CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

High-speed Image Capture and Cone Detection Using a Raspberry Pi and Camera Module: A Design Science Approach

Bachelor of Science Thesis in Software Engineering and Management

JUSTINAS STIRBYS

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

High-speed Image Capture and Cone Detection Using a Raspberry Pi and Camera Module: A Design Science Approach

Supervisor: Christian Berger
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

High-Speed Image Capture and Cone Detection Using a Raspberry Pi and Camera Module: a Design Science Approach

Justinas Stirbys
Software Engineering and Management Program
Department of Computer Science and Engineering
University of Gothenburg
Göteborg, Sweden
E-Mail: gusstirju@student.gu.se

***Abstract*—The purpose of this paper is to provide a high-speed image capturing and cone detection algorithm using low-cost devices, such as a Raspberry Pi and Pi NoIR V2 camera module. Little research focuses on high-speed, low-cost image processing. Most existing literature aimed at achieving a high FPS, use expensive tools, such as LiDARs or high-speed cameras. Thus creating a gap in knowledge. Design science was used to develop and evaluate our artifact. For evaluation, controlled experiments were used to gather data. The collected data shows that our artifact is able to capture and detect cones at 30-35FPS with varying, 42-83%, cone detection rate and cone detection accuracy ranging from 9% to 69%. The results show that low-cost, real-time image capture and processing are achievable. The study could be used to provide a cost effective solution for cone detection. Additionally, the solution's low-cost provides more opportunities to develop and innovate in this area.**

I. Introduction

A. Background

Image processing needs are increasing in many key areas, including the automotive industry [1]. The automotive industry widely utilizes cameras and image processing as sensors. Most commonly to improve safety by using lane, lane marking or vehicle detection algorithms. There are a variety of existing solutions, which have varying frame per second (FPS) image processing capabilities. With most of the existing work being aimed at real-time processing. In the context of this paper, real-time frame rate is considered to be roughly 25FPS, while high-speed image processing is over 100FPS. 10FPS can be achieved with a multi-camera setup and a lane detection algorithm [2]. Solutions focusing on lane and vehicle tracking can achieve 11FPS [3] or in other cases 20FPS [4]. Other algorithms analyzing video sequences to detect and track lane markings are able to achieve a frequency of 30FPS [5]. Image-based solutions have seen a rise in interest and opportunities, due to faster and cheaper computer processing [1]. A reduction in cost can be achieved by using lower cost equipment, such as a Raspberry Pi. Raspberry Pis in the automotive industry have been adapted for vehicle detection, counting and tracking [6].

B. Motivation

The motivation for this research paper is to provide an affordable solution for image capture and cone detection that could be adapted to the automotive industry. The motivation for cone detection, lies in their use to direct traffic and functions as warning signs. By providing an artifact capable of cone detection, it is possible to improve safety in vehicles. For fast traveling vehicles, a high frame rate capture and image processing are required to keep track of the vehicle's surroundings. This is usually solved with LiDARs, high-speed cameras or other expensive systems. The use of expensive systems is reflected in existing literature as well. LiDAR and radar sensors have been used to mask out other cars and objects [2]. Additionally, processing commonly is done with dedicated personal computers (PC) and expensive CPU, such as Intel Core 2 Duo E8400 CPU [7], Pentium IV 2.8GHz [5], Pentium IV 2.8GHz [3] etc.

Expensive equipment limits the number and type of people that can work with high-speed image capture and processing. Studying the use of low-cost devices to analyse a high amount of frames per second, could make research, in this area, more accessible to individuals and smaller organizations. Thus, providing a means for further innovation. However, high-speed, low-cost image capture and analysis has hardly been researched. The research found is not applied to the the automotive industry. Instead, the research focuses on baseball pitch movements [8] and video capture [9]. Thus, indicating a gap in knowledge.

The aim of this paper is to fill the gap in literature by providing a low-cost solution for high-frequency image capturing and processing using cheap, affordable tools such as a Raspberry Pi. The solution is meant to detect cones.

C. Research Questions

There are several predominant trends in vehicle production. Within Europe, awareness of pedestrians and pedestrian safety are several of the major factors to consider [10]. Additionally, savings and safety are some of the drivers influencing autonomous vehicle development [11]. Safety concerns require

vehicles not only to make quick decisions, but for the decisions to be accurate as well. With safety concerns in mind, the following research questions were made:

- **RQ1:** What frame rate can be achieved at maximum when performing cone detection with a low-cost Raspberry Pi and camera setup?
- **RQ2:** What performance can be achieved in terms of accuracy, cone detection rate and number of false positives, and false negatives detected?

To answer these RQs, design science will be used as a research approach, Dynamic analysis and a controlled experiment will be used as evaluation methods. The evaluation will use a Raspberry Pi 3 and Pi NoIR V2 camera. Dynamic analysis will check the time-taken for processing by the artifact's different components, seen Figure 3. For the experiment, the camera module will be placed at different distances in front of a traffic cone. The experiment will be used to calculate frame rate and accuracy. Accuracy is a percentage calculated by the number of frames that correctly detected cones, divided by the number of total captured frames. For a cone to be correctly detected the absolute difference between the cones' real and detected position is less than 5cm. Frame capture and processing speed is calculated by observing the number of correctly detected cones.

D. Contributions

The designed artifact provides contributions to researchers, practitioners and software engineering as a whole. First, researchers may benefit by expanding existing knowledge with a paper on high-speed frame capture and processing on low-cost devices that is applied to the automotive industry. As there is little research on this specific topic, the study will be one of the first to fill this knowledge gap. Second, practitioners may benefit by obtaining a solution for a cheaper, more affordable way of performing cone detection. Thus, reducing the cost of production, while improving performance of the vehicle. Additionally, the proposed solution is develop with low-cost components, open-source libraries and is non-proprietary. This enables a larger amount of people to work with high-speed image capture and processing. Thus, providing greater accessibility and bolstering further innovation for software engineering as a whole. Software engineers are provided with the artifact's design and flow of events, Figures 2 and 3, as a starting basis for their own innovations using Raspberry Pis.

E. Scope

The developed artifact is an algorithm consisting of two main parts. The first, is responsible for high-speed frame capture. This part of the algorithm is realized by integrating existing open-source libraries with the second part. The second part uses captured frames for cone detection. A cone detection algorithm is produced as part of this research project. The designed artifact is developed specifically for traffic cone detection. Therefore, recognizing other objects, such as, traffic signs or cars are outside the thesis scope.

F. Structure

The rest of the report is structured as follows; Section II details similar research done in this area; Section III describes the chosen research approach, data collection/analysis and the study's design cycles; Section IV describes the results related to the research questions; Section V discusses research implications and validity threats; Section VI describes future work and summarizes the paper.

II. Literature Review

A. Image Capture & Processing in Automotive Industry

The automotive industry incorporates a wide-range of image sensors. These sensors are often used for lane, lane marking and vehicle detection. Research on lane marking detection and lane tracking [2], [5] showed results with varying FPS. One solution achieved 10FPS using a 4 camera setup and a 2GHz Pentium with a GeForce 7600GTS graphics card [2], while 30FPS was achieved with a single camera and a PC (Pentium IV 2.8GHz and 512M of RAM) for processing [5]. Additionally, Sivaraman and Trivedi [3], and Wu [4] performed lane detection and expanded their solutions to include vehicle detection. Results provide 11FPS with Pentium i7 2.4GHz PC [3] and on average 20FPS with a Pentium Duo 1.66GHz CPU and 1GB RAM computer setup [4]. Gupta and Choudhary further expanded their research to perform lane detection, lane tracking and road surface marking detection. Using different data sets that included road images under different conditions and a PC (Intel Core 2 Duo E8400 CPU with 8GB of RAM) for processing, their solution achieved between 22 and 46FPS [7].

