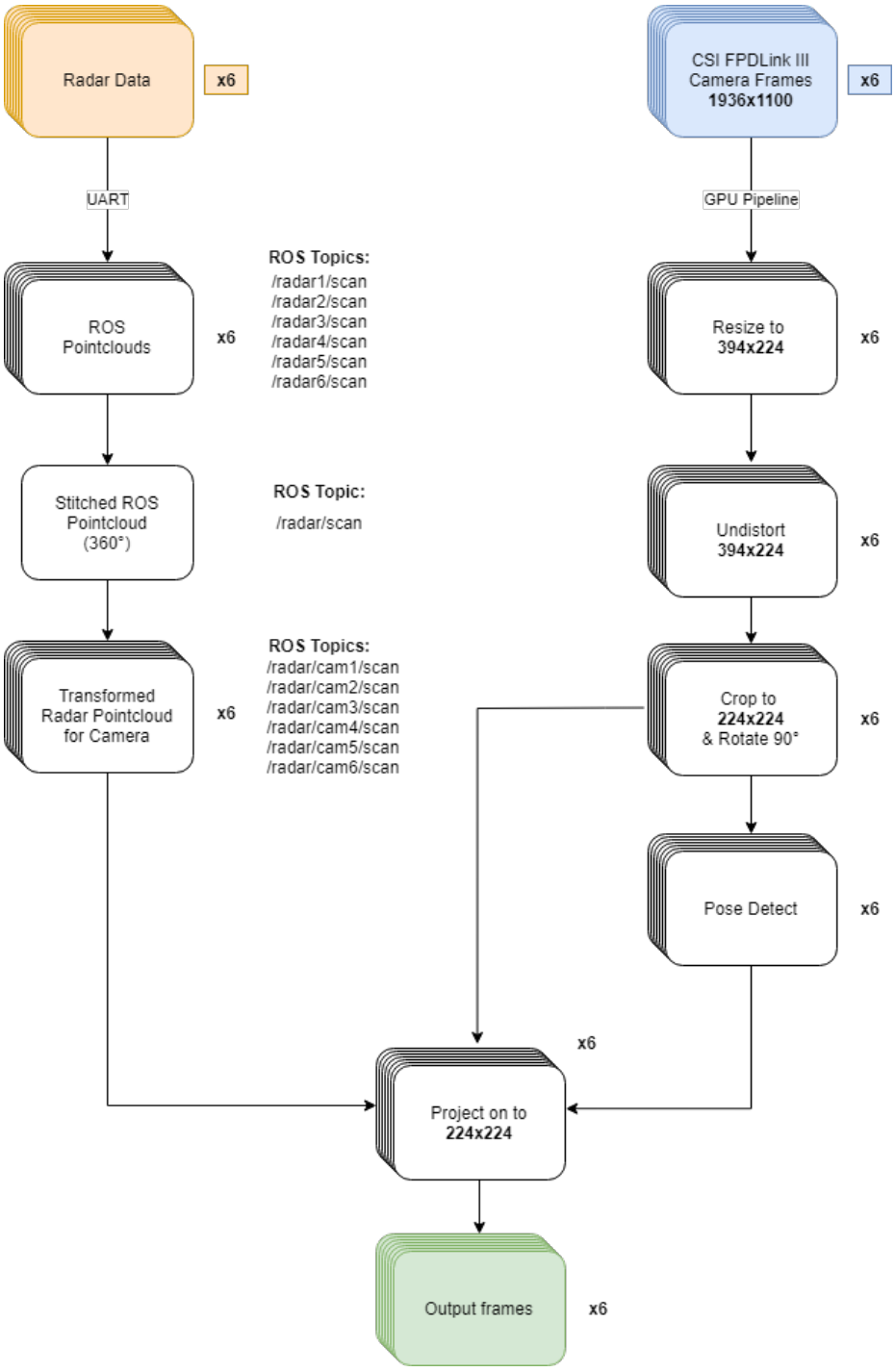


Figure 3.17: Flowchart of software pipeline to get radar data and 2D pose

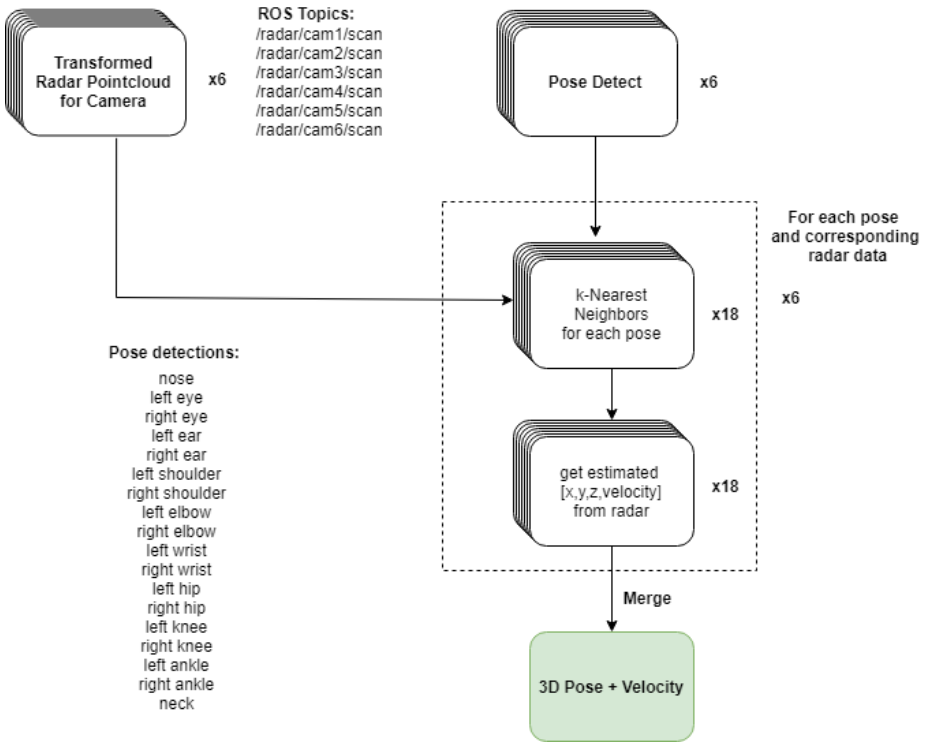


Figure 3.18: Flowchart of combining radar and pose data to create 3D pose

output pointclouds on ROS with respect to their frame of reference. To understand radar data as a whole, their pointclouds need to be transformed according to their locations and orientations in the radar-camera system. This required a rotation about the x-axis by 90 degrees and a rotation about the y-axis by the angle of the sensor's placement. As the sensors are placed 60 degrees apart, knowing which radar is in which location is key to making sure the stitched radar data is seamless. A simple way to move a 3D point in space is to use matrix transformations which are easy and quick to compute. (Shown Below)

$$rotation_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$rotation_y(\theta) = \begin{bmatrix} cos(theta) & 0 & sin(theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(theta) & 0 & cos(theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$rotation_z(\theta) = \begin{bmatrix} cos(theta) & -sin(theta) & 0 & 0 \\ sin(theta) & cos(theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

$$translate(x, y, z) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

$$A = rotation_x \cdot rotation_y \cdot rotation_z \cdot T(x, y, z)$$

$$x' = A[0, 3] \qquad y' = A[1, 3] \qquad z' = A[2, 3]$$

Since the point in 3D space maintains the same distance away from the sensor, the doppler velocity information sent alongside the 3D point in the pointcloud can be preserved. As each radar sensor has an FoV of 130 degrees, there is an overlap in radar coverage which causes redundancy in data. Instead of deleting the data, the choice was made to preserve it. This is because as the redundancies are separate measurements from two separate sensors, the measurement error may not place the same point in the same place, and there is a benefit of an increase in resolution of the pointcloud.

One thing to note is the radar sensors are not synced and pointcloud information is received asynchronously at 10Hz. When stitching the data together, each new pointcloud updates a buffer to the new points only overwriting that module's data. This buffer is sampled at 30Hz and a stitched pointcloud is published. This ensures no data is lost but

assists the image processing algorithm to not wait for new frames. This single pointcloud represents all the radar information from the sensor and any new algorithm that needs to access this data just has to subscribe to the radar topic '/radar/scan' on ROS and to port 7200 on ZMQ.

In the design of the radar and camera ring, each camera is placed in between each of the radar modules offset by 30 degrees. As the cameras are rotated by 90 degrees to give the maximum possible vertical field-of-view, the horizontal field of view of each camera is 120 degrees. For the camera's image frames to be correlated to the radar data, the stitched radar pointcloud needs to be sliced and transformed to match the orientation of the camera. This is done through Procedures 3.1 and 3.2 Because each camera is 120 degrees wide, separating the radar into six slices will not work. The stitched pointcloud needs to be evaluated for each camera and filtered to the corresponding orientation and field-of-view.

