Object Detection and Identification with Sensor Fusion

Design Document

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

Tucker Creger - Project Manager
Kellen O'Connor - Deep Learning Architect
Eric Bishop - Software Developer
Mitch Hagar - Radar System Lead
Clayton White - Hardware Design Engineer
Nihaal Sitaraman - Hardware Developer

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

Revised: 04-28-18/2.0

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

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. Wang for advising our team.

1.2 Problem and Project Statement

As the agriculture and construction industries require autonomous solutions for increased safety and productivity, the need to sense objects in the equipment path increases. 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. Radar is virtually immune to the effects of low light, complete darkness, rain, snow, and fog.

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

1.3 Operational Environment

The operating environment for the system will be on agriculture 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. It will also need to be encased in something weatherproof, or in an area devoid of environmental effects to the system.

1.4 Intended Users and Uses

The intended use is for certain agricultural vehicles and construction equipment that are a key area for our client. The primary user is the vehicle operator, who will use the system to have an increased awareness of objects and dangers in the vehicle'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/environment.
- The system-on-a-chip and radar will be able to operate in a rugged environment.

Limitations:

- The system will only operate up to 15 mph. This will cover a large range of agriculture and construction equipment.
- The system will not be 100% immune to sensor blockage by dust and dirt.

1.6 Expected End Product and 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 an LCD screen in the cab of the objects' positions and types. The delivery date is December 2018. This system will be used for a demo on a piece of construction or agriculture equipment for our client.

Other deliverables include proposals regarding our radar and SOC selection, reports on which deep learning platforms are most suited for use in mobile applications, and a final report regarding the feasibility of implementing radar in construction and agricultural applications. The delivery date of all reports 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 attached LCD screen (LCD screen not property of Danfoss), camera, and Delphi ESR 2.5.
2. Trained deep learning model for object detection relevant to an agricultural/construction vehicle.
3. Python script that utilizes attached components and deep learning model to display object information on in-vehicle LCD screen.

2. Specifications and Analysis

2.1 Proposed Design

Since taking on this project we have examined many different solutions and even started testing some solutions. The main functional requirement for our system is that it uses radar as the primary sensor. Our client proposed a few different radar systems to begin our search and testing. Our client provided three radar kits from Walabot for initial testing.

We initially developed a block diagram as a concept of our system. This is shown below in Figure 1. At this stage in the design process we were still researching into radar technologies and deep learning languages.

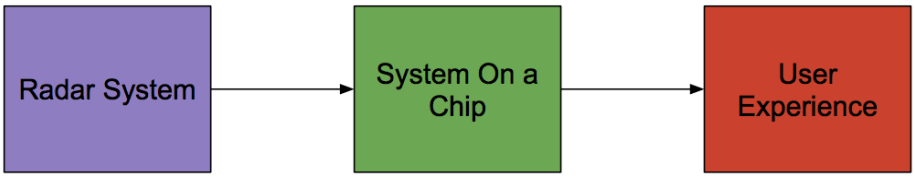


Figure 1

We then began generating functional and non-functional requirements. These are located in our Project Plan on our team website. We also began narrowing down some of our options for deep learning languages. We ultimately decided on using Keras because of our team's experience and its functional API. We also conducted testing on the Walabot kits that were provided by our client. We were able to use these kits for some very short range tests. We were able to use the Walabot SDK to map objects in a short range and look into walls to pick up wires and pipes. We also selected the system on a chip that we would use. We planned on using an NVIDIA Jetson TX2 due to its graphics card, output interferences, and support for Ubuntu.

In our research into radar technologies and systems, we examined many different product offerings. We looked at radar chipsets from Texas Instruments, NXP, Infineon, and Omniradar. We also looked at full radar systems from Delphi, Ainstein, Continental, Uhnuder, Arbe Robotics, Ghostwave, Metawave, Oculli, Vayyar, Steradian Semiconductor, and Lunewave. We held teleconferences with representatives from several of these companies to gain insight into their products and if they would be the right fit for our application. After these discussions, we were able to determine that we needed to formulate our primary system plan and a backup plan depending on the capabilities of our radar.