Although existing research attempts to solve similar problems, their approaches vastly differ. One paper approached lane marking detection by using Canny edge detection [3]. Other solutions created and used ground-plane images mapped to real world coordinates [2], [4]. Lipski et al. used an IMU and GPS to find the vehicles position and rotation, then created a single top-down fusion image in HSV [2]. Wu et al. used a calibrated camera to map 3D points to pixels using the pinhole model. Then performed linear transformation to map the image plane projections to a ground-plane images [4]. To identify lane markings and create road models, Wu et al. used steerable filters [4]. While Lipski et al. examined the contrast, shape and orientation of pixels [2]. Both papers then used multiple iterations of a RANSAC algorithm for lane fitting [2], [4]. Liu et al. [5], Gupta and Choudhary [7] both defined images Regions of Interest (ROI). ROI is a part of an image that is being analysed. For analysis, the aforementioned authors defined ROIs, where the lanes are most visible. Meaning, the lower half of an image. Analysing a part of an image, can improve the systems speed and efficiency [7], because a whole image contains too much data to analyze in real-time [5]. Both papers use segmentation on grayscale images. Segmentation makes the continuous lane markings easier to observe [5]. However, Liu et al. used P-tile method to determine the segmentation threshold [5], while Gupta

and Choudhary [7] used an adaptive threshold. Furthermore, Gupta and Choudhary used a Gaussian filter to reduce the noise in ROIs before segmentation. Liu et al. describe this as a possibility, but did not perform filtering [5]. The two solutions further differ with the way lane fitting was done. Liu et al. used lane markings to construct linear equations that represent the lane [5]. Gupta and Choudhary approached lane fitting by forming and analyzing clusters across multiple frames. The clusters were made of points representing lane or road markings. The points were used to predict a path for the vehicle. If the cluster of the succeeding frame did not match the cluster of preceding frame a new cluster is formed.

However, the existing literature fails to provide a solution to the studys problem. First, because the existing solutions are not focused on high-speed frame capture. Second, the solutions use PCs for processing, which are considerably expensive. Third, the solutions used an expensive LiDAR and radar sensors to mask out other cars, walls etc [2].

B. Image Capture & Processing with Raspberry Pis

Raspberry Pis are cheap single-board computers. Raspberry Pis have been used for varying purposes, from face recognition [12] to motion detection in surveillance system [13]. Raspberry Pis are affordable, portable, small, light and have a lower power consumption [12], which could explain their popularity. Raspberry Pis popularity extends to the automotive industry as well, to such areas as traffic monitoring.

Kochláň et. al utilized background subtraction [14]. The Raspberry Pi was set up on a bridge. The Pi would capture images of ongoing traffic. Moving objects were detected by calculating the difference between the captured frame and an example background frame. After subtraction a threshold is applied, to achieve a black and white resulting image. Then to further remove noisy pixels erosion and dilation were applied [14].

M. Anandhalli and V. P. Baligar have opted to use a vastly different approach for traffic monitoring. M. Anandhalli and V. P. Baligar used a camera capturing in RGB, before converting frames to HSV for a more absolute vehicle colour segmentation [6]. In contrast to Kochláň et. al who used grayscale images for processing [14]. To continue, M. Anandhalli and V. P. Baligar would split the HSV image into three channels and process the channels separately, then merge the processing results to blobs representing the detected vehicles [6]. Another difference between M. Anandhalli and V. P. Baligar and Kochláň et. al, is the scope of their respective solutions. M. Anandhalli and V. P. Baligar use Kalman filter to also track the vehicles [6] and not only detect them, as is the case with Kochláň et. al. However, despite the vastly different approaches the resulting accuracy is similar. With M. Anandhalli and V. P. Baligar having a 96% detection accuracy [6] and Kochláň et. al having 95.7% and 93.2% detection accuracy for traffic volume and vehicle detection respectively [14].

Traffic monitoring implementations using a Raspberry Pi have also been performed with edge detection algorithms. L. Ujjainiya and M. K. Chakravarthi created an experiment in hopes of improving vehicle safety. To improve safety, images of vehicles were captured and processed using Laplacian, Sobel and Canny edge detection. The different edge detection algorithms were the focus of an analytical comparison to determine the best algorithm for vehicle detection. L. Ujjainiya and M. K. Chakravarthi concluded that Canny edge detection provided the most distinguishable objects and best depth analysis for the image out of the three options [15].

The aforementioned research projects do not provide a solution for this thesis, as they are focused on detecting vehicles as opposed to traffic cones. Moreover, the solution are aimed for real-time detection, roughly 25FPS, as opposed to high-speed detection, over 100FPS.

C. High-Speed, Low-Cost Cameras

Most existing solutions utilize expensive high-speed cameras to achieve a high FPS. Similarly, minimal research was found covering high-speed image capture on low-cost devices. Literature on this area focuses on tracking high-speed movement and is not applied to the automotive industry. Wilburn et al. used cheap CMOS sensors to simulate a high-speed camera. For the simulated camera the study used a cluster of 52 Omnivision OV8610 image sensors, each capturing at 30 FPS. Furthermore, Wilburn et al. required 1 PC per 26 cameras to capture the compressed video [9]. Theobalt et al. used affordable cameras to track a baseball pitch and the pitchers hand movements. The study used 4 Olympus Camedia C505 cameras, stroboscopes for lighting and 1 PC to analyze and 3D model the high-speed ball movements [8].

These papers used different camera arrangements to track high-speed movements. Wilburn et al. created a camera cluster. The cameras were all closely situated and captured, at least partly, the same area [9]. Theobalt et al. arranged the 4 cameras along the baseball pitches throwing path [8]. Despite differences in camera placements both papers had similar issues with light and exposure time. Exposure time creates a trade-off. Shorter exposure reduces motion blur, but captures less light. Longer exposure captures more light, but with more motion blur [9]. However, using a camera cluster provides greater flexibility to manipulate exposure time. Additionally, the camera cluster can be used to greatly supplement image capture speed, even up to 1560 FPS [9].

Despite that, the found literature does not provide a solution to the studys problem for a variety of reasons. First, the literature is not applied to vehicles or used for vehicle related uses. Second, the research only shows high-speed frame grabbing an no image processing. It either solely performed videography [9] or did not do 3D modeling of baseball trajectories in real-time [8]. Third, research was performed with old cameras [8] that are no longer sold by retailer. Therefore, the research on this topic needs an update.

D. Object Detection with CMOS sensors

An alternative to object detection with high-speed, low-cost cameras is provided by CMOS image sensors. CMOS sensors

are electronic chips used to create images in digital cameras. An alternative to CMOS chips are CCD chips. However, this section will focus solely on CMOS, since CMOS sensors are better equipped for high-speed imaging than CCDs [16]. Additionally, CMOS are more reliable, can be easier integrated and with a slightly lower production cost than CCD sensors [17].

Although CMOS are single electronic chips, they have been successfully used for object detection and tracking. Therefore, CMOS may be beneficial for our research topic, as CMOS could be used for cone detection and tracking. The research papers discussing CMOS use binary images for image processing to save computation speed. Additionally, to further reduce computational power all papers incorporate ROIs. Zhao et. al uses frame subtraction to generate binary motion events [18]. The same technique, frame subtraction, is used by Choi et. al to detect motion in images [16]. Moving objects in images are identified by subtracting the current frame by an reference frame or the previous frame. In contrast to the aforementioned solutions, Teman et. al identifies objects by detecting the brightest targets in the CMOS field of view [19]. Once the objects are identified, Choi et. al and Zhao et. al transfer the digitized output image to an external digital control unit to find a ROI for the image. ROI is then used for object localization and tracking [16], [18]. Teman et. al first calculates the centroid of the brightest pixels, before using an external control unit to calculate the ROI [19].

