Object Detection and Identification with Sensor Fusion

#18
Client: Michael Olson - Danfoss
Advisor: Dr. Wang

Tucker Creger - Project Manager
Kellen O'Connor - Software Developer
Eric Bishop - Software Developer
Mitch Hagar - Hardware Developer
Clayton White - Hardware Developer
Nihaal Sitaraman - Hardware Developer

sddec18@iastate.edu
http://sddec18-18.sd.ece.iastate.edu/

Table of Contents

List of Definitions

Radar: Radio Detection and Ranging

ESR: Electronically Scanning Radar

LiDAR: Light Detection and Ranging

SoC: System On a Chip

LCD: Liquid Crystal Display

PEP: Python Enhancement Proposal

IEEE: Institute of Electrical and Electronics Engineers

RGB: Red, Green, Blue

SSD: Single Shot Detector

CNN: Convolutional Neural Network

FRCNN: Faster Region-based Convolutional Neural Network

CAN: Controller Area Network

HDMI: High Definition Multimedia Interface

API: Application Program Interface

DRC: Design Rule Check

PCB: Printed Circuit Board

1 Introductory Material

1.1 Acknowledgment

The project team would like to thank Michael Olson and Radoslaw Kornicki from Danfoss for their support on this project. We would also like to thank Dr. Zhengdao Wang for advising our team. Finally, we would like to acknowledge Austin Chesmore for his input on this project.

1.2 Problem Statement

As the agriculture and construction industries require autonomous and driver alert solutions for increased safety and productivity, the need to sense objects in the equipment path and alert the operator increases. Examples include detecting people, animals, vehicles, and other large objects. There are many solutions on the market today using cameras and LIDAR. These solutions have limitations in weather and low light conditions. We were tasked with creating a system using radar to eliminate these limitations.

Our project consists of two main components: the implementation of a radar system and the integration of a deep learning model. The radar system will allow us to detect objects in many different conditions. The deep learning model will use a camera to identify objects in the equipment's path. This will allow for a notification to the equipment operator of objects in the vehicle's path.

1.3 Operating Environment

The operating environment for the system will be on agricultural and construction equipment. This will require the system to be able to withstand water and dust from the operating environment and the vibrations associated with operation.

1.4 Intended Users and Intended Uses

The intended use is for certain agricultural vehicles and construction equipment that are a key area for our client. It will be used as an operator aid to increase safety. A simple example is putting this system on a bulldozer to alert the operator of objects in the bulldozer's path.

1.5 Assumptions and Limitations

Assumptions:

- The operating conditions for the equipment will be normal and not abusive.
- The system will be mounted in an area that is protected from impact.
- The system-on-a-chip (SoC) and radar will be able to operate in a rugged environment.

Limitations:

- The system will only operate up to 5 mph. This will cover a range of agricultural and construction equipment.
- The system will not be 100% immune to sensor blockage by dust, dirt, and snow.

1.6 Expected End Product and Other Deliverables

For this project, our deliverables are a whole system including a radar module and camera working with a deep learning model running on a SoC to perform object localization and classification. The system will notify the vehicle operator via a display. The delivery date is mid-December 2018. This system will be used for a demo for our client.

Other deliverables include proposals regarding our radar and SoC selection, a final report on our design, deep learning platforms that are most suited for use in mobile applications, and the feasibility of implementing radar in construction and agricultural applications. The delivery date of the report is December 2018. The delivery date of proposals to purchase the SoC and radar will be February 2018 and March 2018, respectively.

An itemized list of our deliverables is included below:

1. NVIDIA Jetson TX2 with a display (the display is not property of Danfoss), camera, and PRECO PreView Sentry.
2. Pretrained deep learning model for object detection relevant to an agricultural/construction vehicle.
3. Software written in Python that utilizes attached components and deep learning model to display object information on a display.

2 Project Design

2.1 Project Objective

Our objective is to evaluate various radar technologies for Danfoss and through a combination of sensor fusion and deep learning, perform object detection and localization. By December 2018, we will have selected a radar option, a computing system adequate for a rough environment, selected a neural network and camera, and implemented it on a vehicle to alert an operator of the presence and location of unique objects via a display.

In order to provide value to Danfoss, we will also include in our final report an evaluation of various radar technologies, deep learning platforms, and computing systems to assist them in making a business decision when deciding to implement this technology in the future.

2.2 Functional Requirements

The functional requirements for the proposed design focus on robust detection and operation in agricultural and construction environments. A list of functional requirements is shown below.

1. The system shall have a range of 30 meters.
    a. Rationale: A range of 30 meters is required for early detection and identification. A machine traveling at 5 mph will cover 30 meters in under 13 seconds. This range allows for an object to be detected with sufficient time for action.
2. The system shall function on machines traveling at up to a speed of 5 mph or 2.2 m/s.
    a. Rationale: Most agriculture and construction equipment operates at at speeds around 5 mph.
3. The system shall have FOV of ±30°.
    a. Rationale: An FOV of ±30° is required to detect objects in the vehicle's path with sufficient time to stop.
4. The system shall have a processing speed of 10 frames per second.
    a. Rationale: The system needs to detect an object with sufficient time to react. A frame rate of 10 frames per second on a vehicle traveling approximately 5 mph or 2.2 m/s means the vehicle will travel no further than 0.3 m between each frame update.
5. The system shall detect objects greater than 0.4 m size.
    a. Rationale: A width of 0.4 m is the width of human shoulders. Detection of a human is, at minimum, required for safe operation of the system.
6. The system shall have a probability of missed detection less than 0.3.
    a. Rationale: A probability of 0.3 means that for each subsequent frame, the probability of missing an object multiple times approaches zero, which will yield a sufficiently short stopping distance.
7. The system shall have a probability of false alarm less than 0.3.
    a. Rationale: A false alarm, though undesirable, will be a safer alternative than a missed detection.
8. The system shall detect at least 3 classes of objects.

    a. Rationale: Our system should detect people, cars, and animals.
9. The system shall/should operate in the temperature range from -20 to 75 degrees Celsius.
    a. Rationale: This is a common operating range for automotive sensors.

2.3 Non-Functional Requirements

1. The system shall be weather resistant to water, dust, and shock.
    a. Rationale: Danfoss' target applications involve heavy machinery that works in tough environments.
2. The system shall run off of a 12V power supply.
    a. Rationale: This voltage is easily available from a selection of batteries with a range of capacities. It is also easily available on a heavy equipment chassis. A step-up converter or inverter is acceptable.
3. The system shall fit inside 1'x1'x1' space.
    a. Rationale: Space is limited on a vehicle, so our design must be compact enough to not obstruct operator view or regular vehicle operation.

2.4 Constraints Considerations

As part of the project, we must evaluate various radar options, deep learning platforms, object detection networks, and computing systems. Evaluation of these systems will be centered around the functional requirements, but the behavior of the full system cannot be known without implementing all combinations of each option.