Our primary plan was to use a radar system for object detection and identification. We would train the deep learning model on the radar data by using a camera with a pre-trained model to identify objects and link them to the object in the radar data. This plan would hinge on our ability to get feature-rich data that would be suitable for testing and training. For this system we determined that the Vayyar EVK was the most promising candidate radar system. You can see a block diagram of the system in Figure 2 below.

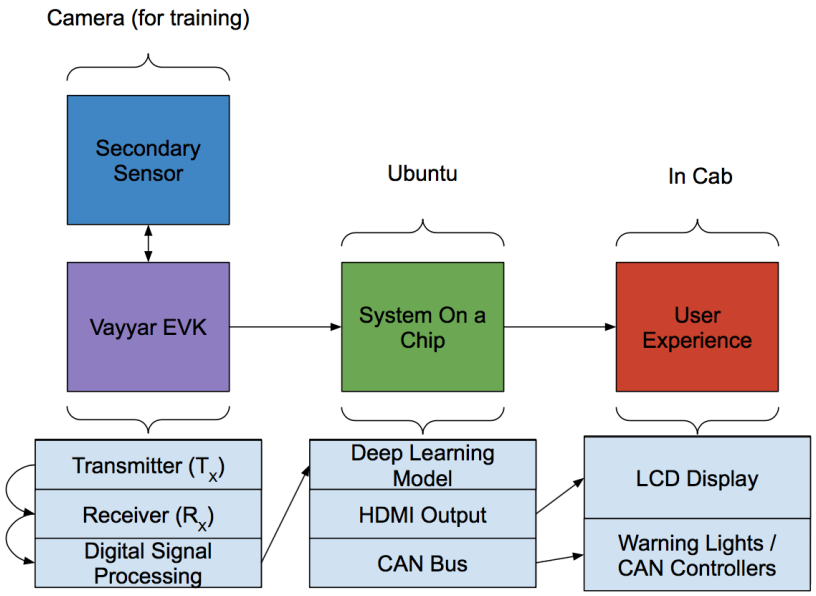


Figure 2

Our back-up (and chosen) plan was to use a radar system only for object detection and to use a camera for object identification. The radar system would provide the range, angle, and power of the received signal for the object and we would map that to the image from the camera. This plan has trade-offs with the camera being susceptible to decreased performance in low light conditions, dust, rain, and fog. The camera would also not be able to operate in the dark. For this plan, we decide the best candidate is the Delphi ESR 2.5 due to its range and CAN capabilities. A block

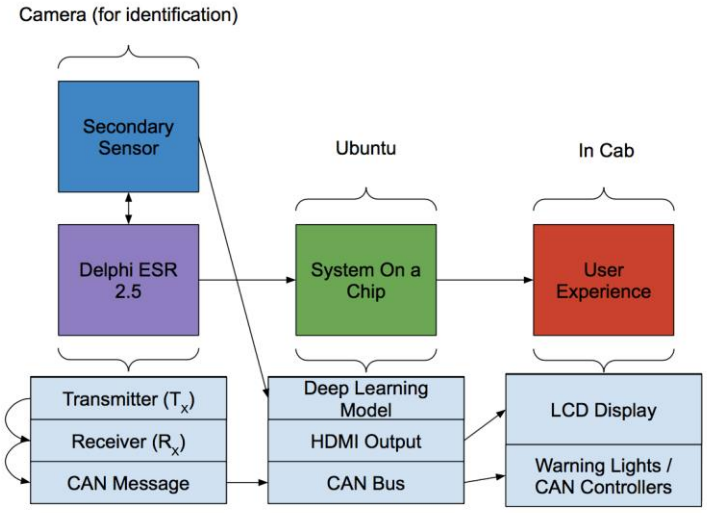


diagram for this system is shown in Figure 3.

Figure 3

The primary system has the advantages that the radar might be suitable for deep learning and would be compatible with our backup plan. The main risk with the primary system is that we do not fully understand the feasibility of using radar "images" with a deep learning model. The backup plan system has the advantages of increased range and a CAN interface. The risks associated with it are related to the limitations of using a camera for identification of the objects in the machine's path. We prepared a proposal document for our client to review detailing the advantages, disadvantages, and risks of the two systems. The client reviewed the document and decided the Delphi ESR 2.5 better aligns with their goals after our project finishes, so the Delphi ESR 2.5 and backup plan will be pursued for the remainder of the project.