The solutions differ in the way object tracking is performed. Zhao et. al forms pixel clusters within ROIs that represent moving objects. Once a new motion event occurs in a frame, a new cluster of pixels is created. Zhao et. al finds the clusters' centroids, then distance between the clusters. The distance is used to decide whether the different clusters represent the same object. Meaning that if the distance is small, the clusters are merged. If the clusters overlap, one of the clusters is enlarged. If the distance is large, then the clusters represent different moving objects [18]. Teman et. al and Choi et. al performed object tracking by checking the distribution of pixels in the target object. The distribution was checked by an external control unit. The unit would determine the movement direction. If movement occurred the ROI area would be moved accordingly in the detected direction to keep track of the target [16], [19]. The difference between Teman et. al and Choi et. al, is that Choi et. al reduces spatial resolution in in ROI to perform processing at a much higher frequency. Additionally, Choi combines ROI images and stationary images to create a multi-resolution images to suppress motion blur [16].

The above literature provide fast solutions for object detection and tracking. Teman et. al and Zhao et. al providing a frame rate of up to 100FPS [18], [19], and Choi et. al capturing normal images at under 30FPS, but lower spatial resolution (ROI) images at 240FPS [16]. However, the solutions do not provide an answer to our research topic, due to small frame size, 64x64 [16], [18], [19] making it difficult to monitor the surroundings of the vehicle. Additionally, the use binary images and frame subtraction captures all motion in frame. Meaning that additional filtering out of objects outside the road boundaries would be required.

III. Methodology

A. Research Approach

A research approach describes broad steps for data collection and analysis required to fulfill the research goal. The goal of this thesis is to produce a new artifact for cone detection. Meaning, that the goal is to address a real-world problem. Action Research (AR) and Design Science Research (DSR) are two seemingly similar methodologies focused on addressing real-world problems. AR is a framework attempting to contribute to both practitioners and academia by solving an immediate problematic situation. DSR focuses on designing, analysing and measuring an artifact's performance [20]. DSR puts artifacts at the core of the discipline [21] and by definition includes creating new artifacts unlike AR [20]. Therefore, the goal of this thesis may be be better realized with DSR.

The result of DSR is an artifact created to address an important organizational problem. Artifacts can be ideas, practices, technical capabilities and products [22]. The paper methodology was based on existing literature. Specifically, the 6 activity design science methodology proposed by Peffers et. al (2007) to better ensure successful artifact creation. The activities described by Peffers et. al are suggestions and function as a template. The activity sequence can be changed, activities can be omitted or replaced by others [23]. For this project, the activity of defining specific goals for the solution was omitted. The omission was done, due to the project short time frame and small size. Performing activities, such as, a full software requirement specification would have consumed a large amount of time. Instead, only several simple requirements were made for each Design Cycle, see following subsections. These requirements guided development and data analysis. Additionally, the demonstration activity, used to showcase the artifact working, has been omitted in favor of the more formal evaluation activity [23]. The evaluation activity demonstrates the artifact's capabilities and provides concrete measurements. These activities have a significant overlap, so demonstration as a standalone step was deemed unnecessary. The activities used in the project are:

- **Problem identification and motivation.** The purpose of this activity is to define a specific problem. Then further justify the need for a solution. This activity involves studying existing literature. The gathered literature is seen in Section II.
- **Design and development.** The purpose of this step is to design and develop the artifact. The artifact's designs are discussed below.
- **Evaluation.** This step requires observing the artifact and measuring how well the artifact meets the established goals. For this project dynamic analysis and a controlled experiment were selected as evaluation methods. The evaluation results are described in Section IV.
- **Communication.** The final step requires to communicate the findings of DSR to relevant audiences. Meaning, com-

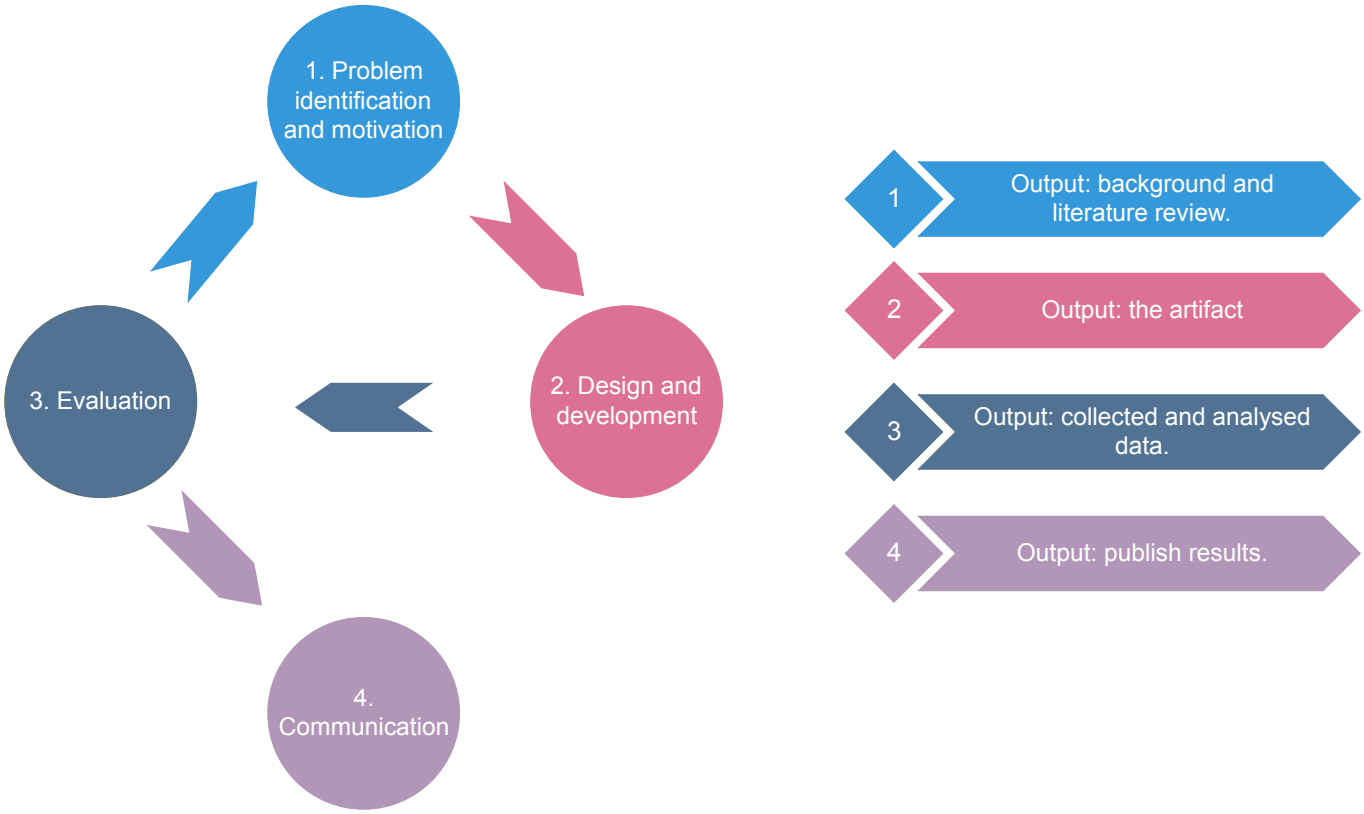

Fig. 1: Design Science Cycle

municating the problem and its importance, the design, the artifact and its effectiveness. This is done through by handing over the report to share the project's details.

These activities are performed in design cycles and can be seen in Figure 1.

B. Data Collection

Data was collected in two ways. For the first design cycle, dynamic evaluation was used to monitor the artifact's performance, while capturing and processing images of the cone. For dynamic analysis the time taken for Canny edge detection and Progressive Probabilistic Hough Transform was monitored for 100 evaluations.