Because this project will ultimately lead to a business decision from Danfoss, cost of the system must be considered. We will strive to minimize cost, but not at the expense of our functional requirements.

Our code must be well commented and accessible to the client. The team will use GitLab for version management of our software. This includes our neural network, data acquisition tools, and any low-level radar code.

Training data acquired during our project must be accessible to our client, yet secure. We will collaborate with Danfoss to ensure data collected (which may include imagery from their facilities) is secure to their internal standards.

For our code, group members will follow an agreed upon coding convention. Because we expect much of our code to be Python, we plan to follow PEP 8 - Style Guide for Python Code [7] when applicable. This is not a formal standard that is required, but rather an organizational structure to ensure readability.

Our system will also utilize the ISO 11783 and SAE J1939 Standards for Controller Area Network.

The IEEE Code of Ethics (IEEE) will help guide our project, and ensure that our work does not violate the health and safety of our members, equipment operators, or Danfoss employees. This is similar to number 1 in the IEEE code of ethics. We will ensure that any research performed is well documented and cited where necessary. This follows number 3 in the IEEE Code of Ethics

This code of ethics is applicable to our project because it may eventually be used to prevent injury, so ethical violations could indirectly cause harm eventually. Overall, our system does not violate any ethical considerations from IEEE, and we will take care to ensure this is the case for the duration of our project.

Beyond the aforementioned standards (IEEE Code of Ethics, PEP 8, and internal Danfoss standards), our project will not violate any ethical considerations. We believe that accurately reporting the system's capabilities is the most significant task we can do, which falls in line with the IEEE Code of Ethics.

2.5 Previous Work and Literature

Literature surrounding the use of deep learning with radar focuses on either close-range object classification, or improvement of synthetic aperture radar.

A study from the University of St. Andrews showcases how a short range radar can be used to differentiate between various objects [6]. This study is encouraging in that it shows how radar waves may be reflected in a unique way from different objects, but it does not show long-range applications, which is a significant shortcoming.

A research paper from Radar Conference [1] shows how deep learning can be used to improve the digital signal processing aspect of radar for synthetic aperture radar. This is beyond the scope of our project, as our project centers more around performing deep learning on RGB imagery. Also, our system must perform real-time detection, not reconstruct an image later.

Literature more relevant to our project is related to object detection on RGB imagery. Several methods have been published that detail balances between speed and accuracy of neural networks for object detection. Towards Data Science[8] describes several object detection methods using deep learning for us to evaluate. Of interest are Single-Shot Detector (SSD)[2] and Faster Region-based Convolutional Neural Network (FRCNN)[3]. These network topologies take different approaches to detect objects. FRCNN proposes regions where an object might be, and evaluates each region to determine where an object lies in an image. SSD analyzes an image with fixed bounding boxes around what the CNN determines are relevant features to determine where an object is. For our purposes, SSD may be a better option to explore due to its improved speed compared to FRCNN, but at the expense of accuracy. Both network topologies deal with image resolutions beyond what our radar is likely capable of producing, so we may need to explore techniques for interpolation.

2.6 Design Analysis and Previous Designs

2.6.1 Design Concepts

Our design focuses on two possible solutions to perform object localization on agricultural and construction vehicles. The first solution is to perform localization and classification of objects using only a radar system, while the second solution is to perform localization with radar and classification with a camera system. The second solution is the solution we have pursued throughout the second semester due to a budget decision from our client.

Research to date indicates that Vayyar's development kit may provide feature-rich data suitable for deep learning, though it was too cost prohibitive to pursue. Vayyar, compared to other

suppliers, was an attractive option due to their API compatible with Ubuntu and Python, which makes it usable for deep learning with Keras. Because their radar system uses an array of antennas at 20 GHz (max frequency), it is possible to capture a relatively high resolution radar image with a range of 30m and an update frequency of approximately 6 Hz. This does not satisfy our initial design constraints of 15 fps or 60m range. However, given that the constraints were created with only the fastest agricultural vehicles in mind, we believe that the slower update rate and shorter range would have been sufficient for vehicles such as skid steers, tractors, combines, graders, etc.

Vayyar was also attractive compared to other suppliers due to their high resolution output. Other suppliers, such as Omniradar, only utilize less antennas in their arrays, yielding data that is not feature-rich and is therefore unsuitable for deep learning. Another supplier, Delphi, makes a system suitable for use with a camera for classification, but gives an output only in the form of a CAN message, making it unsuitable by itself for use with deep learning. However, because we can use a camera as a backup classifier, this is the option we decided on.

Our second option is to use a camera system to perform classification with the radar performing localization. For this proposed solution, we originally planned to utilize the Delphi system to determine where up to 64 objects are up to 60m away at ±45°. The CAN message would be sent to our NVIDIA TX2 development board and mapped to an RGB image, on which we would perform classification. At $6175, the Delphi system is more affordable than the Vayyar kit. We later switched radar systems to the PRECO PreView Sentry due to a budget decision from our client.

For our deep learning system, we originally planned to utilize the Keras Functional API to create a complex model capable of performing multi-object localization and classification. Several methods exist to perform detection on RGB imagery, such as FRCNN, YOLO, and MobileNet-SSD. The NVIDIA Jetson TX2, despite being an advanced embedded system with a GPU, is not as capable as a desktop computer with a graphics card, so we planned to create a relatively light-weight model similar to a Single Shot Detector. The key strength of an SSD is its balance of accuracy and speed, as it only passes the input image through a neural network once, rather than multiple networks for classification, bounding box regression, and confidence. The Keras Functional API supports multiple inputs and multiple outputs, making it suitable for our objective. Due to challenges around collecting sufficient training data we decided to use a pre-trained network. We chose the MobileNet-SSD because it met our requirements for supported classes and we believed it would have fast performance on the TX2.

Strengths of our proposed solution include resistance to inclement weather, utilization of two rapidly growing technologies (radar and deep learning), and versatility. A system that can perform classification with radar alone would be a step above current camera systems because it may perform in the rain, fog, or dark. Radar and deep learning are two quickly growing fields, especially in the automotive sector. By applying these to the agricultural and construction industry, we may position Danfoss in a unique position ahead of its competitors. Also, an object detection system can be used on a wide variety of vehicles, which is good because Danfoss serves a variety of target markets.

Weaknesses of our proposed solution include cost and complexity. Because many radar solutions suitable for our applications are actively being developed or suited for automotive manufacturers, a development kit is costly. Danfoss is also low volume relative to automotive manufacturers, so radar may end up being cost prohibitive to them in the long term. Complexity of the system is another factor that may make it difficult for us to collect a significant amount of training data for the neural network. We expect to detect multiple classes of objects, so training data collection is

crucial. Although this became outside the scope of our project, we recommend that Danfoss strongly begin considering data collection as early as possible.