Procedure 3.1 Calculate the angle difference

Data: θ_1, θ_2
Result: The difference in angle in degrees
return *(θ_1 - θ_2 + 540) % 360 - 180*

Procedure 3.2 Generate radar pointcloud mapped for each camera

Data: pointcloud, cameraID
Result: pointcloud relative to camera
foreach *point in pointcloud* **do**
    θ = atan2(y,x)
    i = position of camera in degrees
    buffer = [ ]
    if *θ is in between (i ± fov of cameraID)* **then**
        add point to buffer
    return *buffer*
end

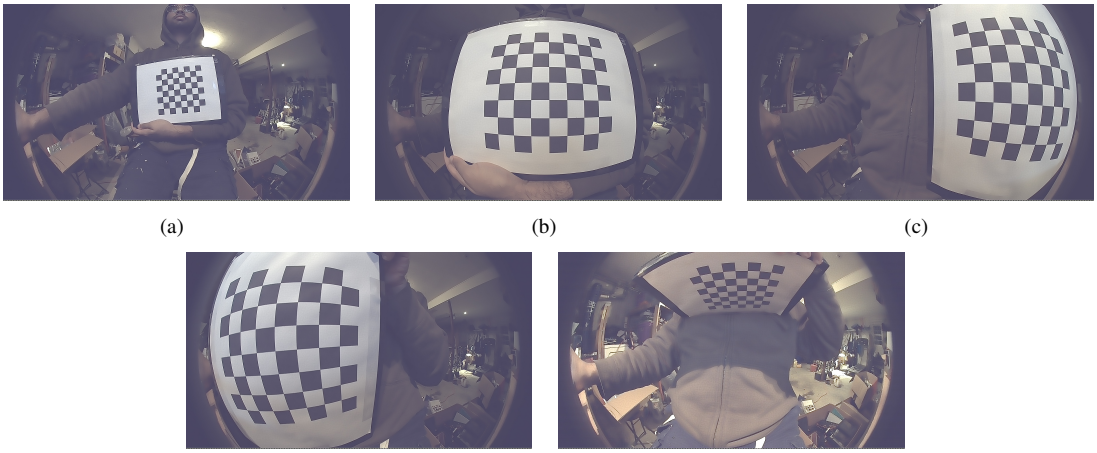
To do this efficiently, each point in the stitched radar pointcloud was compared to each cameras view angle to determine which camera the point belonged to. Then each point was transformed into the inverse transformation from the camera to radar. This inverse transformation was derived by the CAD model to ensure its accuracy and was visualized in Rviz, a visualizer in ROS. Each of these new pointclouds, one for each camera, was published over ROS and ZMQ so that separate analysis of the camera and radar information could be performed.

The processing of the camera data was done in parallel to the radar pointclouds. As the radar data primarily used the ARM CPU on the Nvidia AGX Xavier, the camera data was able to make full use of the onboard Volta GPU. To access the camera frames over the CSI-2 ports, a GPU pipeline was set up using a custom-built gstreamer library for the Xavier. Gstreamer allows one to perform simple image processing all in a pipeline before it is encoded and accessible to the application in need. In this case, the application is OpenCV and is tied to the end of the pipeline where the rest of the image processing will take place.

3.8.1 Un-Distortion of the Image

The first task is to un-distort the image. To do this a calibration checkerboard was used and many images with the board were taken, seen in Figure 3.19. This checkerboard provides a reference plane for an image processing algorithm to estimate the properties of the lens that cause the distortion in the image. Images from the checkerboard detection can be shown in Figure 3.20. This shows colored lines that should be parallel, but are not because of the distortion of the image. This technique to un-distort images assumes the camera lens

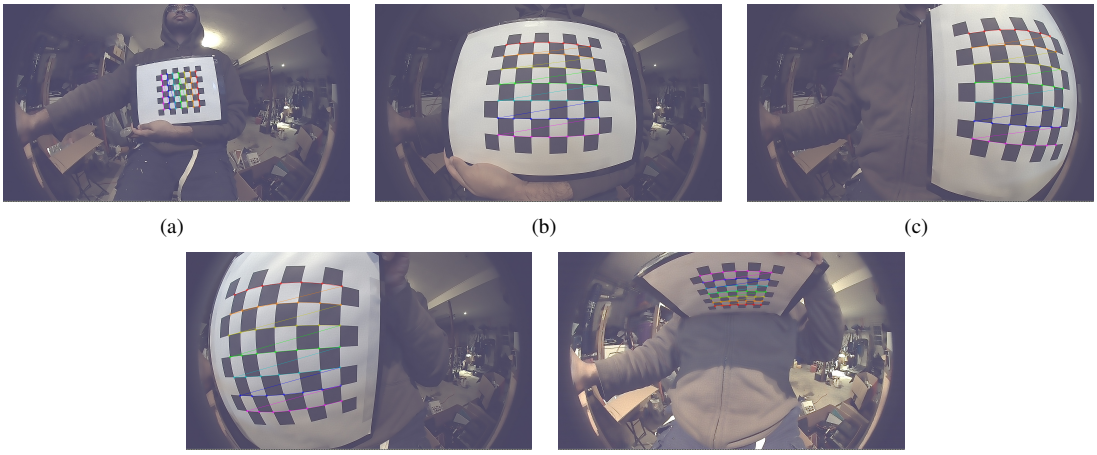
is a pinhole which allows the use of the Zhang's method to solve for a linear system that describes the lens camera system.



(a) (b) (c)

(d) (e)

Figure 3.19: Several of the images used for calibrating the fish-eye lens



(a) (b) (c)

(d) (e)

Figure 3.20: Images after locating the checkerboard and drawing lines across

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

The calibration matrix in equation 3.11 contains the intrinsic characters that will be used to undistort the image. In it are the focal length, characteristics about the pixel size, the

principal points, and the skew coefficient between the x and y-axis which are discribed by α_x, α_y, γ, u_0, and v_0. To perform the undistortion, OpenCV has several methods. Recently, in versions 3.0 and greater, OpenCV has a fisheye library that has a better, more repeatable, methodology to calculate the mapping from a distorted frame to an undistorted one. This was used to calculate the calibration matrix and the mapping of each camera system.

To use the calibration matrix and the distortion coefficients, it needs to be scaled to the image size being used. During the calibration matrix, the image center is calculated with respect to the input image from the camera. When the input image that needs to be undistorted is scaled, so does the mapping. This knowledge was not found in the OpenCV documentation for undistorting images and was learned through much trial and error.

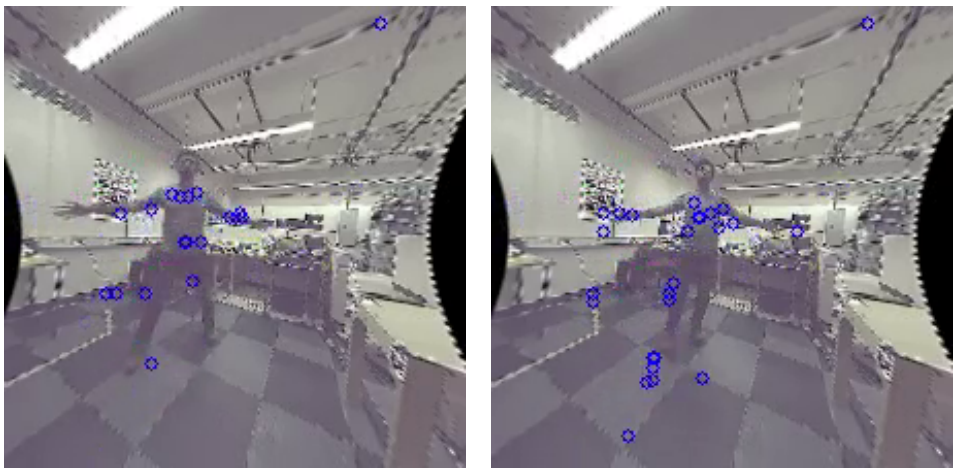
3.8.2 Pose Detection and creation of 3D Pose

For each image to be passed through the pose detection algorithm, it needed to be sized as a 224x224 square image, the size of the input layer for Trt-Pose. To achieve this, the image was first resized to 224x280 and, after the distortion correction, was cropped to 224x224. This kept the horizontal FoV of the image constant which is crucial for mapping the radar data onto the image without skewing the vertical axis of the image.

$$x' = \frac{(x * \alpha_x)}{z} + u_0 \qquad y' = \frac{\left(y * \alpha_y\right)}{z} + v_0 \tag{3.12}$$

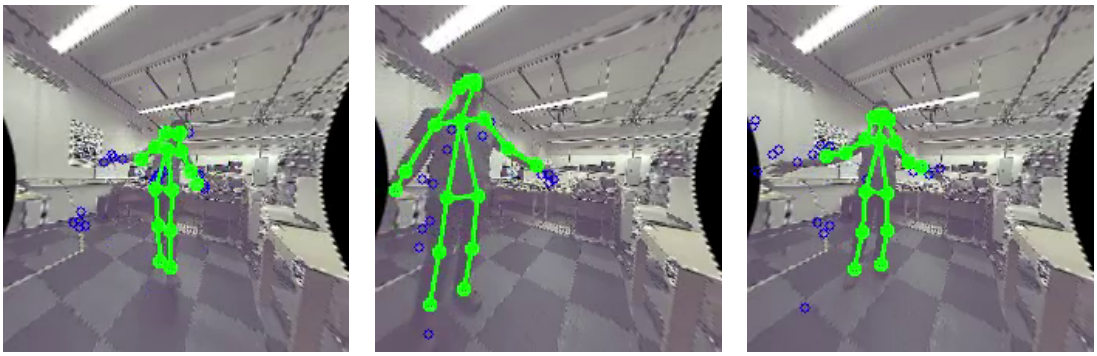
After the pose detection algorithm was run, the pose data and the radar data were projected onto the image. To project the radar data, the calibration matrix was used again. This is described by the set of equations in 3.12. Because it tells the properties of the

lens/camera system in a linear system, it can be used to project 3D points in space onto a 2D plane, shown in Figure 3.21.