For the second design cycle a controlled experiment was used. A controlled experiment measures performance of an artifact under controlled conditions [22]. For an experiment it is important to specify dependent variables, factors, levels, control variables and objects. A dependent variable is the measured outcome of the experiment. A factor is a characteristic that has an effect on a dependent variable. Levels are an extension of factors, specifically a level refers to a possible value of a factor. A control variable refers to an attribute or characteristic that will remain constant throughout the experiment. Lastly, an experimental object is an object on which the experiment is applied [24].

For this project, the experiment has 3 dependent variables. The dependent variables are number of frames grabbed, the frequency of analysed images i.e. the FPS, and cone detection accuracy. Cone detection accuracy is measured in percentages, calculated as the number of correctly detected cones divided by the number of total frames captured. For cones to be accurately detected, the difference between the detected cone position and the real-life position is not greater than 5cm. The factor in this experiment is the distance of the cone from the track. This factor's levels are 1m, 2m and 3m. The constant control variables are the Raspberry Pi and camera module (Pi NoIR V2), the frame size (640x240), the surrounding environment, lighting conditions and the same orange cone. An orange cone was used, since orange traffic cones are the standard colour used to direct traffic. Lastly, the experiment's object is the produced artifact.

C. Design Cycle 1

For the first design cycle the FPS goal was set to approximately 10FPS, see goals listed below. This FPS goal was used to see if cone detection running on a Raspberry Pi can match existing solution that achieved 10FPS [2] or 11FPS [3]. Picamera library was used for RBG image capture. The maximum number of frames grabbed by Picamera is 90 (120 if overclocked) per second for the Raspberry Pi V2 camera module [25]. Cone detection and image processing were done with OpenCV. This choice was motivated based on an

abundance of available sources on OpenCV. OpenCV was used to convert images from RGB to grayscale, then to perform Canny edge detection to extract features. OpenCV provides two separate implementations of Hough Transform; Standard Hough Transform and Progressive Probabilistic Hough Transform (PPHT). Computation-wise, PPHT is a less expensive Hough Transform implementation. It works with finite lines, as a opposed to General Hough Transfrom. It also takes in parameters, such as, minimal line length and max gap between points, to filter out weak line candidates. Thus, reducing the number of lines detected and saving computation costs. The artifact implements PPHT to extract possible lines representing cones from edge images.

The first Design Cycle functioned as a prototype or proof of concept. The goals were used to set a baseline for measurements that could be used at later stages of the project. The Design Cycle 1 goals are:

- **G0:** Use low-cost equipment. Project cost does not exceed 100 Euros.
- **G1.1:** Artifact functions at at least 10FPS.
- **G1.2:** Grayscale conversion does not exceed 5ms.
- **G1.3:** Canny edge detection does not exceed 45ms.
- **G1.4:** PPHT does not exceed 50ms.

D. Design Cycle 2

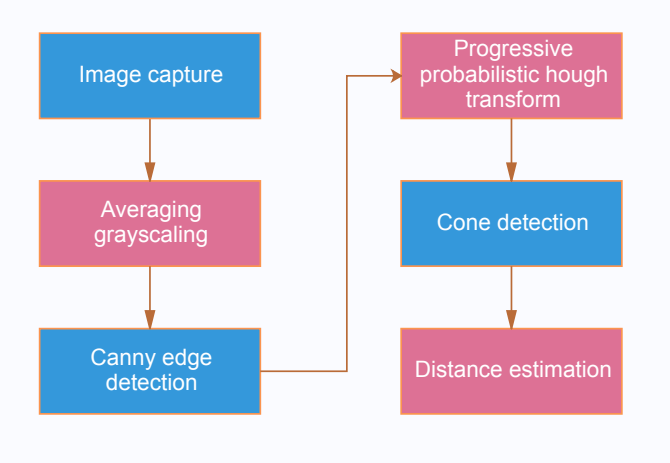

Fig. 2: Artifact Design

The designed artifact during Design Cycle 2 consists of several steps, pictured in Figure 2. Several changes were made in contrast to the first design cycle. First, image capture is done using an open-source library "raspiraw". For Raspberry Pi 3 raspiraw can provide image capture up to 750FPS with a resolution of 640x64 [26]. Replacing Picamera with raspiraw provides faster image capturing, meaning that if the artifact's processing performs well it will not be dependent on image capturing.

Second major change was removing OpenCV. OpenCV is a large and robust library. Removing OpenCV library, means removing some unnecessary components that could slow down the artifact's performance. To improve performance an averaging grayscale method was applied to the RAW images outputted by raspiraw instead of using OpenCV. Grayscale images are still used for Canny edge detection to extract image features. For feature extraction the decision to use Canny edge detection was based on gathered literature. Canny edge detection provides more distinguishable objects and better depth analysis than Sobel or Laplacian operators [15]. Under many evaluation scenarios Canny edge detection performs better that Sobel, Prewitt and Robert's operators. However, this also means Canny is more computationally expensive [27], [28].

The result of the Canny edge detection algorithm is an image containing strong edge pixels forming lines. Analysing these lines, using PPHT, to detect cones is the next step. PPHT is less costly and is more suited for real-time applications than standard Hough Transform [29]. The third change from Design Cycle 1, is addition of linear algebra. Linear algebra is used on lines fitting PPHT's parameters to determine if the lines form a cone. If the lines are deemed to belong to a cone, the distance to a detected cone is estimated using a ratio between the cone's real-world size and the detected size.

The goals for Design Cycle 2 are outlined below. The first major goal is to achieve real-time image capture and processing. The artifact was expanded with cone distance estimation. Several new goals were made for this part of the artifact. The gathered literature showed that some of the solutions managed to achieve over 90% accuracy [3], [14] for vehicle detection. This is the first iteration for distance estimation. Therefore, the accuracy goal for cone detection was set at 75% to act as a baseline, for possible future improvements. Given this accuracy goal and the goal for real-time processing, the number of frames with faulty detected cones can be 6FPS without violating the aforementioned goals. Therefore, the goal for false positives and false negatives was set to at most three per second. The Design Cycle 2 goals:

- **G0:** Use low-cost equipment.
- **G2.1:** Artifact can function at real-time. Image capture and processing is at least 25FPS.
- **G2.2:** Number of false negatives for cone detection does not exceed 3.
- **G2.3:** Number of false positives for cone detection does not exceed 3.
- **G2.4:** Cone detection accuracy is at least 75%.

IV. Results

The developed artifact's components are shown in a component diagram, seen in Figure 3. The artifact uses a blackboard architectural style. The style consists of several concepts, specifically the blackboard, knowledge-sources and a controller. The "blackboard" is a central knowledge store, while the "knowledge-sources" are used to read and write from/to the blackboard. The artifact's blackboard is a library referred to as Multimedia Abstraction Layer (MMAL). The MMAL library is used by by raspiraw to retrieve information of the captured frames that are stored on the Raspberry Pi. MMAL provides

easy access to images for the rest of the artifact [30]. Therefore MMAL's side of interactions with the rest of the applications are expressed through provided interfaces. Several knowledge-source interact with MMAL. Their side of the interactions are expressed with required interfaces, since they receive and use data from MMAL. Raspiraw interacts with MMAL by monitoring the changes made to the stored information. In this scenario raspiraw functions as a "controller" to the blackboard (MMAL). As a controller, raspiraw tells which of the knowledge-sources to execute.