For development of our deep learning model, we began testing our plan to train a network with images labeled from a pre-trained network. We used a pre-trained Caffe network that was able to detect 20 classes of objects. We modified this script to work with a live feed from a webcam and to export the images with the bounding box coordinates in a text file. We used this as the training data for our custom Keras network. After training our network to classify and localize one person, we used the model to identify one person in a webcam feed. This was a quick proof of concept test to verify the feasibility of using data from a pre-trained network to train our own network. You can see this process in Figure 4 below.

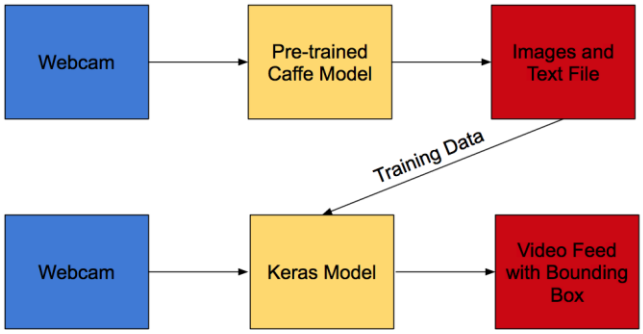


Figure 4

You can also see a diagram of the Keras network we built in Figure 5 below.

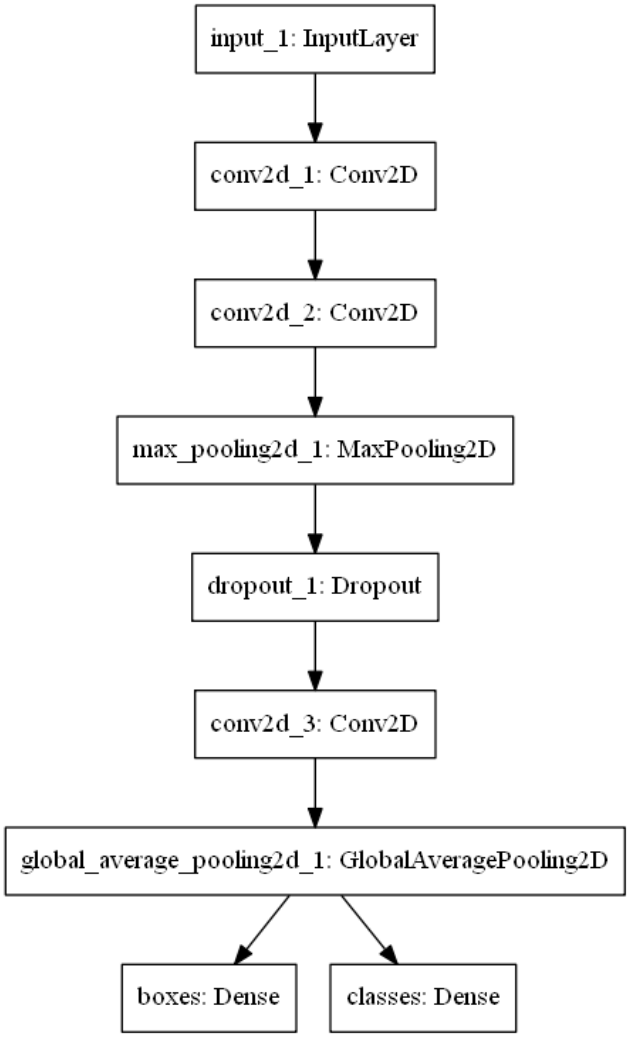


Figure 5

Our next steps were to modify the pre-trained network to support multiple objects and classes in a scalable export. We modified a pre-trained neural network to capture images with the data and time in the image name and a text file per image that supports an infinite number of classes/objects per line. The next design challenge is for us to build a neural network that supports an input of an infinite number of classes/objects per line in the text file. We plan on using Keras' functional API to help us accomplish that. The output space will be a discrete set of possible locations an object can be in the image, with a third dimension corresponding to object class.