(a) (b)

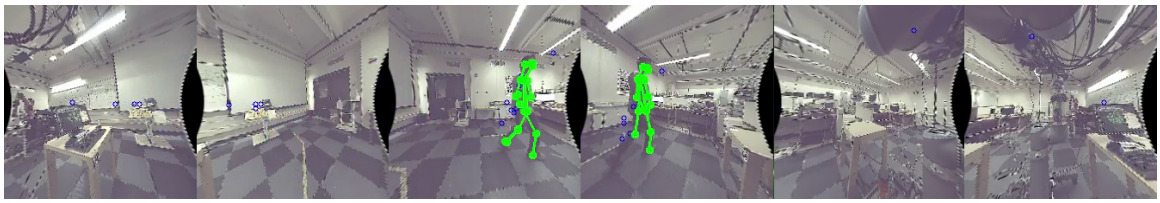
Figure 3.21: Radar data being projected onto the camera frames. (a) Opening up arms increases the number of radar detections on the arm (b) The right leg is moving in an upward kicking manner and the detections also increase. The stray points are from other detections/reflections



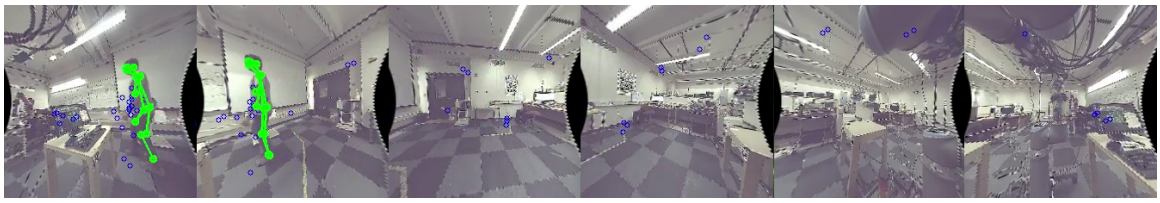
(a) (b) (c)

Figure 3.22: Once overlaying the pose information, the accuracy of the pose and radar become evident.

Once having the projected radar data and the pose data relative to the same frame of reference, seen in Figure 3.22 and 3.23, a k-nearest-neighbors search on the radar data for each pose data will output the estimated pose data in 4D (3D point + velocity). A k-nearest-neighbors search is a process to locate the k closest points in a pointcloud without having



(a)



(b)

Figure 3.23: Here a 360 degree image of overlayed radar and pose data is shown showing the accuracy of the pose estimation and radar data. One key benefit is highlighted as there will always be two humans detected at all times for each human in the real world.

to do an exhaustive search. This was implemented using the sklearn library. The premise is to build a tree on the radar data and query it with the pose data. The output is a list of k points closest to each pose location. For initial testing, *K=1* was used to understand the behavior and was increased to *K=3* to smoothen the result. As there were few radar points at certain samples, there would be issues in finding the closest radar points as there would be high error. This was fixed by using the last five published radar data points aggregated through a rolling buffer, and used for building the k-d tree. Using the closest radar points, the velocity and depth information was inherited by the pose point which makes a 3D point.

3.8.3 Future calculations

Calculating the minimum separation distance between the robot and the human requires both the pose of the human and the location of each joint of the robot. An easy assumption to make is to get the distance of each joint in the human with respect to the radar-camera

system. When the radar-camera system is mounted to the end effector of the UR10 robot, all the distances and velocities are with respect to the end effector of the robot and not to the robot itself. To better understand the shape of the robot, a pose like analysis of the robot needs to be done, also known as kinematics of the robot. To do this, the kinematics of the robot are taken directly by communicating to the robot and each joint angle is converted to joint locations of the six degree of freedom robot. Putting all these points together generates a pose pointcloud of the robot. Here, another k-nearest-neighbors approach can be used to find the closest distance from the human pose and the robot pose.

Chapter 4

Results

The radar camera sensor system was mounted on a UR-10 robot, seen in Figure 4.1, and tested in multiple scenarios for its accuracy and precision in detecting a human in the workspace. As the primary implementation of this sensor system is to be used for dSSM, the sensor needs to understand the position of the human as well as the velocity. For the baseline measurements of the human in the workspace, an OptiTrack motion tracking system was used. This system is capable of tracking objects at 120 fps with a resolution of +/-0.1mm [29]. To track the human, a body suit was used that has retro-reflective markers allowing every joint on the human to be tracked. The motion tracking software by OptiTrack allows for a human pose estimation when it identifies the marker skeleton and will allow one to access the skeleton pose data through its SDK. There is a difference between the Optitrack skeleton and the pose estimate skeleton as the reported points are not in the same location. To simplify the estimation for accuracy, the outliers were discarded. The joints in each pose are seen in Figure 4.2.

All the data from the radar, pose-estimation and OptiTrack were merged into ROS to visualize in a single frame of reference and to easily view the results. ROS published all the data every 30Hz.

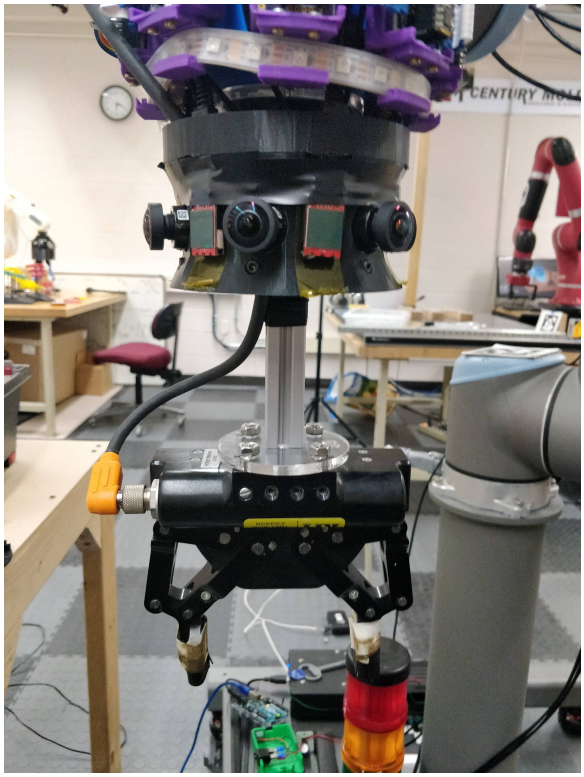
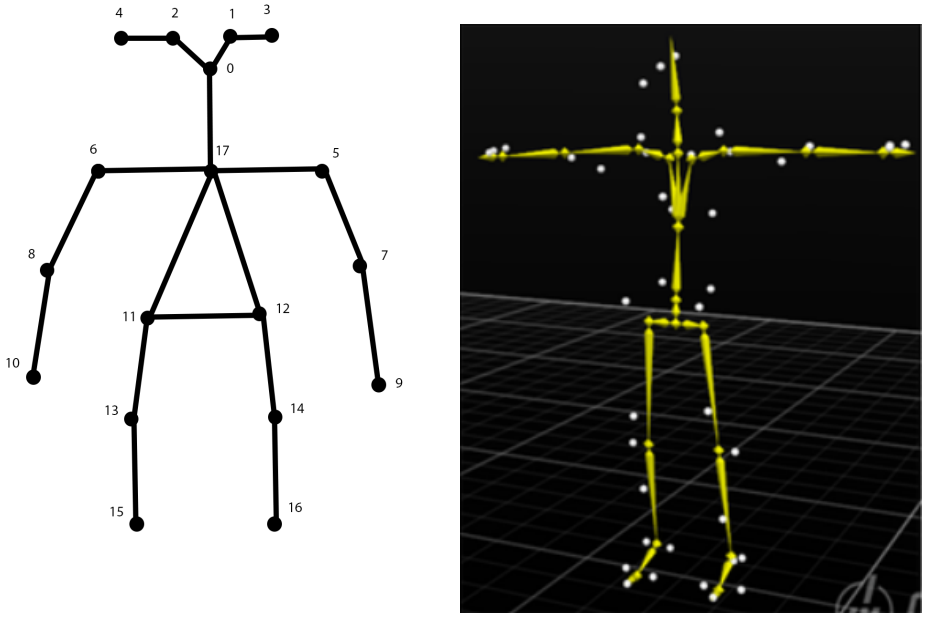