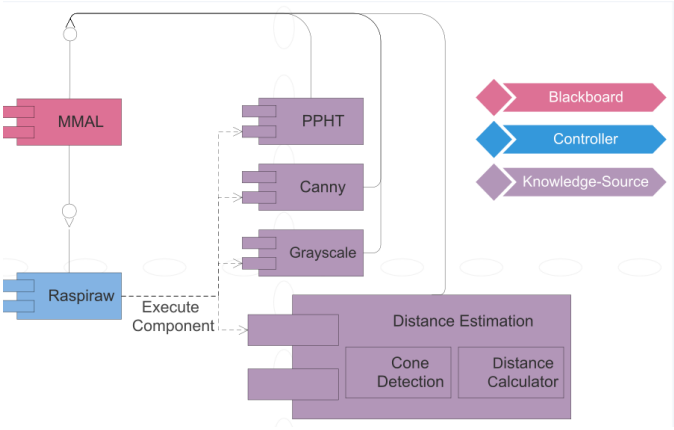

Fig. 3: The Artifact's Component Diagram

The artifact's results for Design Cycle 1 can be seen in Appendix A. These results are used to show the processing time of 100 evaluations of the artifact developed using OpenCV. With the OpenCV implementation the total processing time for detecting a cone ranges from 0.037 to 0.221 milliseconds. The mean processing time is 0.105 milliseconds, making the mean FPS roughly 9 frames per second. Meaning, that G1.1 was not reached.

The primary impediment, based on Design Cycle 1 results, is processing time. To improve processing time, Design Cycle 2 removed OpenCV. OpenCV is a large and robust library that provides more features than necessary for cone detection. Removing OpenCV can streamline the artifact. Additionally, maximum FPS for the Picamera library is 90 or 120 frames, if overclocked [25]. This could interfere with the artifact's frame processing. For instance, in case a sufficient improvement in processing is made, the artifact may have to remain idle while new frames are captured. Therefore, for Design Cycle 2 Picamera was replaced with raspiraw.

TABLE I: Experiment's Frame Rate Results

Distance (m)	Trial Nr.	FPS
1	1	35
1	2	35
1	3	36
1	4	36
1	5	35
2	1	33
2	2	33
2	3	32
2	4	33
2	5	33
3	1	31
3	2	31
3	3	31
3	4	30
3	5	31

The collected data from Design Cycle 2 is in Appendix B. The RAW image files that were processed can be found on Google Drive[1]. The analysed data from the experiment can be seen in Table I and Table II. The artifact was able to achieve approximately 35-36FPS at one meter, 32-33FPS at two meters and 30-31FPS at three meters, with 640x240 image resolution. Raspiraw is able to capture 180 frames per second on Raspberry Pi 3 [26]. Meaning, G2.1 goal for real-time image capture and processing was achieved.

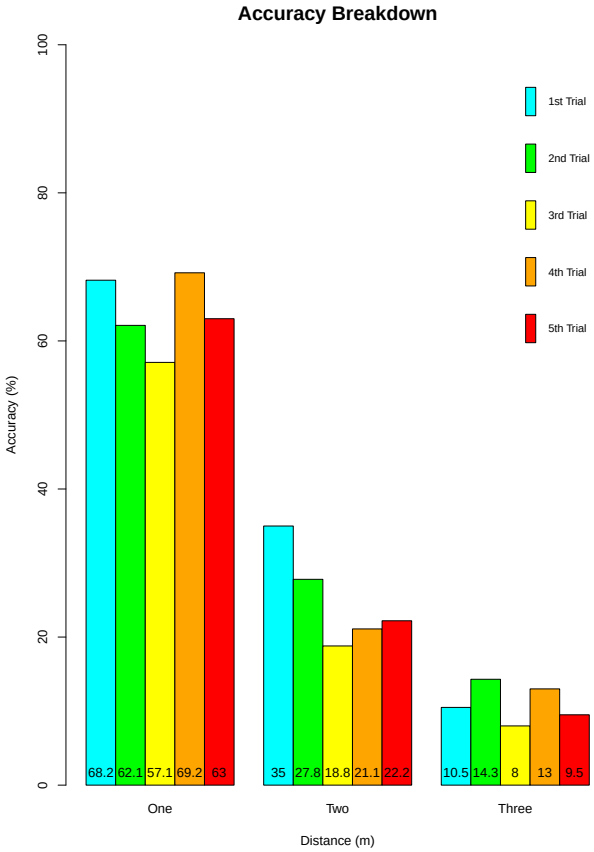

Fig. 4: Accuracy During Experiments

TABLE II: Experiment's Accuracy Results

[1]RAW images from experiment: https://drive.google.com/drive/folders/16zybMfmIkhzDwYeIVvn4F0d_tDXSvVR6?usp=sharing

Distance (m)	Trial Nr.	Accuracy (%)	FP	FN	Detection Rate (%)
1	1	68.2	0	13	62.9
1	2	62.1	0	6	82.9
1	3	57.1	0	8	77.8
1	4	69.2	0	10	72.2
1	5	63.0	0	8	77.1
2	1	35.0	0	14	57.6
2	2	27.8	1	15	51.5
2	3	18.8	1	16	46.9
2	4	21.1	2	14	51.5
2	5	22.2	1	18	42.4
3	1	10.5	0	12	57.6
3	2	14.3	0	10	67.7
3	3	8.0	1	6	77.4
3	4	13.0	1	7	73.3
3	5	9.5	1	10	67.7

Table II shows the distance to cone from the Pi NoIR camera module, the trial number, the processed frames (FPS) and cone detection accuracy expressed as percentages. For a cone to be accurately detect, the detected distance to the cone must equal the real distance ± 0.05. 0.05 (5cm) was selected as an error margin, due to its common use in scientific experiments and statistics. Based on the collected results it can be seen that accuracy decreases as distance increases. The accuracy never reaches our set goal, with the highest accuracy being roughly 69%, seen in Figure 4. The decreasing trean in accuracy can be explained by the way distance to cone is calculated. The calculation is based on the ratio between the real-world object height and the object height in pixels. To calculate the estimated distance *d*, you require focal length *f*, real-world and pixel cone height $height_{CONE}$, image height $height_{IMAGE}$ and sensor height $height_{SENSOR}$. The calculation formula is:

$$d = \frac{f(mm) * height_{CONE}(mm) * height_{IMAGE}(px)}{height_{CONE}(px) * height_{SENSOR}(mm)}$$

TABLE III: Distance vs Possible Pixel Range

Distance (m)	Min Cone Height (px)	Max Cone Height (px)	Pixel Range
1	164	180	16
2	84	88	4
3	57	58	2
4	43	43	1
5	N/A	N/A	0.5

The decrease in accuracy is specifically caused by cone height pixel values. Figure 5 shows the cone height (px) ranges for each distance. If the pixel value of cone height is outside these ranges, then the detected distance exceeds the 0.05 error margin and the cone is not accurately detected. To accurately detected the cone, the cone height must be 164-180px for one meter, 84-88px for two meters and 57-58px for three meters. The pixel ranges are represented in Table III as

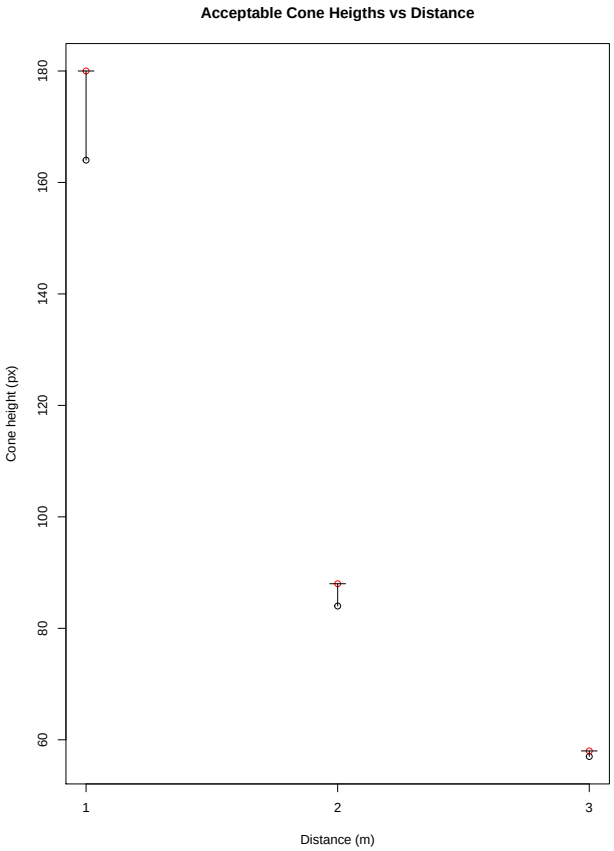

Fig. 5: Acceptable Cone Heights (px)

well. From the table it can be seen that for accurate detection, the possible cone height pixel ranges decrease exponentially. Meaning, that if the artifact would have been evaluated at 4 meters, the detected cone height would have to be exact to 1 pixel. Furthermore, the current formula for calculating distance would not return values, within the error margin, for distances equal or greater than 5 meters.