2.2 Design Analysis

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 will pursue for the remainder of the project 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 50m 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 fewer 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 would 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.

For our deep learning system, we will 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 TX2, despite being an advanced embedded system with a GPU, is not as capable as a desktop computer with a graphics card, so we plan 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.

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. We may need to collect thousands of samples in order to train the model. Evaluating our model requires retraining it each time we make a change, which could take multiple hours. Due to the complex nature of this system, we will have to be thorough when collecting data and ensure that labeling is done properly, or else the time required to train and modify our system may add up.

3 Testing and Implementation

For the deep learning with radar, our group will need to two types of tests. The first test will be for the collection of training data for the deep learning system. The second test will be for testing our final product on Danfoss' test track. To test the data collection, we will implement an array of test data into Python. To ensure the data collection is what we think it is we will compare it with the test data in Keras. The test conducted at the test track will have our system mounted on the machinery Danfoss has available. This test will determine if our combination of data collection and radar panned out.

To test the collection of data, we will gather resources from the internet. These resources will consist of a variety of test data pertaining to our specific test field. The test data will need to be of the same type of data that our data collection is. The test data will serve as the way for us to know if the data collected is viable. We will compare the collected data with the test data on our python script created specifically for this task. As long as the data collection and the test data match on the python script, then we know our data is usable.

Testing our final product will consist of many cases.

The test cases are as follows:

1. Range Test
    a. We will validate the range requirement with testing of objects at 5, 10, 15, 20, 25, 35, 45, 55, and 65 meters. 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 2, 5, 10, and 15 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 center line of the machine. We will conduct this test with objects at 5, 10, 15, 20, 25, 35, 45, 55, and 65 meters.

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 15 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.

3.1 Interface Specifications

The group will use the Delphi ESR 2.5 to gather radar data. The NVIDIA TX2 will perform computations on the gathered radar data. This radar data will be compared with data acquired from a camera during the training process. Although the specific RGB camera has not been finalized yet, it will most likely be a standard webcam, such as the Logitech C920. The gathered

information will be passed through our neural network, built with Keras. We will be coding in Python to control how our model operates since Python is the language that Keras runs on.

3.2 Hardware and Software

The Delphi ESR 2.5 will be the radar unit used for object detection. The interpretation will be done by the NVIDIA TX2 Module. The camera will be used to perform classification. We will load the Keras deep learning program onto the SOC so that the collected data may be analyzed in real time. The system will be designed to display information on an LCD screen if an object is detected. Information includes angle, distance, and objection classification.

3.3 Functional Testing

Functional testing will be done to see if the angle, distance, and object classification information returned to us by the system is accurate to a certain degree. The farther and smaller the object is the lower the accuracy will be. Minimizing this error will be a key task in finalizing the project. Specifications for functional requirements are given in our Project Plan, found on our team's website.

3.4 Non-Functional Testing

The designed system must operate in the weather conditions provided by the client. Rainy, snowy, cold, and hot weather conditions should have minimal effect on the operation of the system. Protective casing will most likely be designed to prevent any environmental damage done to the system.

The devices being used in the design have been certified to be not be temperature dependent. To be sure, temperature chambers can be used to increase the temperature of the devices to ensure that the system operates as expected.

Lastly, the casing should be compact, robust, and appealing to the eye. Testing will be done to guarantee the casing is strong enough and also not an eye-sore.

3.5 Process

At the end of the first semester, we have only been able to conduct limited component level testing of the final components we plan to use in our final system. We have been able to conduct additional proof of concept testing on technologies we will likely not be using in our final system.

We have conducted limited testing on our SOC, the NVIDIA Jetson TX2. After we troubleshooted a display driver issue with the Jetson TX2 we were able to test using OpenCV and a webcam with the Jetson. We tested this by using a python test script to use the OpenCV API to display the output of the webcam on an LCD screen.

We have not been able to conduct any component level testing on our final radar solution, the Delphi ESR 2.5. We plan to conduct several bench and stationary tests of the radar once we have finalized the wiring harness design for the radar. We will use these test to also test the processing speed of our deep learning model. We hope to begin conducting these tests in September 2018.