Figure 4.1: The sensor mounted on the UR-10 with a gripper attachment



(a) (b)

Figure 4.2: The pose data from (a) TrTPose vs (b) OptiTrack

4.1 Static testing conditions

The first test is to understand how well the radar camera system performs when both the human and the robot are stationary. These will be difficult for the sensor as the radar is configured to be sensitive to velocity changes and when there are no moving objects in the field of view, it will assume the object is background noise. Both the radar sensor and the pose estimation algorithm rely on movement in the workspace to update its estimation. To show this, the human was still in the workspace as the radar detections were monitored, as seen in Figure 4.3. However stray noise from the environment in the signal threshold causes detections not related to the human. This is shown to be a contributing factor to a minimum number of detections in the robot's workspace. In the static tests it was eight. As the human tries to stay still for the duration of the experiment, there are velocity spikes become evident ranging in 15cm/s velocities. These signals could be tied to breathing rate or heart rate of the human but currently this has not been tested.

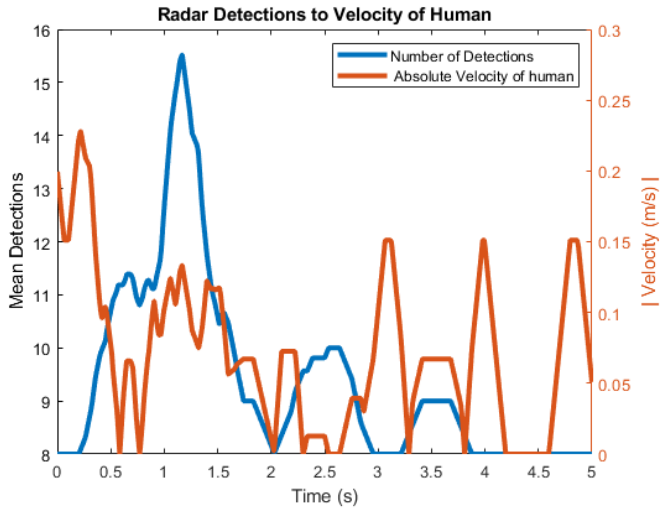
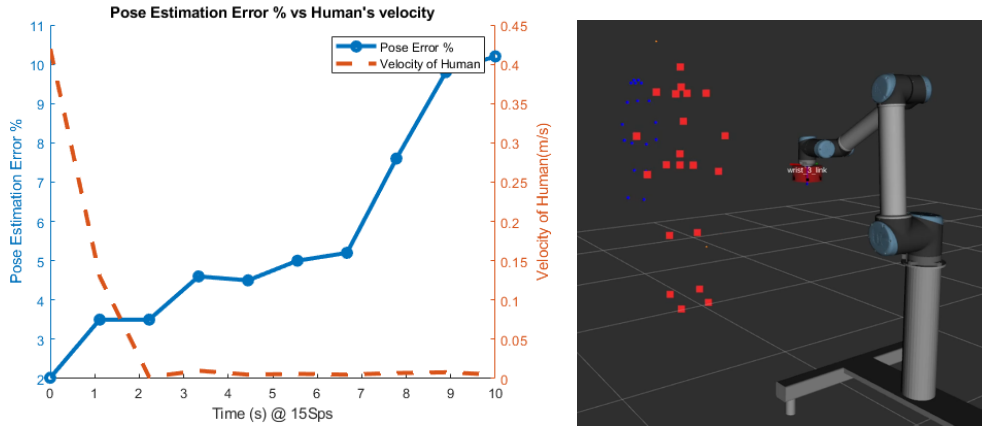


Figure 4.3: This shows as the velocity of the human comes to a standstill, there still are radar detections. Sps (Samples per second)

When observing the data, while the human moves, there was an at least -0.20m (20cm) consistent deviation between the radar system's pose estimate and the OptiTrack system. When the data captured was replayed though Rviz, ROS's 3D visualizer, it was apparent that this was not an issue with the sensor but a limitation of the system altogether. When the OptiTrack provides a skeleton pose estimate, it uses the center of the human body for this. The radar system on the other hand can only detect the front surface of the human closest to the robot and as the pose estimate fits to this, it will be off by 0.2m. The pose error was calculated by subtracting the joint position estimate from the Optitrack, adding the 0.2m offset and averaging the error.

For the 3D pose estimate, because it uses the last known position of the human, when the radar detections disappear, the estimate starts to develop a chaotic behavior. As the estimation is fitting the pose data onto the radar data, any stray radar detections will quickly move the pose estimate to the stray detections and subsequently increase the error of the pose estimate. This is seen in Figure 4.4 a.

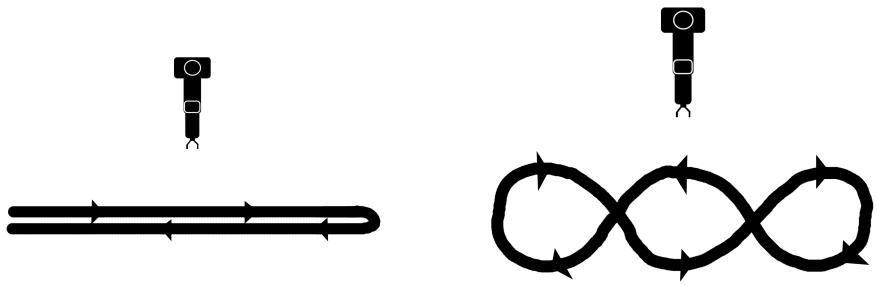


(a) (b)

Figure 4.4: (a) As the human slows down, the pose error increases significantly (b) Shows the pose estimate (blue) behind the human's actual pose (red) .

4.2 Human Moving, Robot Stationary

In situations where the human is not moving, there is a significant improvement in the detection and tracking of the human. To understand this better, three tests were designed to show how the movement around the workspace affects the accuracy of the tracking. One test requires the human to walk in a straight line at a constant velocity parallel to the robot and return on the same trajectory, shown in Figure 4.5a. This will test for tracking ability in a predictable manner. Next a more dynamic double figure-eight, seen in Figure 4.5b, was performed which includes movements toward and away from the robot and sensor system. The pseudo-random movement will show if the direction of movement has any affect on the tracking. Both of these tests are designed to take 10 seconds per lap which was maintained as best as possible.



(a) (b)

Figure 4.5: The tests (a) Straight Test vs (b) Double Figure-8 with respect to the robot

4.2.1 Straight Test

When looking at the stationary test, there is an offset that needs to be applied when comparing the pose from the OptiTrack to the pose estimation from the radar-camera system. This

offset should not affect the velocity of the human but because the velocity measurement is calculated radially, there is a slight difference between the measurement of the human and what can be calculated from the OptiTrack data. This effect gets more noticeable when the human is closer to the sensor than when further away as the estimate is greater than the OptiTrack. This was most noticeable with the double figure eight movement. On average the velocity estimate was within 5% deviation, however, the radar and the OptiTrack could correctly detect when the human had stopped moving without any error.

The straight line path was another difficult task for the pose estimation of the human as the pose had to be determined with the narrowest profile of the human. Unfortunately when the human becomes the narrowest directly in front of the camera there is no pose information however there is still radar data. The radar system on the other hand was able to keep track of the human throughout the test without much losses. This straight line showed an increase to the number of points per pointcloud frame to 140 from 20 when the human was stationary. Not only was the pose detection much more stable but visually, in ROS, one can visually outline the human, its legs and hands.The pose estimation had a 30% loss in frames but was able to be within 10cm of OptiTrack.

Figure 4.6 shows the frames lost as the human walks by the robot in the straight line test. The biggest deviations are shown clearly when the number of lost frames over time jumps. These jumps correlate to when the human walks in front of the camera as It finds difficulty detecting the pose. There are two jumps as one for each direction of travel. Compared to the camera's, throughout this test, the radar system was able to track the human. To calculate its error, the OptiTrack pose locations were used to see how well it could have

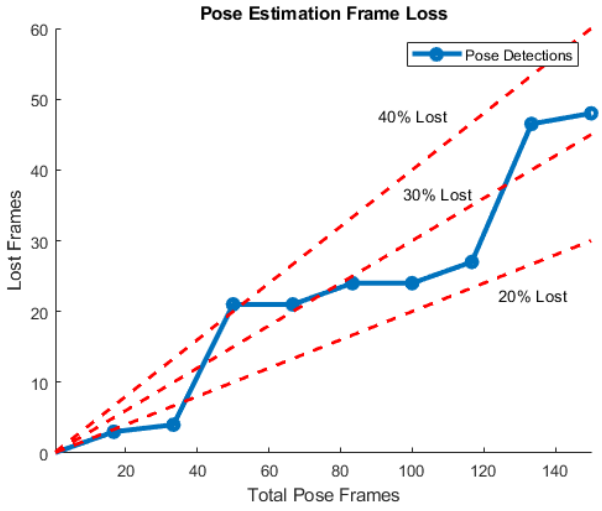


Figure 4.6: The frame losses as the human walks by the robot in the straight line test.

performed without the pose from the cameras. The OptiTrack pose locations were passed into the same kNN, with *K=3*, that estimated 3D pose but with a tree constructed from the full stitched radar pointcloud. The average of the points were recorded and compared to the OptiTrack pose.