The number of False Positives (FP) and False Negatives (FN) in terms of cone detentions can be seen in Table II. False positives were found by checking if the detected distance exceeds a certain threshold. The threshold being 50cm, or half of the first distance examined. False negatives were calculated by subtracting number of false positives and detected cones from total processed frames. From Table II, it can be seen that the number of FP is at most 2 per experiment run. Therefore, G2.3 was reached. However, the same cannot be said for G2.2. Using our definition for FN, it can be seen that there are a large quantity of false negatives that greatly exceed our goal's threshold. The cone detection rate, expressed as a percentage, was calculated by dividing the the number of correctly detected cones by total number of frames processed, or FPS. From Table II it can be seen that cone detection rates vary, between 42% and 83%.

Lastly, during testing, the impact of lighting on the artifact was discovered. Image processing would be less effective

under lower lighting. The lack of light reduces the visibility of edges, making them less pronounced and smoother when processing images. Making it more difficult for PPHT to detect cones. Similarly, the edges become less pronounced as distance to the cone increases. Thus requiring an increase in exposure time for the camera module. With a larger exposure time, the background becomes more pronounced, requiring more time to process. Thus, the artifact's FPS decreases with distance.

V. Discussion

A. Research Question 1

To answer RQ1 the results show that a Raspberry Pi with a Pi NoIR V2 camera is able to achieve approximately 30-35FPS while detecting cones from 1 to 3 meters. The results show that high-speed, low-cost solution with the current artifact is not possible, due to large image processing time. However, a real-time, low-cost solution is achievable. The developed artifact has a higher FPS than previously mentioned lane and vehicle tracking solutions that managed to achieve 11FPS [3] and 20FPS [4] respectively. Our artifact's FPS is in the same range (30FPS) as Liu et. al who analyzed video sequences to detect and track lane markings [5]. If comparing solely FPS, our artifact is able to match existing solutions that use PCs for processing. However, these research projects used image datasets to evaluate their developed solutions under different road and weather conditions.

In comparison, our developed artifact lacks robustness and was evaluated in a controlled environment. Meaning, a direct comparison in performance cannot be made. Additionally, our solution under-performs in terms of FPS, when compared to solutions using CMOS, described in Section II-D, and the cluster based lane and road marking detection algorithm made by Gupta and Choudhary [7]. Using raspiraw [26] library it is possible to perform high-speed image capture using a Raspberry Pi and Pi camera module. However, the image capture speed is still lower than other low-cost solutions that managed to achieve 1560FPS videography. The 1560FPS solution has achieved by using a cluster of 52 cameras [9]. The cluster camera solution used more cameras to supplemented the individual camera speeds, whereas the proposed paper solution attempted to maximise a singular cameras speed.

B. Research Question 2

To answer RQ2, the developed artifact used a ratio based calculation to estimate distance to cones. The results show that the developed artifact is lacking in terms of accuracy. The detection rate of cones varies between 42% and 83%, which is significantly lower than existing solutions by M. Anandhalli and V. P. Baligar that has a 96% vehicle detection accuracy [6] and Kochláň et. al solution that has 95.7% and 93.2% detection accuracy for traffic volume and vehicle detection respectively [14]. Moreover, Sivaraman and M. M. Trivedi also managed achieved 93.2% lane localization accuracy [3].

Our developed solution achieved 57-68% cone detection accuracy at 1 meter, 18.8-35% at 2 meters and 8-14% at 3 meters. Additionally, the artifact has a high false negative count, meaning the artifact failed to identify present cones in frames. The low accuracy combined with a high false negative count, make the artifact unsuited for safety critical systems, such as vehicles. At least without future improvements.

C. Threats to Validity

Several limitations affecting generality and applicability of the thesis have been identified.

1) Dependability: Dependability refers to the research findings' consistency and repeatably [31]. The developed artifact implements PPTH. This particular variation of Hough Transform requires randomly selecting one of the edge pixels from the resulting edge detection image, to begin finding geometric primitives [29]. Meaning that PPTH performance may vary between trials. The developed artifact does not account for randomization.

Another example of a dependability threat could be seen with cone detection accuracy. For instance, because of the current implementation for a cone to be accurately detected from 3 meters, the cone height can only have two possible values. This small margin may provide inconsistent results.

2) Internal Validity: Internal validity refers to factors influencing causal relations. Specifically, ensuring that factors outside the conducted research do not influence the investigated goal [32]. Study inclusion/exclusion bias refers to problems that can arise when looking for supporting studies [33]. At the start of the project, the authors inclusion/exclusion criteria were vague and generic, mainly just "image capturing/processing". This may resulted in a skewed foundation of knowledge and literature review that could have then negatively affected the research outcome.

Similar to the previous validity threat, construction of search string refers to problems that are caused by a poorly formed research paper search string, criteria [33]. This may have caused the author to find some irrelevant studies or miss some relevant ones. The impact of this validity threat is also a lower quality foundation of knowledge.

Selection of digital libraries is threat to validity, which refers to problems caused search engines when looking for related work [33]. The different search engines can be either too broad, or too specific. As the previous internal validity threats, the impact of this threat is skewing your foundation of knowledge. In the context of this study, the author mainly used Google Scholar, University of Gothenburg Library[2] and ScienceDirect[3]. Google Scholar provides a wide range of sources, from peer reviewed papers to patents. By using Google Scholar the author may have used low quality studies in their research. On the other hand, ScienceDirect seems to be a specific and narrow search engine, which may have complicated finding related work.

3) External Validity: Generalizability, or external validity, refers to the extent that a particular situation can be applied

[2]University of Gothenburg Library ub.gu.se/sv
[3]ScienceDirect https://www.sciencedirect.com/

to the other settings [34]. In this context, the artifact was evaluated in a controlled environment. Therefore its performance in a different environment, such as, a city or a race track is unknown. Similarly, the artifact has not been evaluated under different weather conditions, therefore the artifact's performance in rainy, foggy etc weather cannot be deduced. High performance under different weather conditions or in a different environment are a necessity for vehicles to ensure driver safety. Thus, evaluating the artifact in a controlled environment severely limits its generalizability.

Another external validity threat is the artifact's dependence on light. PPHT and Canny edge detection are highly dependent on image preprocessing. Without sufficient lighting, the already low cone detection accuracy and rate will decrease further. Meaning the artifact cannot be effectively used in the dark.

For cone detection, the artifact has a maximum possible range of 4 meters. Meaning, that in a real-world scenario the artifact is only suited for short distance cone detection. Canny edge detection and PPHT would still be performed on strong edges in images, making it possible to detect the general area of cones that are further away. However, the artifact is capturing frames using raspiraw at 180FPS with a 640x240 image resolution. The captured images were low quality, making long distance cone detection more difficult, due to a lack of salient edges in low quality images.