We have also conducted proof of concept testing on using a pre-trained network to classify and label objects. These labeled images were used to train our own custom network.

3.6 Results

In our testing of the Jetson TX2 we have had positive results using the OpenCV API. This confirms an important part of our software system is functioning for our final system.

We have also had positive results using a pre-trained network with a webcam to classify people and other objects. We were able to successfully label one object per image and use those images as the input for our custom network. We will have to refine this method for further testing with multiple objects and classes per image in the fall of 2018.

Our other testing of hardware and software has been limited in the first semester due to the many factors. The majority of the semester was spent trying to source a radar module for our project. The remaining portion of the semester was spent on working on a wiring harness design and sourcing additional components.

We plan to have more test results early in the fall semester.

4 Closing Material

4.1 Conclusion

In summary, our main goal is to have a system using radar which will be effective with farm equipment and will be functional in various weather conditions. Thus far, we have started to order parts and have begun testing with a camera system and deep learning models. We hope to have our radar system finalized and working with the rest of our system in the next few weeks. We have two proposed ideas for our system and once we can begin testing the radar and secondary sensor, we will be able to finalize a solution. We have decided on our system-on-a-chip solution, having picked and ordered the NVIDIA Jetson TX2. This is the product which we will be using to run our neural network and analyze the data we collect from the radar and the secondary sensor.

Using Keras and Tensorflow linked to data from an RGB camera and a radar, we can localize and classify objects in our path. Using this information, we can prevent accidents and will implement some sort of warning system. We will have our units exchange signals and information using a CAN Bus and HDMI output to a screen. This is the ideal solution to reach our goal because it is feasible and a tried-and-tested method, used by the autonomous vehicle industry.

4.2 References

F. Corrigan, "12 Top Lidar Sensors For UAVs and So Many Great Uses," *DroneZon*, 12/17/2018. [Online]. Available: https://www.dronezon.com/learn-about-drones-quadcopters/best-lidar-sensors-for-drones-great-uses-for-lidar-sensors/.

"Luminar" *Luminar*, 2018. [Online]. Available: https://www.luminartech.com/technology/index.html.

"Phoenix LiDAR Systems" *Phoenix LiDAR Systems,*2018. [Online]. Available: https://www.phoenixlidar.com/scout/.

Z. He, C. Huang, G. Ning, Z. Zhang, "Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking," 2016. [Online]. Available: https://arxiv.org/pdf/1607.05781.pdf.

"Real-time object detection with YOLO," 5/20/2017. [Online]. Available: http://machinethink.net/blog/object-detection-with-yolo/.

"Ghostwave," *Ghostwave*, 2018. [Online]. Available: http://www.ghostwaveinc.com/.

"Getting started with the Keras functional API," *Keras Documentation,* 2018. [Online]. Available: https://keras.io/getting-started/functional-api-guide/.

"Rudi Embedded System," *Jetson TX2/TX1 Embedded System,* 2018. [Online]. Available: http://www.connecttech.com/pdf/ESG503_Rudi.pdf.

3Blue1Brown. "Neural Networks," *YouTube*, 11/3/2017. [Online]. Available: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi.

J. Rieke. "Object detection with neural networks – simple tutorial using keras," *Towards Data Science,* 6/12/2017. [Online]. Available: https://towardsdatascience.com/object-detection-with-neural-networks-a4e2c46b4491.

"Oculii Automotive radar (OAR) – 200 Overview," *Oculii,* 2018. [Online]. Available: http://www.oculii.com/OAR-200.html.

"Ainstein Radar," *Ainstein,* 2018. [Online]. Available: http://ainstein.ai/.

T. Peynot. L. Stanislas. “Characterisation of the Delphi Electronically Scanning Radar for Robotics Applications,” *ARAA,* 2012. [Online]. Available: http://www.araa.asn.au/acra/acra2015/papers/pap167.pdf.

A. Sachan. “Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN, YOLO, SSD,” *CV-Tricks,* 2017. [Online]. Available: cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/.

A. Roseborck. “Object detection with deep learning and OpenCV,” *pyimagesearch,* 9/11/2017. [Online]. Available: https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/.