2.6.2 CAN Controller PCB Design

The PCB was designed to contain all required I/O ports for the CAN Controller. Using a PCB would reduce the area used by the CAN USB Adapter (PCAN-USB.) This would make it a more suitable option for agricultural vehicles. Also, the PCB fabrication and shipping costs are less than the adapter. Below in Table 1 is a list of the parts needed for the PCB:

Vendor Part Number	Name	Manufacturer Part Number
1734-1646-ND*	4 Pin Female Board Mount	DTF13-4P
1734-1033-ND*	2 Pin Female Board Mount	DTF13-2P
595-TCAN4420DR**	CAN Controller	TCAN4420DR

Table 1: PCB Parts

*Digikey Part Number
** Mouser Part Number

The software chosen to create the PCB was Eagle. A footprint of the Deutsch Connectors had already been created for Eagle. The footprints for both the 2-pin and 4-pin connectors were found on SnapEDA.com. Using Eagle allowed us to finish the board design sooner because the team did not need to design the footprints from scratch.

Figure 1 Below is a picture of the Eagle representation of our PCB. The dimensions of the PCB are 2”x3”

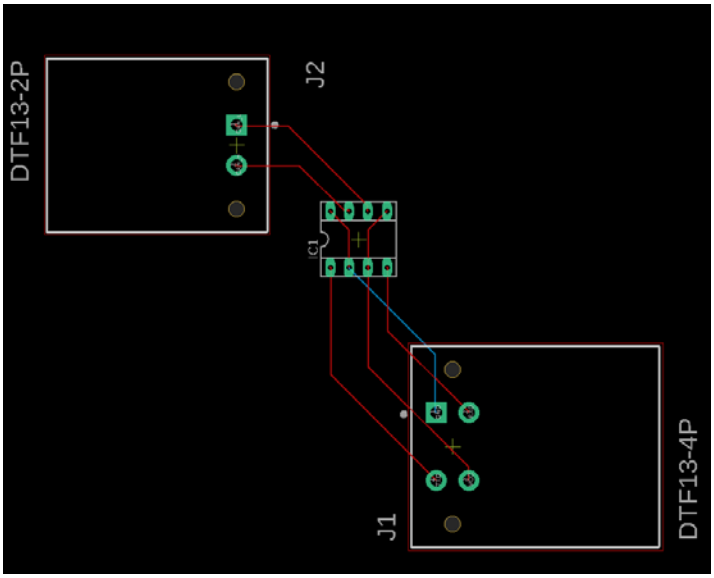

Figure 1: Eagle Schematic of PCB

We used the TCAN4420DR Datasheet from the Texas Instrument website as guidance for our PCB design. The datasheet provided examples for board layout and device specifications. It also provides package sizing needed for pin layout and footprint. After passing Design Rule Check (DRC) requirements, an exported .brd file was uploaded to the OSH Park website. OSH Park was chosen due to its simplicity, and lower costs. Once the PCB was fabricated, we tested the PCB with the Jetson and the Radar. We planned to use the CAN controller to bridge traffic between the Jetson and the radar. The CAN controller failed to send or receive any CAN traffic. Due to time constraints we did not have the time to troubleshoot, rather we went with the CAN USB Adapter. We believe this is due to the CAN drivers for the Jetson and not the hardware we designed.

2.7 Final System Design

2.7.1 Overview

In the end, our designed system must be able to detect and identify an object up to 30 meters away, as long as the object is within the 60 degree FOV. The radar will be a PRECO PreView Sentry, connected to an NVIDIA Jetson TX2 through CAN. A block diagram of our system in shown below in Figure 2.

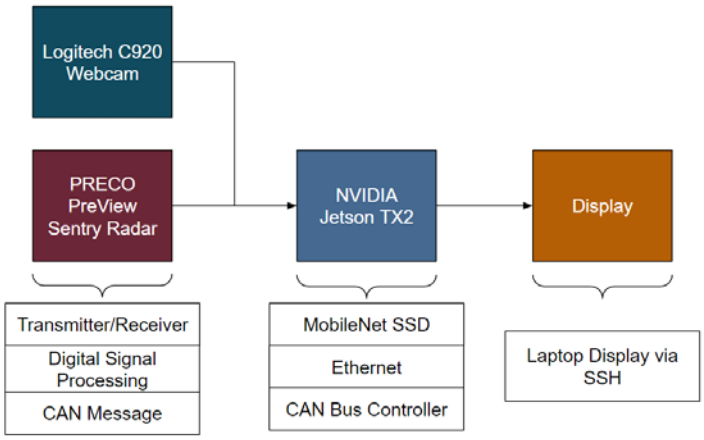

Figure 2: System Block Diagram

For the system output, the operator will be notified on a laptop display, connected to the Jetson TX2 via SSH with ethernet. The system is also compatible with using an HDMI connected display.

The system must be able to identify the object. To do so, we will be using a pre-trained neural network, specifically the MobileNet-SSD.

The system must be able to operate in all normal weather conditions such as cold, hot, windy, rainy, etc. The PRECO PreView Sentry, as an automotive radar, is suited for the task.

Below in Figure 3 is a photo of the full system:

Figure 3: Full System

2.7.2 Strengths and Weaknesses

Our proposed solution has several strengths and weaknesses, outlined below:

Strengths:

- Redundant system for increased safety, as radar and camera will be used for detection
- Easy to understand output, because the operator will have a screen showing detected objects
- Robust against inclement weather because the radar can still see through fog when the camera cannot
- Improved functionality and cost compared to automotive LiDAR
- Deep learning model can determine object type and image position

Weaknesses:

- Limited field of view relative to LiDAR, which may present problems for objects moving quickly into the vehicle path from a shallow angle
- High cost relative to only vision. Cameras alone are significantly cheaper than radar
- Python is a slower language than C++, so implementation in the future would likely need to switch languages in order to improve performance
- An additional display takes up extra room in the vehicle cab

2.7.3 Camera and Camera Calibration

In our final system design, we are using a Logitech C920 Web Camera as shown below in Figure 4. We chose this is camera this camera for its price point, ease of use, and familiarity

Figure 4: Logitech C920 Web Camera

All cameras, regardless of price or performance, have distortion. An extreme example of distortion is a fisheye camera - the image is warped. Calibrating a camera is the process of quantifying the distortion parameters and camera intrinsics, which may then be applied to the image to remove the distortion. This allows the computer vision engineer to map from image coordinates to world coordinates, constrained only by a scale factor.

Camera calibration is performed by taking upwards of 50 images of a checkerboard of a known size with a fixed-focus camera, then passing them to the provided camera calibration scripts. The parameters calculated are camera specific, not model specific. The following figures show the relationship between pixel coordinates and world coordinates.