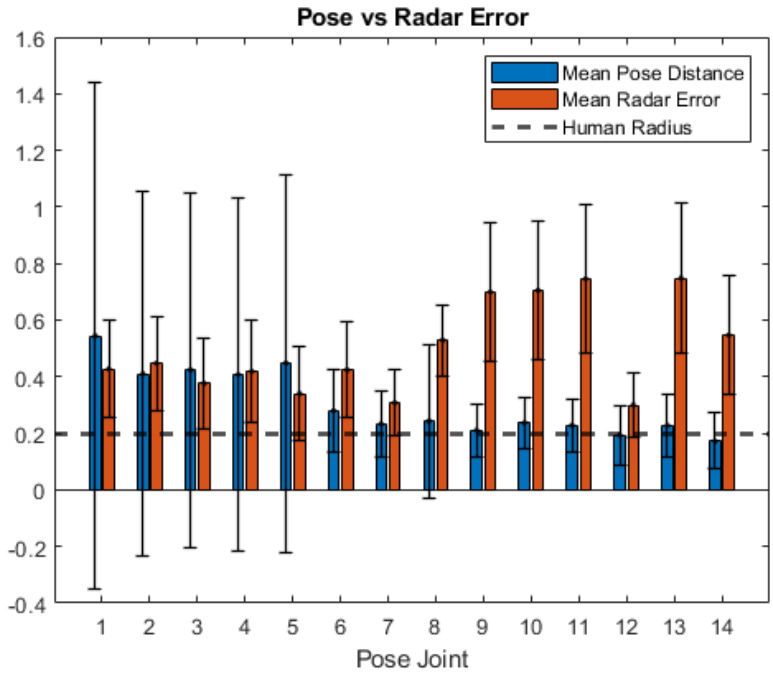


Figure 4.7: The comparison between the radar's accuracy and the human pose estimation compared to OptiTrack for the straight line test.

In the straight line test, the radar's estimation of where the human was, was better than the pose estimation for certain joints. For certain joints, the radar could not detect the points. This is because as the human walks by, it only detects the closest points on the human which can be far from the other locations. On the other hand, the pose estimation did well trying to fit the pose on to the human, however for certain joints the standard deviations was high. These inaccuracies were due to the missed frames from the pose detection. The mean and standard deviations were averaged for all the pose joints and are shown in Table 4.1. The radar was more precise in its measurements but was less accurate than the pose estimation.

Table 4.1 The mean and standard deviation for the pose detection for all joints for the straight line test.

		Pose	**Radar**
Mean	min	0.1717	0.2997
	avg	**0.3038**	**0.5008**
	max	0.5440	0.7483
Std	min	0.0918	0.0918
	avg	**0.3268**	**0.1855**
	max	0.8940	0.2658

4.2.2 Double Figure-Eight Test

During the figure eight test in Figure 4.8, the results from the error of the pose and the radar system looks to be identical. The radar stays consistent for each joint as the human is the same in the test as well as the path of the human should not affect the accuracy. On of the biggest differences is the number of dropped frames for the pose detection which changed from 33% down to 15%. There is still frame drops when the human's side profile is visible to the robot. Also for the pose detection, the standard deviation of error increases for every joint as seen in Table 4.2. This shows that the more random nature of the human's movement causes increases errors in each of the joints.

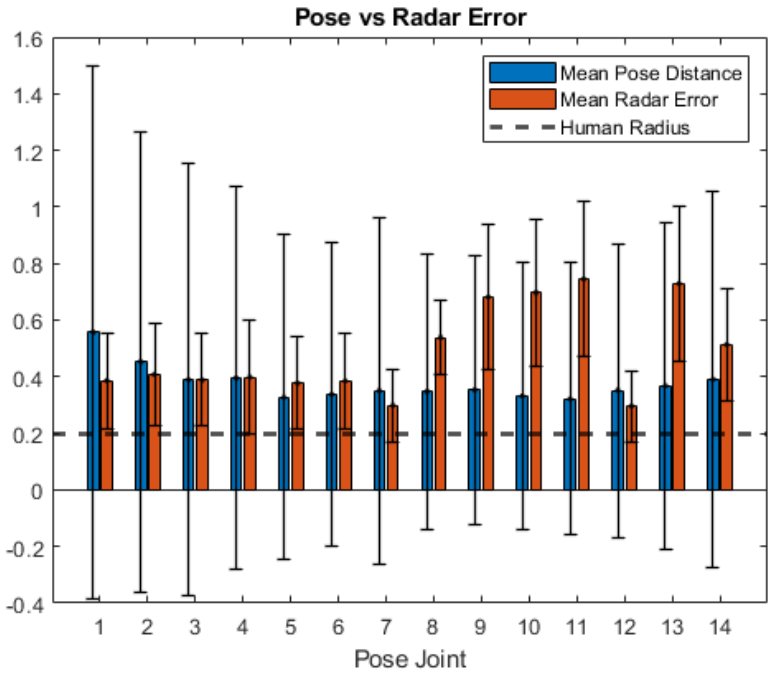


Figure 4.8: The comparison between the radar's accuracy and the human pose estimation compared to OptiTrack for the double figure-eight test.

Table 4.2 The mean and standard deviation for the pose detection for all joints for the double figure-eight test.

		Pose	**Radar**
Mean	min	0.3208	0.2953
	avg	**0.3765**	**0.4884**
	max	0.5564	0.7443
Std	min	0.4730	0.1254
	avg	**0.6144**	**0.1920**
	max	0.9426	0.2750

4.2.3 Velocity Measurements

One of the benefits to having a radar module is the velocity calculation does not depend on time between samples as the radar module directly estimates velocity. Also straight from the configuration, velocity measurements are in SI units (m/s) which is consistent with how the OptiTrack's position estimated are calculated. To understand the velocity of each joint in the pose, the closest three detections to each joint from the radar are averaged to get the velocity of the joint. Because the radar measures velocity toward and away from the robot, understanding the radar's estimate for the human's true velocity is more difficult. An depiction of the velocity measurements are shown in Figure 4.9.

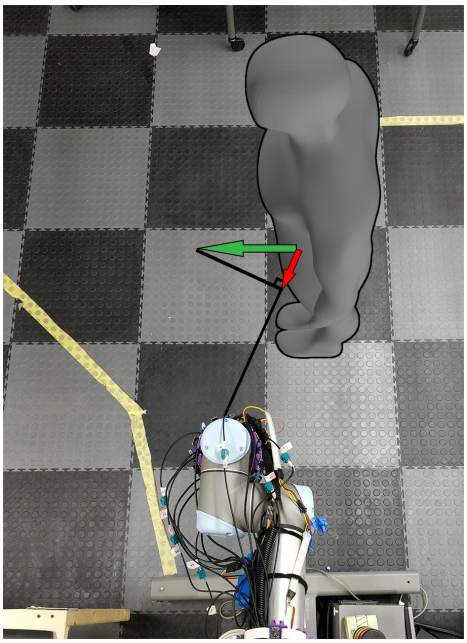


Figure 4.9: The green arrow is the actual human velocity and the red arrow is the velocity perceived by the radar

To compensate for this discrepancy, the OptiTrack's velocity was calculated with respect to the sensor on the robot. This allowed for direct comparison in the error.

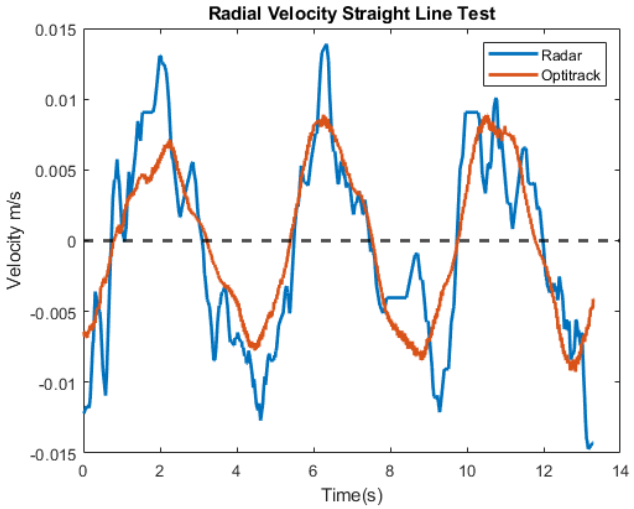


Figure 4.10: The velocity of the humans as it performs the straight test. The oscillation shows the as the human comes closer to the robot and moves further away.