“Udacity-SDC-Radar-Driver-Micro-Challenge,” *GitHub*, 9/22/2017. [Online]. Available: https://github.com/diyjac/Udacity-SDC-Radar-Driver-Micro-Challenge.

“Delphi ESR 2.5,” *Autonomous Stuff,* 2018. [Online]. Available: https://autonomoustuff.com/product/delphi-esr-2-5-24v/.

“NVIDIA Jetson TX2 Module,” *NVIDIA,* 2018. [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-tx2.

“Embedded low-power deep learning with TIDL,” *Texas Instruments*, 2018. [Online]. Available: http://www.ti.com/lit/wp/spry314/spry314.pdf.

“Short-Range Radar (SRR) Reference Design Using AWR1642,” *Texas Instruments*, 2018. [Online]. Available: http://www.ti.com/tool/tidep-0092.

“Automotive 77GHz Module Reference Design with Object Data Output,” *Texas Instruments,* 2018. [Online]. Available: http://www.ti.com/tool/TIDA-01570.

“Vayyar,” *Vayyar,* 2018. [Online]. Available: https://vayyar/com.

4.3 APPENDICES

Delphi Electronically Scanning RADAR

Delphi has applied more than 20 years of radar experience to develop its award-winning electronically scanning radar (ESR). Leveraging expertise gained from radar production that began in 1999, Delphi brought ESR to market at a price that is helping make radar-based safety and convenience systems more affordable in the automotive market.

Delphi's multimode ESR combines a wide field of view at mid-range with long-range coverage to provide two measurement modes simultaneously. While earlier forward looking radar systems used multiple beam radars with mechanical scanning or several fixed, overlapping beams to attain the view required for systems like adaptive cruise control, Delphi's multimode ESR provides wide coverage at mid range and high-resolution long-range coverage using a single radar. Wide, mid-range coverage not only allows vehicles cutting in from adjacent lanes to be detected but also identifies vehicles and pedestrians across the width of the equipped vehicle. Long-range coverage provides accurate range and speed data with powerful object discrimination that can identify up to 64 targets in the vehicle's path.

Delphi's technologically advanced ESR uses proven solid state technology plus class-leading performance, packaging and durability to offer customers game-changing forward radar detection. The quality of data provided by Delphi's system enables powerful functionality including adaptive cruise control, forward collision warning, brake support and headway alert.

ESR enables the following features:

- Adaptive Cruise Control with Stop & Go
  - Enhances driver convenience
- Forward Collision Warning
  - Helps reduce the potential for an accident and injury
  - Helps reduce the potential for property damage
- Brake Support
  - Helps reduce the potential for an accident and injury
  - Helps reduce the potential for property damage
- Headway Alert
  - Provides distance information
  - Alerts driver when the preset time-gap to vehicle ahead is violated

AUTONOMOUSTUFF

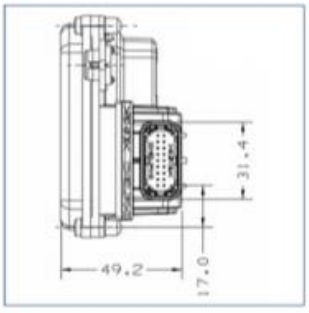
Tel. 309.291.0966 | www.AutonomouStuff.com | info@AutonomouStuff.com