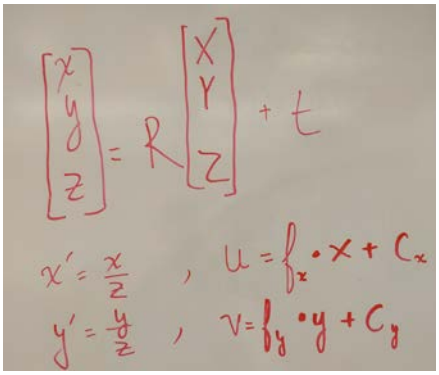

Figure 5: World Coordinate and Image Coordinate Relationship

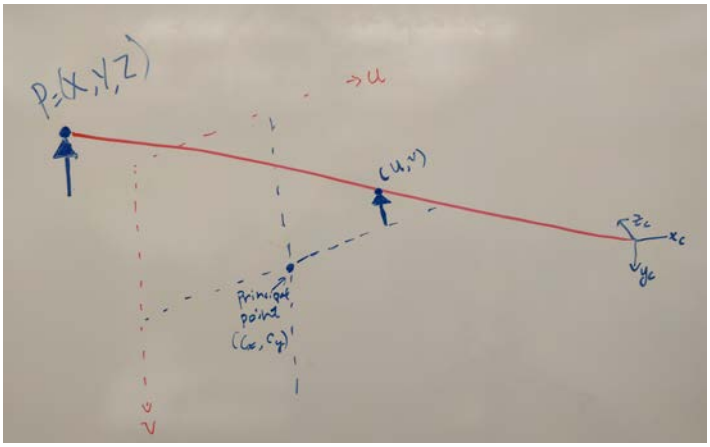

Figure 6: World Coordinate and Image Coordinate Graphical View

Note that f_x, f_y, c_x, and c_y are obtained through camera calibration, R is the identity matrix, and t is a zero vector.

Other applications for camera calibration include stereo vision and visual odometry. An undistorted image is important for a variety of applications and should be used in most computer vision applications where accuracy is required.

We have provided camera calibration scripts to Danfoss so they may calibrate their own cameras.

2.7.4 Interfacing with the PRECO PreView Sentry Radar

The documentation for the PRECO PreView Sentry Radar (henceforth called the radar) should be referenced for detailed information about CAN messages. This information is included as a brief summary to better understand how the software works, and should be considered supplemental to the documentation included in the code. Below in Figure 7 is a photo of the radar system.

Figure 7: PRECO PreView Sentry Radar

The radar provides a variety of CAN messages describing the location of objects relative to the radar. For this project, we make use of the Individual Targets message, which contains information such as the target ID (0-16), the angle and distance to the target, a confidence metric, and the signal to noise ratio. We parse this information in its own thread to prevent I/O operations from slowing down the main sensor fusion thread.

We utilize the top-down graphing view to show precisely what the radar sees without filtering, aside from removing objects with no confidence as specified by the radar. An example of the top-down graphing view is shown below in Figure 8.

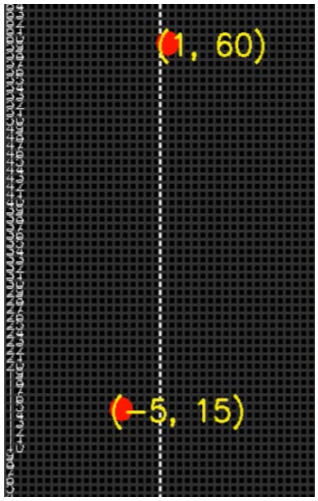

Figure 8: Top-down Graphing View

The user has the ability to select between drawing the object's location or the object's type (e.g. person, car, animal).

Because we know the camera calibration parameters, as discussed earlier, we can map from these radar coordinates to pixel (image) space. However, we cannot solve for an exact bounding box from only an (x,y) location from the radar. Instead, we are able to constrain the search region to the x direction for sensor fusion because the y-pixel will be constant regardless of object height or distance from the ground.

2.7.5 Deep Learning Information for Final System

In this section, we will discuss the neural network implementation that we chose to use, as well information about research we performed and topics of consideration when designing or choosing a neural network.

We utilize a pre-trained MobileNet-SSD, as provided on PyImageSearch [5]. MobileNet refers to the feature detector portion of the network, which is essentially a set of convolutional layers that has been trained to detect common features in a wide variety of images. MobileNet is designed to run efficiently on mobile devices, making it a light-weight network well suited to work on mobile

phones or a device like the Jetson TX2. The SSD refers to a Single Shot Detector, an object detection architecture that finds bounding boxes of all objects in an image. In short, the MobileNet-SSD utilizes the MobileNet feature extractor and SSD architecture to efficiently compute bounding boxes for all objects in an image.

The pretrained MobileNet-SSD that we used was selected because it was trained on a variety of objects that would be important to detect in the context of agriculture and construction. Examples include humans, horses, dogs, cars, motorcycles, bicycles, and cows.

The primary reason we transitioned from training our own network to utilizing a pretrained network was the lack of sufficient training data. Our original plan was to utilize web scraping with Google Images to collect training data, however we later found that finding objects in realistic context was difficult. A consideration for Danfoss is to begin collecting and labeling data from their customers, with permission, so they can build a dataset to train their own neural networks.

This section provides a high-level overview of deep learning for object detection. Deep learning, in essence, is the nonlinear mapping from an image to any desired output. The desired output may be multidimensional, including parameters such as object position, confidence, and type. As long as the engineer designing the network can create a differentiable loss function that describes the error between ground truth and a prediction, training the neural network will likely succeed provided sufficient training data.

When training a network, it is common practice to have a training dataset and a test dataset. Neural network weights are adjusted only during back propagation with the training dataset. The test dataset should be used only to evaluate network performance. By doing so, the network designer can prevent overfitting, which occurs when the network only performs well on the training dataset and not other inputs.

2.7.6 Multithreaded Functionality

This section describes the software architecture. The reader is encouraged to see the software for more detailed information, such as how each class is defined.

Input/Output operations (I/O) are slow. Examples of I/O operations in our program include reading CAN messages and reading images from the camera. By having these running on unique threads and sharing data with a main thread, the main thread is not I/O constrained and will run more efficiently. In Python, this is accomplished by using the threading library and queues, which are essentially thread-safe data structures that can contain things such as lists, images, and individual variables.

A diagram showing the overall software architecture is given below in Figure 9.