In the straight line test, the human walked at a consistent velocity for each pass. Looking at Figure 4.10, it is clearly shown how the human walks as the velocity oscillates as the human gets closer to the robot and further away. In fact, each period of the velocity determines one pass as the human moves from left to right or vice-versa in Figure 4.5. The root mean-squared error (RMSE) is 0.0035 m/s between the radar and the OptiTrack's velocity estimate. Also there is noticeable noise on the straight line test which could be as the human walked, the arms were swinging which gave irregular velocities.

In the double figure-eight test in Figure 4.11, the more chaotic velocity movements outline the complicated nature of the motion. In this test, the magnitude of the velocity is shown to be less than from the straight line test. Here, the OptiTrack's velocity estimate is greater than the radar's measurement. What is noticeable is how the zero-crossing for the graph lines up showing the accuracy of the radar's measurements. Lastly, the RMSE is 0.0034 m/s between the radar and the OptiTrack's velocity estimate.

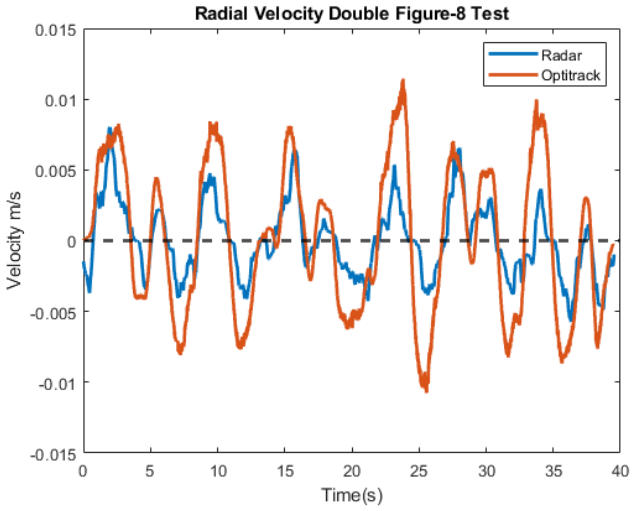


Figure 4.11: The velocity of the humans as it performs the double figure-eight test

4.3 Human Moving, Robot Moving

The last scenario is a fully dynamic test where both the robot and the human move around. For the robot, a simplified simple pick and place movement was programmed onto the UR10 and ran at three separate speeds: 20, 60 and 80 degrees per second. For the human, the same tests were performed where the human was stationary, moved in a straight line and moved in a double figure eight.

Immediately what is noticeable is the radar sensor detects the environment much more as in Figure 4.12. This is separate from getting a doppler effect in the transmission and receiving of the radar chirp. The increase in detections are due to valid readings in the real world as the radar moves around. One way to mitigate this is to increase the threshold for CFAR in the radar, however, this can not be changed on the fly and needs the radar module to be rebooted. The current approach for human detection does not work as well because the additional environmental data becomes addition noise to filter. Using the kNN's approach

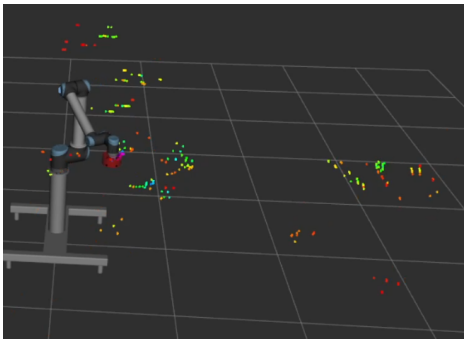


Figure 4.12: The background noise as the robot moves the camera-radar sensor

allows a simple way to fit the pose onto the radar data but the radar data can not understand what is human and what is not. As the data is squished and merged, the closest point no longer belongs to the human and this brings additional error into fitting the pose on the human. The following tests were all performed using the same approach as for the stationary tests.

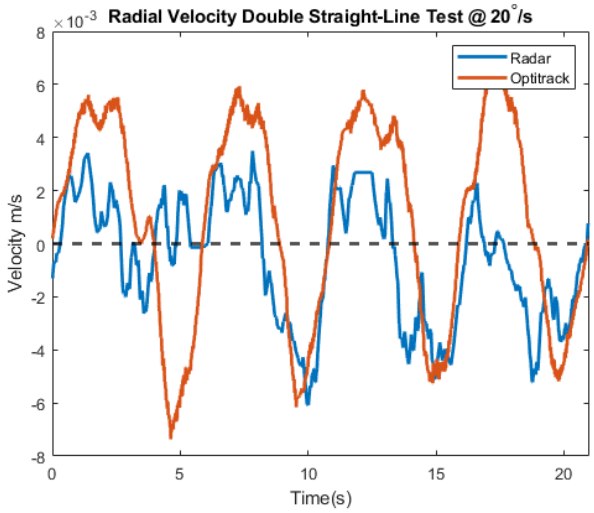


Figure 4.13: Velocity from the radar measurements and OptiTrack system with the robot moving at 20°/s

When the robot moves at 20 degrees per second, the current approach for pose detection is still feasible, but with an increase in pose error and velocity error, shown in Figure 4.13. The faster the robot moves, the larger the error becomes and the harder it is to detect

the human. At 60 degrees per second, the human is detectable but the pose location has an increased error. At 80 degrees per second, the detection of the human starts to fail as the number of frames the pose was detected decreases significantly. This signifies the capabilities of the radar system and the camera system separately. As the radar has difficulty differentiating what is human and what is the environment, the camera does as well.

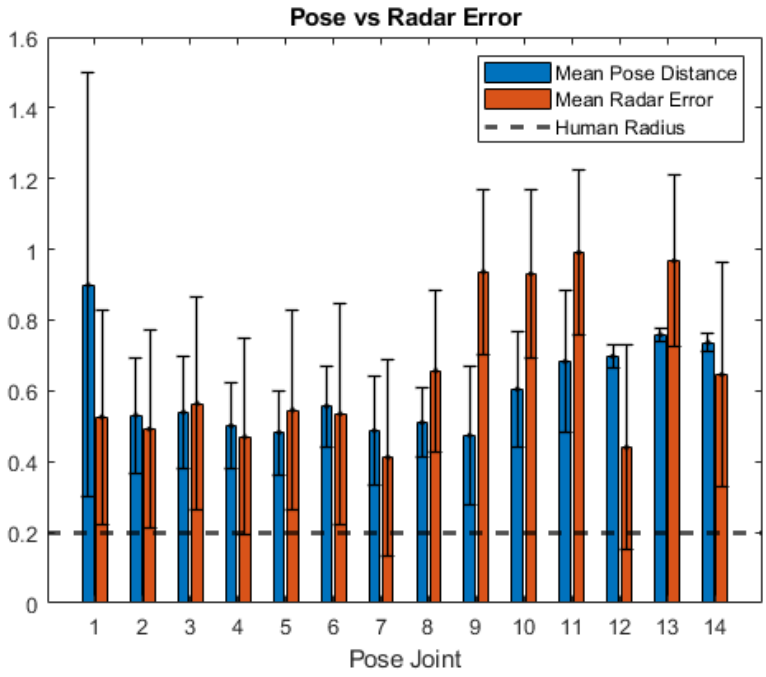


Figure 4.14: The Pose and Radar error as the robot moves at 20°/s

Table 4.3: Data from Figure 4.14

		Pose	**Radar**
Mean	min	0.4738	0.4111
	avg	**0.6040**	**0.6505**
	max	0.8993	0.9903
Std	min	0.0176	0.2285
	avg	**0.1544**	**0.2723**
	max	0.5991	0.3178

While the robot moves at 20 degrees per second, the average pose and radar error increase together, shown in Figure 4.14 and Table 4.3. This is more than when compared to the stationary tests in Table 4.1 and 4.2. For a pose error of 0.6m, humans are able to get close to the robot with it understanding where the human might be. This paired with the velocity estimates in Figure 4.13, reaffirms that the human is still trackable at slow robot speeds.