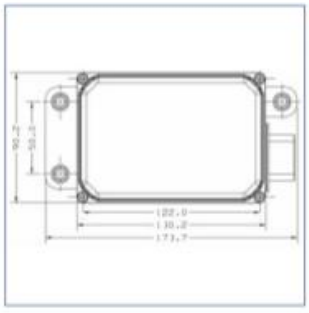
Delphi Electronically Scanning RADAR

The Delphi Advantage

- Multi-mode, multi-application capability
  - Simultaneous long- and mid-range functionality allows one radar to be used for multiple safety systems including adaptive cruise control, headway alert, collision warning and mitigation and brake support
- Solid-state Technology
  - No moving parts
  - Extremely reliable
- Class-leading performance and durability
  - Resistant to vibration and extremely robust
  - Innovative design provides excellent multi-target discrimination plus precise range, approach speed and angle data
  - Dual-mode classification enhances object reliability
  - Simultaneous Transmit and Receive Pulse Doppler (STAR PD) Waveform provides independent measurements of range and range-rate and superior detection of clustered stationary objects
- Compact packaging
  - Complete radar module, including electronics, measures just 173.7 x 90.2 x 49.2 millimeters including mounting features
  - Compact design makes it easier to locate the sensor on the vehicle without compromising vehicle styling
- High value
  - Produced using processes proven in high-volume manufacture of engine control units
  - Proven manufacturing processes increase affordability for high-volume automotive segments where radar systems have not previously been available

The radar module, including electronics, measures just 173.7 x 90.2 x 49.2 millimeters including mounting features.





Description		Jetson TX2 System-on-Module*
Pascal GPU [0]		
256-core GPU | End-to-end lossless compression | Tile Caching | OpenGL® 4.5 | OpenGL® ES 3.2 | Vulkan® 1.0 | CUDA® 8.0 | GPGPU		
Maximum Operating Frequency		1.12GHz
CPU Complex ‡		
ARMv8 (64-bit) heterogeneous multi-processing (HMP) CPU architecture; two CPU clusters (6 processor cores) connected by a high-performance coherent interconnect fabric. NVIDIA Denver 2 (Dual-Core) Processor: L1 Cache: 128KB L1 instruction cache (I-cache) per core; 64KB L1 data cache (D-cache) per core | L2 Unified Cache: 2MB ARM® Cortex® -A57 MPCore (Quad-Core) Processor: L1 Cache: 48KB L1 instruction cache (I-cache) per core; 32KB L1 data cache (D-cache) per core | L2 Unified Cache: 2MB		
Maximum Operating Frequency per Core NVIDIA Denver 2 ARM Cortex-A57		 2.0GHz 2.0GHz
HD Video & JPEG		
Video Decode(Number of Streams Supported): H.265 [11]: Main 10, Main 8 H.265 [11]: Main 444 H.264 [11]: Baseline, Main, High H.264 [11]: MVC Stereo (per view) VP9 [11]: Profile 0 (8-bit) and 2 (10 and 12-bit) VP8: All MPEG1/2: Main MPEG4: SP/AP VC1: SP/MP/AP		 (2x) 2160p60 | (4x) 2160p30 | (7x) 1080p60 | (14x) 1080p30 2160p60 | (2x) 2160p30 | (3x) 1080p60 | (7x) 1080p30 (2x) 2160p60 | (4x) 2160p30 | (7x) 1080p60 | (14x) 1080p30 2160p60 | 2160p30 | 1080p60 | 1080p30 (2x) 2160p60 | (4x) 2160p30 | (7x) 1080p60 | (14x) 1080p30 2160p60 | (2x) 2160p30 | (4x) 1080p60 | (8x) 1080p30 2160p60 | (2x) 2160p30 | (4x) 1080p60 | (8x) 1080p30 (4x) 1080p60 | (8x) 1080p30 (2x) 1080p60 | (4x) 1080p30
Video Encode (Number of Streams Supported): H.265 H.264: Baseline, Main, High WEBM VP9 WEBM VP8		 2160p60 | (3x) 2160p30 | (4x)1080p60 | (8x) 1080P30 2160p60 | (3x) 2160p30 | (7x) 1080p60 | (14x) 1080p30 2160p30 | (3x) 1080p60 | (7x) 1080p30 2160p30 | (3x) 1080p60 | (6x) 1080p30
JPEG (Decode & Encode)		600 MP/sec
Audio Subsystem		
Industry-standard High Definition Audio (HDA) controller provides a multi-channel audio path to the HDMI interface | 4 x I2S | DMIC | DSPK | 2 x I and Q baseband data channels | PDM in/out		
Display Controller Subsystem		
Support for DSI, HDMI, DP and eDP | Two multi-mode eDP/DP/HDMI outputs.		
Captive Panel		
MIPI-DSI (1.5Gbps/lane)	Max Resolution	Support for Single x4 or Dual x4 links | 2560x1600 at 60Hz
eDP 1.4 (HBR2 5.4Gbps)	Max Resolution	3840x2160 at 60Hz
External Display		
HDMI 2.0a/b (6Gbps)	Max Resolution	3840x2160 at 60Hz
DP 1.2a (HBR2 5.4 Gbps)	Max Resolution	3840x2160 at 60Hz
Imaging System		
Dedicated RAW to YUV processing engine process up to 1.4Gpix/s | MIPI CSI 2.0 up to 2.5Gbps (per lane) | Support for x4 and x2 configurations (up to 3 x4-lane or 6 x2-lane cameras)		
Clocks		
System clock: 38.4 MHz | Sleep clock: 32.768 KHz | Dynamic clock scaling and clock source selection		
Boot Sources		
Internal eMMC and USB (recovery mode)		