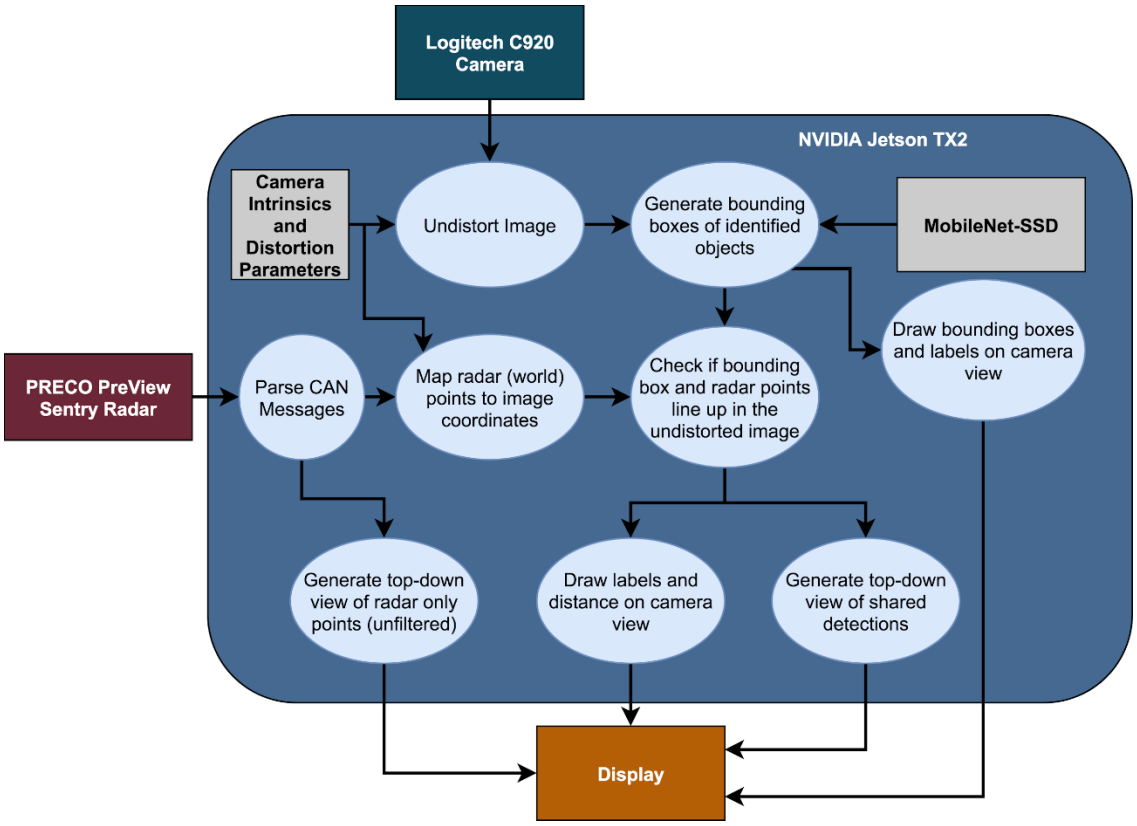

Figure 9: Software Architecture

There are unique threads for the deep learning object detector, CAN parser, and the main sensor fusion script. Each thread shares information with the other threads (as shown with the arrows in the diagram) with queues. An example of the output of each thread is shown on the next page in Figure 10. This is what the operator would see in the cab.

Figure 10: Graphical User Interface

For use on the NVIDIA Jetson TX2, another thread was added to facilitate visualization with the use of cv2.imshow(). We found through testing that cv2.imshow() could not be called from multiple threads on the Jetson, so a unique thread was created to handle only visualization. On Windows, this was not an issue. This is slightly slower, as it requires a large amount of data to be shared (four images for every frame from the camera).

2.7.7 System Stand for Radar and Camera

For the stand, there were a few different designs created. Initially, the design was sketched up and presented to ETG. After refining the design with their input, the final stand design now includes 3D printed parts to mount the camera and radar to an aluminum frame. The stand is made with extruded aluminum-alloy pipes from 80/20. We chose these parts because of their slotted sides which allow for easy assembly. The assembly was done by Radek Kornicki and his team at Danfoss in Ames.

The initial proposal was to use a box to house all the units, but we changed the design so the radar could be at the optimal operational height. The radar we used requires an operational height of three feet for optimal results, so the stand is just over three feet tall. The 3D printed parts are sturdy and weather resistant, just like the aluminum stand. The base of the stand is an H-frame which makes it stable. In Figure 11 and 12 on the next page, are a prototype design and final stand.

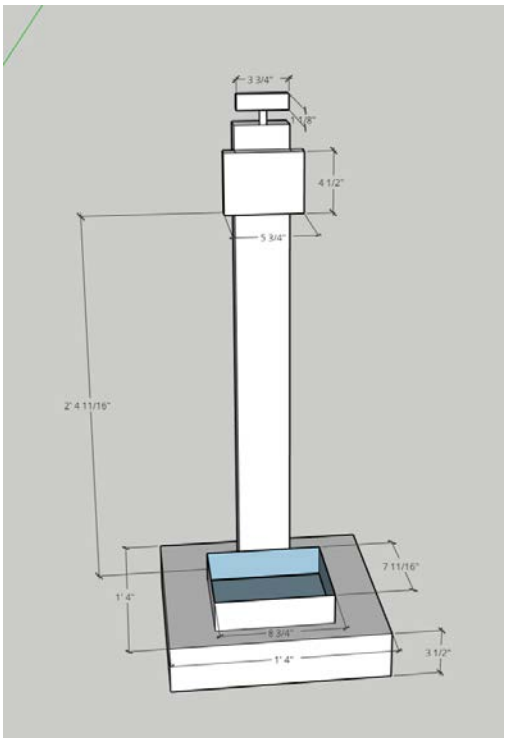

Figure 11: Prototype Design

Figure 12: Final Stand

2.8 Technology Considerations

Many different radar systems and deep learning APIs are available. Early in the design process we looked at using Keras for our system due to its simplicity, versatility, and a team member's previous experience with it. However, we ended up using a pre-trained network due to the challenges around collecting training data.

A radar system by Walabot has been used as a test subject. This radar system does not have the range needed to meet design requirements, but it does come with an interface that is simple to operate. The system was used as a test platform to collect radar data as imagery.

Texas Instruments offers a chip and another supplier has a radar system that meets our distance requirements, but does not offer an interface. Also, the system is quite expensive and exceeds our client's budget.

The Vayyar EVK is an attractive solution due to its relatively long range and ability to return an image. However, the increased cost and uncertainty about reliability in agricultural and construction environments led us to the Delphi ESR 2.5.

The Delphi ESR 2.5 was chosen at first as our final radar due to its simplicity to use. It returns the distance and angle to all objects in a series of CAN messages, which may be interpreted from the NVIDIA Jetson TX2. Although this radar does not provide an image we can use, we will utilize a camera to perform object classification for all detected objects.

The PRECO PreView Sentry replaced the Delphi as our radar system in August under direction from our client. The PRECO also uses CAN to communicate. It does have a shorter range than the Delphi. We found in our testing that the PRECO radar is jumpy and does not perform well for safety critical applications, which will be discussed sections 4.3.1.

2.9 Safety Considerations

If soldering circuits becomes a task, burns are a possible risk. Typical solder temperatures range from 500-800 degrees Fahrenheit. The team member attempting to use a soldering station should have basic knowledge on how it operates.

Our design will eventually be tested on large machinery such as agriculture and construction equipment which will create safety risks. Personal protective equipment (PPE) and/or training may be required in order to avoid cuts, head trauma, slips, etc.