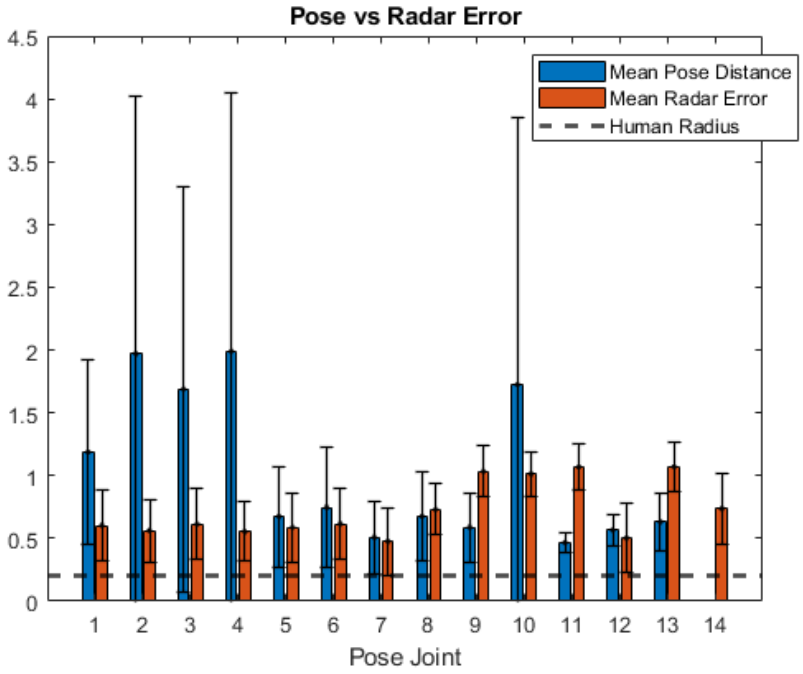


Figure 4.15: The Pose and Radar error as the robot moves at 60°/s

Table 4.4: Data from Figure 4.15

		Pose	**Radar**
Mean	min	0.4665	0.4754
	avg	**1.0318**	**0.7254**
	max	1.9900	1.0694
Std	min	0.0767	0.1806
	avg	**0.8322**	**0.2424**
	max	2.1317	0.2872

When analyzing the pose detection at 60 degrees per second, in Figure 4.15 and Table 4.4, there is a considerable amount of error in the pose, however the radar can track the human to a similar accuracy as when the robot moved at 20 degrees per second. For the first four joints, there is an increase to the average error in the 3D pose estimation where it surpasses the radar's error.

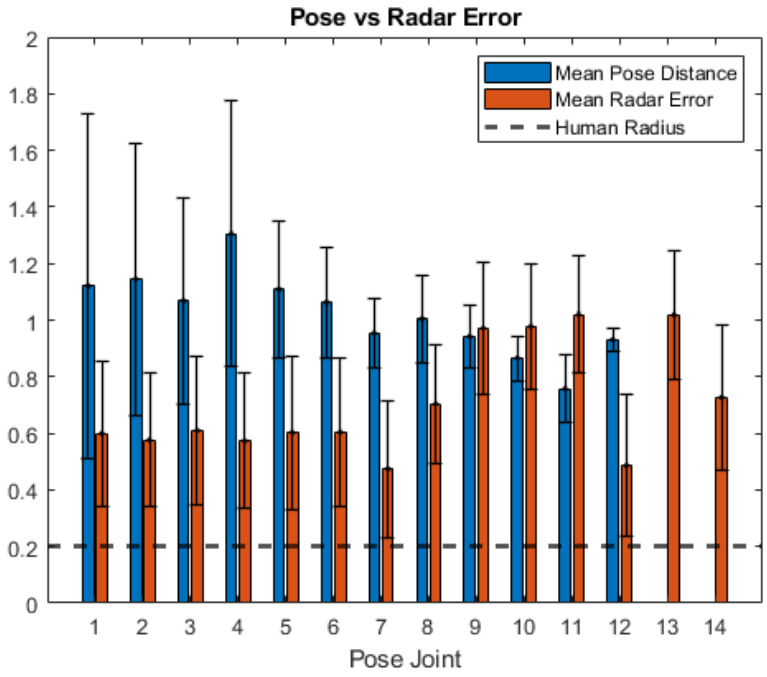


Figure 4.16: The Pose and Radar error as the robot moves at 80°/s

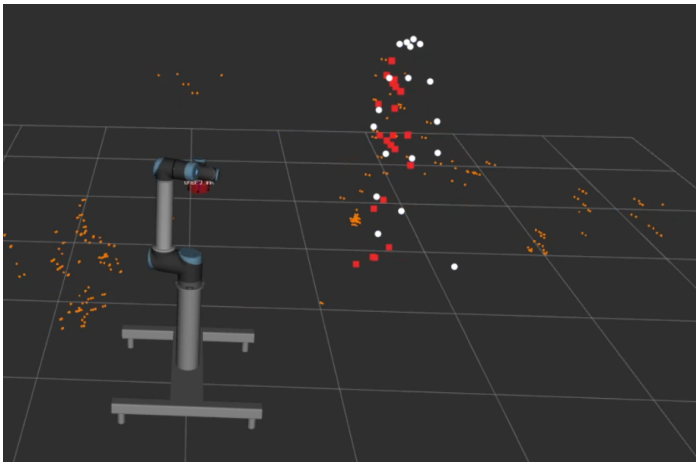
Table 4.5: Data from Figure 4.16

		Pose	**Radar**
Mean	min	0.7578	0.4731
	avg	**1.0218**	**0.7099**
	max	1.3050	1.0200
Std	min	0.0431	0.2082
	avg	**0.2490**	**0.2411**
	max	0.6094	0.2716

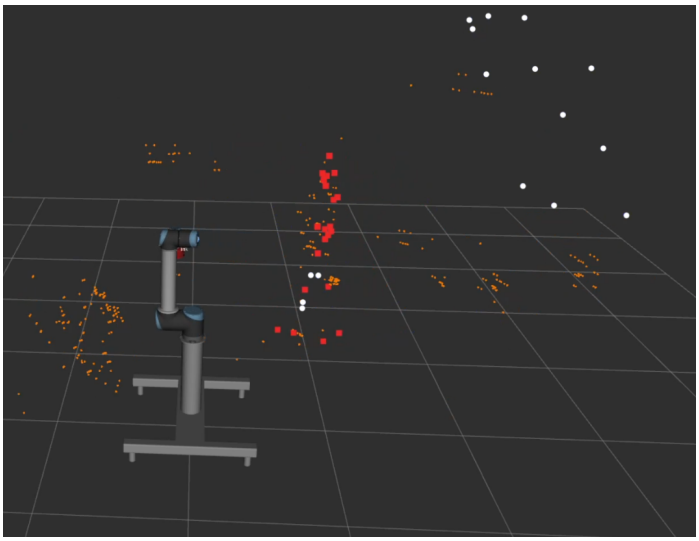
Lastly, understanding the pose detection as the robot moves at 80 degrees per second

shows that there are significant errors in the 3D pose estimation. In Figure 4.16 and Table 4.5, there is a increase in magnitude for the pose error for all joints, however, the radar error stays around the ballpark of the error when the robot moved at 60 and 20 degrees per second. One noticeable difference between the last two operating speeds of the robot is at 60 degrees per second, the camera is able to detect the pose more often but incorrectly place the pose, where as at 80 degrees per second the robot the camera is unable to place the pose altogether. This would explain the decrease in standard deviation between the two operating speeds while the mean increases. Counting the frames lost shows that 55% of the frames were lost at 80 degrees per second where as only 38% were lost at 60 degrees per second.

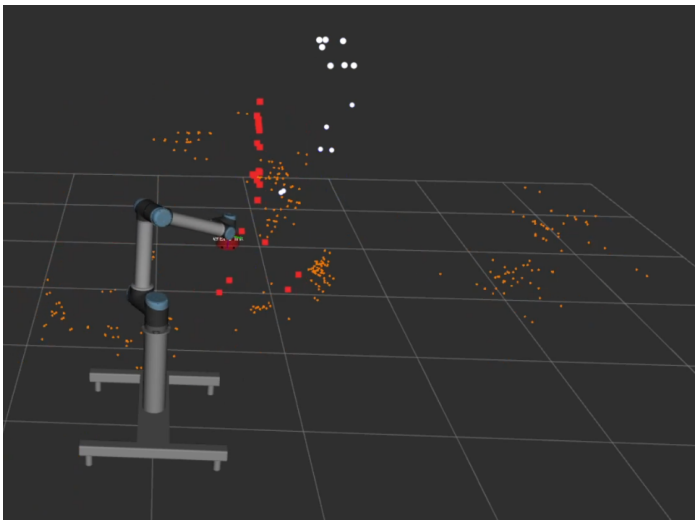
As seen, the camera starts to fail as the speed increases while the radar is able to detect the human thought the motion of the tests. Visually, through ROS, the radar seems to pick up the human much more even as the human changes it's velocity, shown in Figure 4.17. This shows that an ideal 3D pose estimation using a camera and radar system should rely on camera estimates for lower robot speeds and radar estimates at higher speeds. During the dynamic tests, blind spots were decreased significantly as the pose is able to be detected from multiple angles as the robot moves around. The data suggests that this sensor system can be used for many purposes as its accuracy and precision in detecting a 3D pose of a human is on of its strengths.