VI. Conclusion

Fast traveling vehicles required to make fast and accurate decisions, which is usually solved with expensive systems. The aim of this study was to use design science, dynamic analysis and controlled experiments to provide an affordable, high-speed solution for cone detection. The project used a Pi NoIR V2 camera module for Raspberry Pi 3. Raspberry Pis are cheap and affordable, meaning that individuals wanting to experiment with high-speed cameras are not required to possess expensive equipment. The reduced expensiveness means that the solution is applicable to a broader audience, for instance academics, students, and hobbyists. The results showed that the project aim for high-speed image capture and processing was not reached. However, low-cost appliances can be used to achieve real-time cone detection at 30-35FPS using Canny edge detection, Progressive Probabilistic Hough Transform and open-source software. However, the solution has high amount of false negatives, poor cone detection rate and varying cone detection accuracy, 9-69%. For future work the existing distance calculation formula could be replaced with the Pinhole camera model to improve the artifact's accuracy. To reduce computation the solution could incorporate frame subtraction, to create regions of interest and detect objects in motion more robustly. Another possibility is exchanging Canny edge detection for either Sobel, Prewitt or Robert's operators, which are computationally less expensive than Canny edge detection.

References

[1] A. Conci and A. Sanchez, "Scientific image processing," *Computing in Science & Engineering*, vol. 13, no. 3, pp. 6–8, 2011.

[2] C. Lipski, B. Scholz, K. Berger, C. Linz, T. Stich, and M. Magnor, "A fast and robust approach to lane marking detection and lane tracking," in *2008 IEEE Southwest Symposium on Image Analysis and Interpretation*. IEEE, 2008, pp. 57–60.

[3] S. Sivaraman and M. M. Trivedi, "Integrated lane and vehicle detection, localization, and tracking: A synergistic approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 906–917, 2013.

[4] Y.-J. Wu, F.-L. Lian, C.-P. Huang, and T.-H. Chang, "Image processing techniques for lane-related information extraction and multi-vehicle detection in intelligent highway vehicles," *Int. J. Automotive Technology*, vol. 8, no. 4, pp. 513–520, 2007.

[5] W. Liu, H. Zhang, B. Duan, H. Yuan, and H. Zhao, "Vision-based real-time lane marking detection and tracking," in *2008 11th International IEEE Conference on Intelligent Transportation Systems*, Oct 2008, pp. 49–54.

[6] M. Anandhalli and V. P. Baligar, "A novel approach in real-time vehicle detection and tracking using raspberry pi," *Alexandria engineering journal*, vol. 57, no. 3, pp. 1597–1607, 2018.

[7] A. Gupta and A. Choudhary, "A framework for camera-based real-time lane and road surface marking detection and recognition," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 4, pp. 476–485, 2018.

[8] C. Theobalt, I. Albrecht, J. Haber, M. Magnor, and H.-P. Seidel, "Pitching a baseball: tracking high-speed motion with multi-exposure images," in *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 540–547.

[9] B. Wilburn, N. Joshi, V. Vaish, M. Levoy, and M. Horowitz, "High-speed videography using a dense camera array," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, June 2004, pp. II–II.

[10] S. MacNeill and J.-J. Chanaron, "Trends and drivers of change in the european automotive industry:(i) mapping the current situation," *International journal of automotive technology and management*, vol. 5, no. 1, pp. 83–106, 2005.

[11] F. Kuhnert, C. Stürmer, and A. Koster, "Five trends transforming the automotive industry," *PricewaterhouseCoopers GmbH Wirtschaftspriüfungsgesellschaft: Berlin, Germany*, 2018.

[12] G. Senthilkumar, K. Gopalakrishnan, and V. S. Kumar, "Embedded image capturing system using raspberry pi system," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 3, no. 2, pp. 213–215, 2014.

[13] H.-Q. Nguyen, T. T. K. Loan, B. D. Mao, and E.-N. Huh, "Low cost real-time system monitoring using raspberry pi," in *2015 Seventh International Conference on Ubiquitous and Future Networks*. IEEE, 2015, pp. 857–859.

[14] M. Kochláň, M. Hodoň, L. Čechovič, J. Kapitulík, and M. Jurečka, "Wsn for traffic monitoring using raspberry pi board," in *2014 Federated Conference on Computer Science and Information Systems*. IEEE, 2014, pp. 1023–1026.

[15] L. Ujjainiya and M. K. Chakravarthi, "Raspberry pi based cost effective vehicle collision avoidance system using image processing," *ARPN J. Eng. Appl. Sci*, vol. 10, no. 7, 2015.

[16] J. Choi, S.-W. Han, S.-J. Kim, S.-I. Chang, and E. Yoon, "A spatial-temporal multiresolution cmos image sensor with adaptive frame rates for tracking the moving objects in region-of-interest and suppressing motion blur," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2978–2989, 2007.

[17] D. Litwiller, "Ccd vs. cmos: Facts and fiction," *Photonics spectra*, vol. 35, no. 1, pp. 154–158, 2001.

[18] B. Zhao, X. Zhang, S. Chen, K. Low, and H. Zhuang, "A 64×64 cmos image sensor with on-chip moving object detection and localization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 4, pp. 581–588, April 2012.

[19] A. Teman, S. Fisher, L. Sudakov, A. Fish, and O. Yadid-Pecht, "Autonomous cmos image sensor for real time target detection and tracking," in *2008 IEEE International Symposium on Circuits and Systems*. IEEE, 2008, pp. 2138–2141.

[20] J. Iivari and J. R. Venable, "Action research and design science research-seemingly similar but decisively dissimilar," 2009.

[21] M. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, "Action design research," *Management Information Systems Quarterly*, vol. 35, no. 1, pp. 37–56, 2011.
[22] R. H. Von Alan, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
[23] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.
[24] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.
[25] Picamera documentation. Accessed: 2019-04-07. [Online]. Available: https://picamera.readthedocs.io/en/release-1.12/fov.html
[26] Hermann-SW and 6by9, "Raspiraw," https://github.com/Hermann-SW/fork-raspiraw, 2019.
[27] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International journal of image processing (IJIP)*, vol. 3, no. 1, 2008.
[28] G. Shrivakshan and C. Chandrasekar, "A comparison of various edge detection techniques used in image processing," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 269, 2012.
[29] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119–137, 2000.
[30] techyian, "Mmal," https://github.com/techyian/MMALSharp/wiki/What-is-MMAL%3F, 2019.
[31] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research-an initial survey." in *Seke*, 2010, pp. 374–379.
[32] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.
[33] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Ver-beek, and A. Chatzigeorgiou, "Identifying, categorizing and mitigating threats to validity in software engineering secondary studies," *Information and Software Technology*, 2018.
[34] J. Maxwell, "Understanding and validity in qualitative research," *Harvard Educational Review*, vol. 62, no. 3, pp. 279–301, 1992.