When testing the system, the vehicle selected must have a secondary seat. The secondary seat will be used for a group member to analyze our system's performance. This will allow the vehicle operator's full attention to be on safely operating the vehicle.

2.10 Possible Risks and Risk Management

The cost of our system, in total, will reach at least $6475, The Delphi ESR 2.5 costs $6175 while the NVIDIA Jetson TX2 cost $300. Due to the high cost of the system, it has taken longer than expected to receive these necessary items. Other costs include a camera, connectors for various subsystems and CAN controllers.

Another risk is the time spent to train our neural network properly. Each neural network revision may require a trip to the Danfoss test track to test, or at the very least a significant time to retrain it. In order to deal with this risk, we will attempt to record a significant amount of test data (videos and radar logs) to verify our systems performance.

Aside from the aforementioned risks, technical challenges will be the biggest issue for our team to overcome. To manage technical challenges, we will collaborate with each other as well as reach out to Danfoss and Iowa State faculty members when we encounter issues. To prevent an issue from recurring, steps taken to solve an issue will be documented in our shared Google Drive for future reference.

2.11 Project Proposed Milestones and Evaluation Criteria

Deciding on a particular radar system is our first key milestone. This radar will need to meet all functional requirements. When we believe we have found the correct radar we will purchase it. The next milestone will be hooking the system up. Once we have the system setup we will test to make sure all components work with each other by testing the input and output of the system. Our final product was to be tested on Danfoss' test track, but we had to test only on ISU's campus. If our product detects the object it is supposed to, then we know it works. The test plan for this is in Section 2.13.

2.12 Project Tracking Procedures

Our group is using GitLab and Google Drive to track our process. Tasks will be assigned to individuals for tracking in GitLab, and all relevant documents will be shared on Google Drive to ensure all group members have access.

2.13 Expected Results and Validation

Our desired outcome is to be able to detect three different classes of objects. Not only do we want to detect these objects, but we want the driver to be able to see where the object is and what the object is via a user interface.

We will validate each requirement at different levels of testing. This will ensure that our system is performing to specifications.

In order to meet each functional requirement, a test will be performed for each of the requirements.

1. The system shall have a range of 30 meters.
    a. Criteria for success: We will consider the test a success if the system is able to detect objects at that range.
    b. Test: Range Test
2. The system shall function on machines traveling at up to a speed of 5 mph.
    a. Criteria for success: We will consider the test a success if the system is able to detect objects while traveling up to that speed.
    b. Test: Speed Test
3. The system shall have a FOV of ±30°.

    a. Criteria for success: We will consider the test a success if the system is able to detect objects within that angular range.
    b. Test: Angle Test
4. The system shall have a processing speed of 10 frames per second.
    a. Criteria for success: We will consider the test a success if the system is able to have a processing speed of 10 frames per second.
    b. Test: Processing Speed Test
5. The system shall detect objects greater than 0.4 m size.
    a. Criteria for success: We will consider the test a success if the system is able to detect an object of 0.4 meters wide at 30 meters.
    b. Test: Range Test
6. The system shall be weather resistant to water, dust, and shock.
    a. Criteria for success: We will consider the test a success if the components we selected have the appropriate IP rating and do not fail during full system testing on a machine.
    b. Test: This requirement will be tested in several on-machine tests.
7. The system shall have a probability of missed detection less than 0.3.
    a. Criteria for success: We will consider the test a success if the probability of missed detection is less than 0.3
    b. Test: Missed Detection and False Alarm Test
8. The system shall have a probability of false alarm less than 0.3.
    a. Criteria for success: We will consider the test a success if the probability of false alarm is less than 0.3
    b. Test: Missed Detection and False Alarm Test
9. The system shall run off of a 12V power supply.
    a. Criteria for success: We will consider this requirement met if the system is able to run off a 12V supply with or without a boost converter or inverter.
    b. Test: This requirement will not be tested.
10. The system shall fit inside 1’x1’x1’ space.
    a. Criteria for success: We will consider this requirement met if the radar module is able to fit in the required space and if the enclosure for the SoC is able to fit in the required space.
    b. Test: This requirement will not be tested.
11. The system shall detect at least 3 classes of objects.
    a. Criteria for success: We will consider this requirement met if the system is able to detect at least 3 classes of objects.
    b. Test: Object Classes Test
12. The system shall/should operate in the temperature range from -20 to 75 degrees Fahrenheit.
    a. Criteria for success: We will consider this requirement met if the components are rated for operation in the specified temperature range.
    b. Test: This requirement will not be tested.

The test plan in the next section describes in detail how we will administer the tests to validate the requirements.

2.14 Test Plan

We will conduct several different levels of testing. We will conduct testing at just a software level of the deep learning model. We will conduct component level bench testing and whole system bench testing. We will conduct whole system level testing with a stationary mount. Lastly, we will conduct whole system level testing on a machine.

1. Range Test
    a. We will validate the range requirement with testing of objects at 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95 feet. We will conduct this test with the system stationary and with the system moving on a machine at rated speed.
2. Speed Test
    a. We will validate the max operating speed by testing the system moving at up to 5 mph.
3. Angle Test
    a. We will validate the field of view requirement by testing the system with objects at different angles. We will test at 0°, 5°, 10°, 15°, 20°, 25°, and 30° with relation to the centerline of the machine.
4. Processing Speed Test
    a. We will validate the frame rate by recording data for a set period of time and verifying that the number of frames corresponds to an average of 10 frames per second.
5. Missed Detection and False Alarm Test
    a. We will validate the probability of Missed Detection and False Alarm be verifying that the number of objects detected in a certain time period corresponds to a range of .7 times the real number of objects to 1.3 times the real number of objects.

2.15 Test Results

We were able to conduct a few different tests with our system. We first conducted testing indoors with the full system. The results from this testing were not very promising. The range of the radar was very limited due to the indoor environment we were testing in. We were able to test the processing speed of the system. It performed at 4 FPS. This did not meet our functional requirement.

A picture of our indoor testing is shown below in Figure 13.

Figure 13: Indoor Testing

We also conducted testing outside up to 95 feet with a human and a car as objects. This testing was much more promising and we were able to verify a few of our functional requirements.

Table 2: Human Testing

Actual (feet)	Measured (feet)	Camera Detection
5	5	Yes
10	10	Yes
15	15	Yes
20	20	Yes
25	25	Yes
30	30	Yes
35	35	Yes
40	40	Yes
45	45	Yes
50	50	Yes
55	55	No

60	60	No
65	66	No
70	72	No
75	76	No
80	81	No
85	85	No
90	91	No
95	95	No

Table 3: Car Testing

Actual (feet)	Measured (feet)	Camera Detection
20	21	Yes
25	26	Yes
30	30	Yes
35	35	Yes
40	40	Yes
45	45	Yes
50	51	Yes
55	55	Yes
60	60	Yes
65	65	Yes
70	71	Yes
75	73	No
80	81	No
85	85	No
90	91	No
95	96	No

A screenshot of our outdoor testing is shown below in Figure 14.