(a)



(b)



(c)

Figure 4.17: Pose 3D estimate viewed with ROS as the robot moves at (a) 20°/s (b) 60°/s (c) 80°/s

Chapter 5

Conclusions

The designed radar-camera system is an effective tool to assist with the safety in human-robot collaborative applications. It brings forth a more comprehensive methodology to understand where the human is in the workspace, along with how fast it moves. This sensor system was extensively tested in static and dynamic scenarios for both the human and the robot and it performed satisfactory in all. Its use of industry ready sensors allows it to be easily constructed for deployment in industry. It is able to be mounted to a wide variety of robots including robot arms and mobile robots. Its compact nature and light weight allows for easy implementation in existing robots without much modification. And the data it receives allows for extensive research and better understanding of how to integrate humans in a robot's workspace. This sensor system is one of a kind and will change human robot collaboration altogether.

Chapter 6

Future Work

Because this sensor system has much to offer, in terms of capabilities and data, there is additional research that can be performed in a variety of fields including, but not limited to, autonomous driving, mobile robot exploration, human behavior identification, and human task recognition. The idea that this sensor can be treated as two independent 360 degree sensors that can be interfaced separately, allows for even more possibilities. Having a wide angle 360 degree camera can allow for the development of new algorithms in image processing. Having a 360 degree radar can help with high accuracy occupancy detection.

Currently, the sensor system can be improved further in terms of efficiency and in dynamic workloads (where the human and the robot are moving). One can perform post processing of the radar data before stitching it: filtering out environment noise and keeping just the human information. Also one can incorporate the signal to noise ratio data in processing to increase the accuracy of the detections.

Currently all the radar modules are tuned for identifying where the human is but because there is overlap in the sensor readings, every other sensor can be serving a different purpose. These radar modules can be tuned to read bio signals of a human from breathing rate to heart rate. If half of these sensors were to be used, possibly this radar sensor could

understand where the human is and how the human feels in the robotic workspace. This would give more information to the robot allowing it adapt its motion to make the human feel safer in the workspace.

For this sensor system, one can take the raw radar data and camera data and train a machine learning network to process human pose. This will significantly reduce the processing time and increase the framerate of the sensors. The model can also be tuned to adapting for movements of the sensor system, which the current process struggled to perform well in. Once this is developed, it can be ported to run on the existing Nvidia Xavier making full use of the tensor cores and portable nature of the system.

Lastly, this radar-camera on-robot system doesn't need to be the only sensor in the workspace of the robot. If integrated with Lidar systems or external depth cameras, the precision of the human tracking will significantly increase. A dataset can be created implementing as many sensors as possible to give researchers as much data as they need to find solutions to the next problems. And one day humans will be able to work right next to robots, assisting us and making our lives easier.

Bibliography

[1] ISO/TS 15066:2016, "Robots and robotic devices - collaborative robots," International Organization for Standardization, Geneva, CH, Standard, feb 2016.

[2] E. Kim, R. Kirschner, Y. Yamada, and S. Okamoto, "Estimating probability of human hand intrusion for speed and separation monitoring using interference theory," *Robotics and Computer-Integrated Manufacturing*, vol. 61, no. June 2019, p. 101819, feb 2020.

[3] S. Kumar, S. Arora, and F. Sahin, "Speed and Separation Monitoring using On-Robot Time-of-Flight Laser-ranging Sensor Arrays," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, vol. 2019-Augus. IEEE, aug 2019, pp. 1684–1691.

[4] C. Byner, B. Matthias, and H. Ding, "Dynamic speed and separation monitoring for collaborative robot applications – Concepts and performance," *Robotics and Computer-Integrated Manufacturing*, vol. 58, no. September 2018, pp. 239–252, aug 2019.

[5] ISO 13855:2010, "Safety of machinery — Positioning of safeguards with respect to the approach speeds of parts of the human body," International Organization for Standardization, Geneva, CH, Standard, may 2010.

[6] J. A. Marvel and R. Norcross, "Implementing speed and separation monitoring in collaborative robot workcells," *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 144–155, apr 2017.

[7] M. J. Rosenstrauch, T. J. Pannen, and J. Krüger, "Human robot collaboration - using kinect v2 for ISO/TS 15066 speed and separation monitoring," *Procedia CIRP*, vol. 76, pp. 183–186, 2018.

[8] C. P. C. Andres, J. P. L. Hernandez, L. T. Baldelomar, C. D. F. Martin, J. P. S. Cantor, J. P. Poblete, J. D. Raca, and R. R. P. Vicerra, "Tri-Modal Speed and Separation Monitoring Technique using Static-Dynamic Danger Field Implementation," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information*

Technology,Communication and Control, Environment and Management (HNICEM), vol. i. IEEE, nov 2018, pp. 1–6.

[9] F. Vicentini, M. Giussani, and L. M. Tosatti, "Trajectory-dependent safe distances in human-robot interaction," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, sep 2014, pp. 1–4.

[10] M. Terreran, E. Lamon, S. Michieletto, and E. Pagello, "Low-cost Scalable People Tracking System for Human-Robot Collaboration in Industrial Environment," *Procedia Manufacturing*, vol. 51, pp. 116–124, 2020.

[11] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for 3D LiDAR-based human detection: experimental analysis of point cloud clustering and classification methods," *Autonomous Robots*, vol. 44, no. 2, pp. 147–164, jan 2020.

[12] P. Svarny, Z. Straka, and M. Hoffmann, "Toward safe separation distance monitoring from RGB-D sensors in human-robot interaction," *arXiv*, oct 2018.

[13] X. V. Wang, X. Zhang, Y. Yang, and L. Wang, "A Human-Robot Collaboration System towards High Accuracy," *Procedia CIRP*, vol. 93, pp. 1085–1090, 2020.

[14] O. Mazhar, B. Navarro, S. Ramdani, R. Passama, and A. Cherubini, "A real-time human-robot interaction framework with robust background invariant hand gesture detection," *Robotics and Computer-Integrated Manufacturing*, vol. 60, no. June 2018, pp. 34–48, dec 2019.

[15] B. P. Jeppesen, N. Roy, L. Moro, and F. Baronti, "An FPGA-based controller for collaborative robotics," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. IEEE, jun 2017, pp. 1067–1072.

[16] L. Balan and G. Bone, "Real-time 3D Collision Avoidance Method for Safe Human and Robot Coexistence," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2006, pp. 276–282.

[17] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for human classification in 3D LiDAR-based tracking," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2017-Septe. IEEE, sep 2017, pp. 864–871.

[18] U. B. Himmelsbach, T. M. Wendt, and M. Lai, "Towards Safe Speed and Separation Monitoring in Human-Robot Collaboration with 3D-Time-of-Flight Cameras," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*, vol. 2018-Janua. IEEE, jan 2018, pp. 197–200.

[19] E. Sugawara and H. Nikaido, "Properties of AdeABC and AdeIJK efflux systems of Acinetobacter baumannii compared with those of the AcrAB-TolC system of Escherichia coli." *Antimicrobial agents and chemotherapy*, vol. 58, no. 12, pp. 7250–7, dec 2014.

[20] C. Stetco, B. Ubezio, S. Muhlbacher-Karrer, and H. Zangl, "Radar Sensors in Collaborative Robotics: Fast Simulation and Experimental Validation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2020, pp. 10 452–10 458.

[21] SangHyun Chang, Ta-Shun Chu, J. Roderick, Chenliang Du, T. Mercer, J. W. Burdick, and H. Hashemi, "UWB human detection radar system: A RF CMOS chip and algorithm integrated sensor," in *2011 IEEE International Conference on Ultra-Wideband (ICUWB)*. IEEE, sep 2011, pp. 355–359.

[22] M. Geiger and C. Waldschmidt, "160-GHz Radar Proximity Sensor With Distributed and Flexible Antennas for Collaborative Robots," *IEEE Access*, vol. 7, pp. 14 977–14 984, 2019.

[23] M. Zlatanski, P. Sommer, F. Zurfluh, S. G. Zadeh, A. Faraone, and N. Perera, "Machine Perception Platform for Safe Human-Robot Collaboration," in *2019 IEEE SENSORS*, vol. 2019-Octob. IEEE, oct 2019, pp. 1–4.

[24] M. Zlatanski, P. Sommer, F. Zurfluh, and G. L. Madonna, "Radar Sensor for Fenceless Machine Guarding and Collaborative Robotics," in *2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE, aug 2018, pp. 19–25.

[25] M. J. Rosenstrauch, T. J. Pannen, and J. Krüger, "Human robot collaboration - using kinect v2 for ISO/TS 15066 speed and separation monitoring," *Procedia CIRP*, vol. 76, pp. 183–186, 2018.

[26] B. Shu, G. Sziebig, and S. Pieska, "Human-Robot Collaboration: Task Sharing Through Virtual Reality," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, oct 2018, pp. 6040–6044.

[27] V. Kuts, T. Otto, T. Tähemaa, K. Bukhari, and T. Pataraia, "Adaptive Industrial Robots Using Machine Vision," in *Volume 2: Advanced Manufacturing*. American Society of Mechanical Engineers, nov 2018, pp. 1–8.

[28] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[29] "Flex 13 datasheet - an affordable motion capture camera."