Description	Jetson TX2 System-on-Module*
Security	
Secure memory with video protection region for protection of intermediate results | Configurable secure DRAM regions for code and data protection | Hardware acceleration for AES 128/192/256 encryption and decryption to be used for secure boot and multimedia Digital Rights Management (DRM) | Hardware acceleration for AES CMAC, SHA-1, SHA-256, SHA-384, and SHA-512 algorithms | 2048-bit RSA HW for PKC boot| HW Random number generator (RNG) SP800-90 | TrustZone technology support for DRAM, peripherals | SE/TSEC with side channel counter-measures for AES | RSA-3096 and ECC-512/521 supported via PKA	
Memory ††	
128-bit DRAM interface | Secure External Memory Access Using TrustZone Technology | System MMU	
Memory Type	4ch x 32-bit LPDDR4
Maximum Memory Bus Frequency (up to)	1866MHz
Memory Capacity	8GB
Storage	
eMMC 5.1 Flash Storage	
Bus Width	8-bit
Maximum Bus Frequency	200MHz (HS400)
Storage Capacity	32GB
Connectivity	
WLAN	
Radio type	IEEE 802.11a/b/g/n/ac dual-band 2x2 MIMO
Maximum transfer rate	866.7Mbps
Bluetooth	
Version level	4.1
Maximum transfer rate	3MB/s
Networking	
10/100/1000 BASE-T Ethernet | IEEE 802.3u Media Access Controller (MAC) | Embedded memory	
Peripheral Interfaces Δ	
XHCI host controller with integrated PHY: (up to) 3 x USB 3.0, 3 x USB 2.0 | USB 3.0 device controller with integrated PHY | 5-lane PCIe: two x1 and one x4 controllers | SATA (1 port) | SD/MMC controller (supporting eMMC 5.1, SD 4.0, SDHOST 4.0 and SDIO 3.0) | 5 x UART | 3 x SPI | 8 x I^2C | 2 x CAN | 4 x I2S: support I^2S, RJM, LJM, PCM, TDM (multi-slot mode) | GPIOs	
Operating Requirements ♦	
Temperature Range	-25C – 80C
Module Power	7.5W (Max-Q) / 15W (Max-P)
Power Input	5.5V – 19.6V
Applications	
Intelligent Video Analytics, Drones, Robotics, Industrial automation, Gaming, and more.	

* Refer to the software release feature list for current software support.

◊ GPU Maximum Operating Frequency: 1.3GHz supported in boost mode.
Product is based on a published Khronos Specification and is expected to pass the Khronos Conformance Process. Current conformance status can be found at www.khronos.org/conformance.

‡ CPU Maximum Operating Frequency: 1-4 core = up to 2.0GHz; greater than 4 core = up to 1.4GHz

(††) For max supported number of instances: bitrate not to exceed 15 Mbps per HD stream (i.e., 1080p30), overall effective bitrate is less than or equal to 240 Mbps

†† Dependent on board layout. Refer to Interface Design Guide for layout guidelines.

Δ Refer to the Interface Design Guide and *Parker Series SoC Technical Reference Manual* to determine which peripheral interface options can be simultaneously exposed.

♦ Refer to the *Jetson TX2 OEM Product Design Guide* and *Jetson TX2 Thermal Design Guide* for evaluating product power and thermal solution requirements. See the software documentation for information on changing the default power mode (default: Max-P).