Figure 14: Outdoor Testing

Our testing capabilities were limited and not as extensive as originally planned due to time constraints and scope changes this semester. We were not able to conduct any testing on a moving vehicle due to safety concerns. We were not able to conduct any missed detection or false detection testing, though subjectively we found that false detections with the neural network were low, but quite high with the radar. Similarly, missed detections with the neural network were low, but the radar would not detect people too far away.

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline

A breakdown of our tasks for the first and second semester is given below for 16 two-week sprints. This covers both semesters of senior design.

Table 4:

Sprint #	Dates	Deliverables
1	01/08 - 01/21	Schedule and Roles: Solidified our schedule, roles, responsibilities, and allotted times for meetings. Discussed individual positions and assigned tasks and deadlines for specific portions of the project.
2	01/22 - 02/04	Target Parameters, System Requirements, and Website: We will work on improving our website. We will create system requirements. We will determine the parameters we need for our radar.
3	02/05 - 02/18	Hardware Selections: We will determine our possible hardware options and their pros and cons. We will consult with our advisor and Danfoss.
4	02/19 - 03/04	Testing: We will begin testing some neural networks to gain a better understanding of them.
5	03/05 - 03/25	Final Hardware Selections: We will select our final hardware by this sprint. We will consult with our advisor and Danfoss.
6	03/26 - 04/08	System Integration Planning: We will plan how we will need to integrate parts of our system together and what roadblocks we may run into.
7	04/09 - 04/22	System Integration and Planning: We will continue planing how we will integrate our system.
8	04/23 - 05/04	Documentation and Planning: We will begin preparing to plan for next semester and our presentation.
Break	5/05-8/19	Summer Break
9	8/20-9/9	Updates from Scope Changes: Gathered info on our new radar from PRECO, and created the schedule for the upcoming weeks. Made the plan to build the harness for the radar. Researched integrating the data from the PRECO radar to the Jetson TX2.
10	9/10-9/23	Software Development and PCB Design: We will continue software development to prepare for a demonstration once the hardware is ready. We will continue to design a PCB for the CAN controller.
11	9/24-10/7	Software Development and PCB Creation: We will continue development on software to interface with a camera and CAN. We will order the PCB to be fabricated. We will order parts to build the system harness.
12	10/7-10/21	Harness Fabrication and Testing: We will build our harness and begin testing of the radar.
13	10/22-11/4	System Testing and Validation: We will begin testing our final system and build our CAN controller PCB. We will not make any hardware changes after this

		sprint.
14	11/5-11/19	System Testing and Validation: We will continue testing our final system and make any software changes.
15	11/20-12/2	Report/Presentation: We will thoroughly document our system and prepare to present our project.

A Gantt chart illustrating our plan for both semesters is shown below:

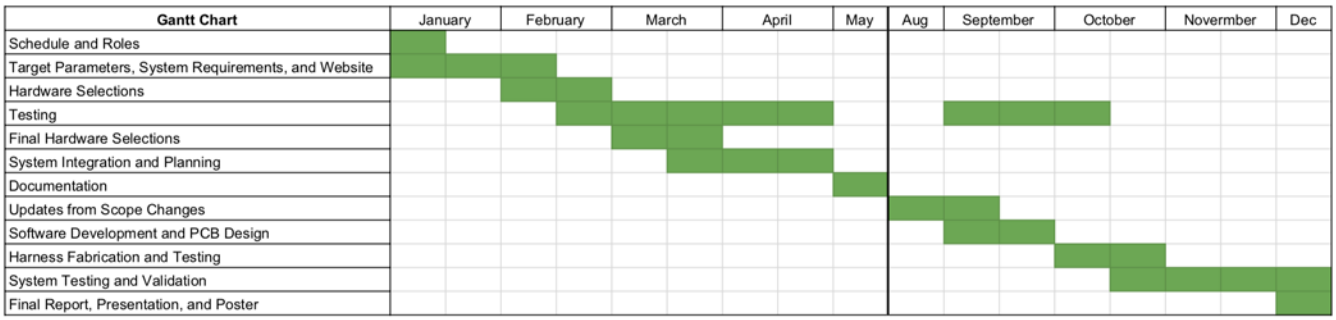

Figure 15: Gantt Chart of proposed schedule for the spring and fall semester

3.2 Feasibility Assessment

Our project has several challenges that we will need to overcome. The first is managing cost. The Delphi ESR 2.5 is $6175, while our NVIDIA Jetson TX2 is $300. Smaller costs, such as CAN controllers, a camera, and connectors must be considered as well. The camera will likely be a <$100 webcam in order to demonstrate functionality. Connectors and associated tools will be from Deutsch, and will cost approximately $450. Demonstrating to Danfoss the value of the system as a prototype is the first hurdle that must be overcome - therefore we will create written reports justifying component choices and associated costs. Danfoss made a hardware change from the Delphi ESR 2.5 to the PRECO PreView Sentry, this radar is approximately $1000 with a PCAN-USB adapter.

The feasibility of designing and training an object detection deep learning model is high. Object detection is a well-researched field in deep learning, and many resources exist to learn from. Combining the deep learning model output with the output of the radar will be the largest risk here. In order to keep this feasible, we will design our software in such a way that we warn the operator of a risk if seen by the radar and use the neural network to enhance information. An example is identifying the object, giving it a risk rating, and drawing a correct size bounding box around the object on the LCD. We can also use the neural network for non-maximum suppression to decrease duplicate hits for large objects detected by the radar..

3.3 Personnel Effort Requirements

This information is found on our GitLab page[4]. Each action item includes dates, assignees, and details all listed out on the Issue Board section of our Git. For a final summary of personal contributions, please see the weekly and biweekly reports on our team website.

3.4 Other Resource Requirements

Our group must get a general understanding for radar, deep learning, and Python in order to effectively make this project work. To learn about each of these, we will utilize resources publicly available on the internet and library. Examples include a variety of machine learning blogs and programming examples for Python found online.

To collect training data, we may possibly need access to a vehicle and a testing area. Danfoss will provide this at their Ames facility.

Storage of training data must follow Danfoss' data privacy rules. If our client wishes for us to store data collected from their testing area on their servers, we will do so.

3.5 Financial Requirements

This system has several associated costs. Because this system is a single-quantity prototype, the cost of associated components will be high relative to volume production. A breakdown of costs is listed below.

- Delphi ESR 2.5: $6175
- PRECO PreView Sentry with PCAN-USB: $1000
- NVIDIA Jetson TX2: $300
- Logitech C920 Web Camera: $80
- Deutsch tools, connectors: $450
- PCB: $55
- CAN Controllers: $13

4 Closure Materials

4.1 Conclusion

Our project is to develop a system for Danfoss to use on machinery such as tractors, loaders, excavators, and other heavy equipment that can use radar to detect objects. We will be utilizing deep learning to recognize objects in order for these vehicles to determine if there is a hazard in range of the radar.

Danfoss decided to use the PRECO PreView Sentry because of its cost versus the Delphi ESR 2.5. This radar will be connected along with a camera to the Jetson TX2 computer which will then process the data and output information to a display. We hope to create a device that we can put into testing on Danfoss' test track in Ames in order to show leadership and engineers from Danfoss the work we have accomplished and the system we have created.

By leveraging deep learning with sensor fusion we were able to detect and localize objects. This will appear on the display with the classification of the object and its location. Two views will be provided to the operator: a top down view and a camera view.

With all of this accomplished we will have tested a new technology for Danfoss. This technology can help make safer working conditions. Hopefully, this technology will be feasible for future development into a product for our client, and a safer society surrounding the machinery our system is implemented on.

4.2 References

[1]E. Mason, B. Yonel and B. Yazici, "Deep learning for radar," *2017 IEEE Radar Conference (RadarConf)*, Seattle, WA, 2017, pp. 1703-1708. doi: 10.1109/RADAR.2017.7944481

[2]W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A. C. Berg. “SSD: Single Shot MultiBox Detector.” *Cornell University Library*, 29 Dec. 2016, https://arxiv.org/abs/1512.02325.

[3]S. Ren, K. He, R. Girshick, J. Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” *Cornell University Library*, 6 Jan. 2016, https://arxiv.org/abs/1506.01497.

[4] Iowa State ECE Senior Team. “Senior Design GitLab Repository.” *GitLab*, 23 Apr. 2018, https://git.ece.iastate.edu/sd/sddec18-18

[5]A. Rosebrock, "Object Detection with Deep Learning and OpenCV," *PyImageSearch*, https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/

[6]Stewart, Louis. “RadarCat for Object Recognition.” *SACHI*, sachi.cs.st-andrews.ac.uk/research/interaction/radarcat-exploits-googles-soli-radar-sensor-for-object-and-material-recognition/.

[7]Van Rossum, Guido. “PEP 8 -- Style Guide for Python Code.” *Python.org*, 1 Aug. 2013, www.python.org/dev/peps/pep-0008/.

[8]Xu, Joyce. “Deep Learning for Object Detection: A Comprehensive Review.” *Towards Data Science*, Towards Data Science, 11 Sept. 2017, towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9.

4.3 Appendices

4.3.1 Recommendations for Future Development

Through our testing, as described in section 2.14, we came to the conclusion that the PRECO PreView Sentry radar should be avoided for safety-critical applications. False positives were abundant, and even detection of large objects such as cars was jumpy. Instead, we recommend the continued research of computer vision exclusively. For the low-speed applications that Danfoss targets, stereo vision may provide better results at a lower cost than a better radar. Computer vision is ultimately more flexible than a radar as well. In our testing, we found that computer vision and the neural network were better able to resolve individual objects, such as two people standing next to each other. If spatial information is needed, stereo vision will likely provide more stable results than the radar. Although stereo vision is outside the scope of our project, we believe it is a logical next step for Danfoss to explore.

We also recommend that Danfoss does not use the NVIDIA Jetson TX2. The Jetson’s main selling point is an embedded GPU for use with deploying deep learning applications. However, we found that its performance, even on the lightweight MobileNet-SSD, was only about 4 frames per second. Another issue with the Jetson is the lack of proper documentation. NVIDIA has provided official solutions on forums for issues customers have, rather than addressing the issues in documentation that should be provided. One example is the configuration of the CAN bus.

Instead of the Jetson, we recommend Danfoss explores NVIDIA's higher power options designed for the automotive market, or consider FPGA solutions from Xilinx.

As computing power increases and cost decreases, neural networks will continue to become more attractive to implement in consumer products. Radar, as we found, can be jumpy and inaccurate, making it hard to recommend for safety-critical applications. Ultimately we recommend that Danfoss goes all-in on computer vision and avoids the use of cheap automotive radar. A decent deep learning platform to start development with is Keras. Keras makes it easy to design and test neural networks. If speed is required, we recommend using a different platform such as Caffe for designing the model that will actually be deployed. In short, Keras makes research easier but is not optimal for deployment. Caffe will provide better results for deployment but will make development more difficult. A neural network architecture can be implemented on any given deep learning platform. We also recommend that a production solution be written in C++ rather than Python due to resource usage and speed.

4.3.2 User Manual

Running the program is intended to be simple and easy to understand. Minimal configuration is required for use on the NVIDIA Jetson TX2. A zip folder with the code for the project has been provided to the client and may be copied to the Jetson to any location. The Jetson username is nvidia and the password is nvidia.

Prior to running the program, the various components must be connected to the Jetson TX2. The radar must be connected to the PCAN-USB adapter, which is subsequently connected to the USB hub to the Jetson TX2. The camera is also attached to the USB hub. The radar is connected to the 12V battery with the included harness. Be sure to plug in the Jetson TX2.

To run the program, simply open a terminal, navigate to the folder in which the entire project is contained, and type "python Main_RUN_ME.py".

If the user is interested in doing further development or running the scripts on a Windows computer, they must install the following items:

1. Python 3.6
    a. OpenCV 3.4.0
    b. Numpy
    c. python-can
2. PCAN-USB drivers, available online

It is recommended that the user installs the Python library with pip, which is an included package manager with Python. A wide variety of tutorials exist online about installing Python and packages for it. These have already been installed on the Jetson TX2 by our team.

To mount the device on a vehicle, it is recommended to utilize the 80/20 aluminum mounting system to create a vehicle-specific mounting solution.

Our system includes debugging messages that the user may follow in the event of an error. These are summarized below.

Error Message	Recommended Action
`"ERROR: Failed to load neural network. Check that the files exist and paths are correct."`	It is likely that the user attempted to run the program without ensuring that the paths to the neural network were correct. If they simply run the script from the provided folder without moving anything, this error will never occur.
`"ERROR: CAN disconnected. Check connection and restart in a new terminal."`	The radar became disconnected. Because it was not able to undergo a graceful shutdown, the user will need to rerun the script in a new terminal once they check the connection everywhere. If the connections are good, ensure the battery did not die.
`"Warning: Sensor blockage. Check radar."`	This is not an error, but it is likely that the radar is physically blocked by a dense object. The operator should ensure they did not run into anything that stuck to the radar. Examples include snow and mud.
`"ERROR: CAN failed to connect. Likely closed improperly last time. Try running from a new terminal, then check connections."`	The CAN bus did not connect, likely due to a not-graceful shutdown from a previous failure. Restart the script in a new terminal.
`"ALERT: CAMERA ERROR. CHECK CONNECTION.”`	Check the camera connection.

4.3.3 Code

Due to the NDA with Danfoss, we refer the reader to our GitLab page to access our code rather than including it here. You may also reach out directly to Kellen O'Connor (kelleno@iastate.edu) to access it.