APPENDIX A
DESIGN CYCLE 1 RESULTS

TABLE IV: Design Cycle 1 Performance Measurements

Trial Nr.	Edge Detection (s)	PPHT (s)	Total (s)
1	0.010	0.073	0.083
2	0.006	0.099	0.105
3	0.020	0.151	0.171
4	0.005	0.048	0.053
5	0.006	0.032	0.038
6	0.004	0.169	0.173
7	0.013	0.030	0.043
8	0.006	0.088	0.093
9	0.015	0.101	0.116
10	0.013	0.044	0.056
11	0.005	0.137	0.142
12	0.007	0.151	0.158
13	0.014	0.040	0.054
14	0.007	0.127	0.134
15	0.005	0.067	0.072
16	0.011	0.087	0.098
17	0.016	0.105	0.121
18	0.008	0.064	0.072
19	0.005	0.071	0.076
20	0.006	0.156	0.162
21	0.005	0.063	0.069
22	0.010	0.093	0.102
23	0.013	0.170	0.183
24	0.005	0.047	0.052
25	0.004	0.048	0.052
26	0.004	0.196	0.200
27	0.012	0.089	0.101
28	0.016	0.111	0.127
29	0.007	0.067	0.074
30	0.006	0.145	0.151
31	0.005	0.059	0.064
32	0.010	0.117	0.128
33	0.014	0.108	0.122
34	0.014	0.082	0.096
35	0.013	0.067	0.080
36	0.011	0.174	0.185
37	0.009	0.034	0.043
38	0.006	0.098	0.104
39	0.010	0.118	0.128
40	0.009	0.083	0.092
41	0.005	0.130	0.135
42	0.013	0.051	0.064
43	0.010	0.084	0.094
44	0.006	0.097	0.104
45	0.010	0.079	0.089
46	0.005	0.127	0.133
47	0.010	0.032	0.042
48	0.018	0.132	0.149
49	0.009	0.120	0.129
50	0.005	0.040	0.045
51	0.006	0.092	0.098
52	0.011	0.084	0.094
53	0.014	0.117	0.130
54	0.011	0.043	0.054
55	0.016	0.178	0.194
56	0.014	0.152	0.166
57	0.017	0.071	0.087
58	0.010	0.060	0.069
59	0.010	0.123	0.134
60	0.010	0.197	0.207
61	0.013	0.080	0.093
62	0.011	0.045	0.056
63	0.007	0.036	0.043
64	0.011	0.101	0.112
65	0.011	0.132	0.142
66	0.006	0.095	0.102
67	0.011	0.097	0.108
68	0.011	0.111	0.121
69	0.013	0.080	0.093
70	0.005	0.038	0.043
71	0.014	0.147	0.161
72	0.006	0.118	0.124
73	0.006	0.092	0.098
74	0.007	0.030	0.037
75	0.016	0.191	0.206
76	0.010	0.086	0.097
77	0.010	0.044	0.054
78	0.007	0.142	0.149
79	0.010	0.026	0.037
80	0.010	0.057	0.067
81	0.012	0.104	0.116
82	0.010	0.120	0.130
83	0.010	0.117	0.127
84	0.006	0.100	0.106
85	0.013	0.053	0.065
86	0.016	0.081	0.096
87	0.005	0.147	0.152
88	0.005	0.113	0.118
89	0.008	0.049	0.057
90	0.009	0.111	0.120
91	0.009	0.069	0.079
92	0.006	0.112	0.118
93	0.012	0.118	0.130

94	0.010	0.064	0.073
95	0.011	0.038	0.049
96	0.007	0.185	0.192
97	0.017	0.204	0.221
98	0.008	0.044	0.052
99	0.009	0.038	0.047
100	0.005	0.142	0.148

APPENDIX B
DESIGN CYCLE 2 RESULTS

TABLE V: 1 Meter, Trial 1

Detected Distance (mm)	Cone Height (px)
970.769	177
876.664	196
954.589	180
1041.370	165
938.940	183
1022.774	168
1004.831	171
1004.831	171
1004.831	171
918.856	187
987.506	174
1016.722	169
998.989	172
913.969	188
987.506	174
981.863	175
944.099	182
933.837	184
987.506	174
1010.742	170
904.348	190
1022.774	168

TABLE VI: 1 Meter, Trial 2

Detected Distance (mm)	Cone Height (px)
949.315	181
954.589	180
933.837	184
959.922	179
954.589	180
1028.899	167
998.989	172
1041.370	165
1028.899	167
993.214	173
959.922	179
899.613	191
913.969	188
913.969	188
1035.097	166
913.969	188
944.099	182
993.214	173
1010.742	170
998.989	172
998.989	172
1028.899	167
959.922	179
944.099	182
933.837	184
938.940	183
987.506	174
959.922	179
928.790	185

TABLE VII: 1 Meter, Trial 3

Detected Distance (mm)	Cone Height (px)
1010.742	170
981.863	175
944.099	182
965.315	178
954.589	180
1041.370	165
1028.899	167
949.315	181
918.856	187
928.790	185
987.506	174
890.291	193
1004.831	171
1010.742	170
1004.831	171
965.315	178
981.863	175
949.315	181
987.506	174
938.940	183
928.790	185
959.922	179
928.790	185
1004.831	171
976.285	176
918.856	187
928.790	185

TABLE VIII: 1 Meter, Trial 4

Detected Distance (mm)	Cone Height (px)
959.922	179
970.769	177
1041.370	165
938.940	183

954.589	180
928.790	185
904.348	190
872.214	197
1004.831	171
965.315	178
1028.899	167
899.613	191
981.863	175
970.769	177
876.664	196
976.285	176
918.856	187
987.506	174
1060.655	162
998.989	172
1028.899	167
1010.742	170
1004.831	171
976.285	176
976.285	176
1010.742	170

TABLE IX: 1 Meter, Trial 5

Detected Distance (mm)	Cone Height (px)
1035.097	166
976.285	176
1041.370	165
933.837	184
981.863	175
993.214	173
904.348	190
954.589	180
959.922	179
899.613	191
976.285	176
1022.774	168
976.285	176
976.285	176
987.506	174
1035.097	166
1010.742	170
899.613	191
923.796	186
938.940	183
938.940	183
933.837	184
987.506	174
949.315	181
1035.097	166
1010.742	170
909.133	189

TABLE X: 2 Meters, Trial 1

Detected Distance (mm)	Cone Height (px)
2070.194	83
2231.508	77
2095.440	82
1909.179	90
1789.855	96
2095.440	82
2121.310	81
2095.440	82
2231.508	77
1952.569	88
2021.483	85
2454.658	70
2121.310	81
2021.483	85
1997.978	86
2121.310	81
2147.826	80
1997.978	86
2021.483	85
2045.549	84

TABLE XI: 2 Meters, Trial 2

Detected Distance (mm)	Cone Height (px)
2021.483	85
2045.549	84
1909.179	90
1909.179	90
1909.179	90
2202.898	78
2321.974	74
2070.194	83
2321.974	74
2070.194	83
2095.440	82
2147.826	80
2202.898	78
1975.012	87
1997.978	86
2564.568	67
2202.898	78
1997.978	86

TABLE XII: 2 Meters, Trial 3

Detected Distance (mm)	Cone Height (px)
2771.388	62
1975.012	87
1952.569	88

1888.199	91
2386.473	72
2291.014	75
1909.179	90
2231.508	77
2231.508	77
2147.826	80
2231.508	77
2231.508	77
2147.826	80
2175.014	79
2021.483	85
2147.826	80

TABLE XIII: 2 Meters, Trial 4

Detected Distance (mm)	Cone Height (px)
2202.898	78
2147.826	80
2321.974	74
1975.012	87
2121.310	81
1808.696	95
2147.826	80
1997.978	86
2260.870	76
2603.426	66
1930.630	89
2175.014	79
2095.440	82
2021.483	85
2490.233	69
2771.388	62
2095.440	82
1952.569	88
2231.508	77

TABLE XIV: 2 Meters, Trial 5

Detected Distance (mm)	Cone Height (px)
2045.549	84
1997.978	86
2021.483	85
2095.440	82
2147.826	80
2202.898	78
2454.658	70
2231.508	77
1867.675	92
2231.508	77
2454.658	70
2231.508	77
1997.978	86
1930.630	89
2231.508	77
2070.194	83
2603.426	66
2490.233	69

TABLE XV: 3 Meters, Trial 1

Detected Distance (mm)	Cone Height (px)
2816.821	61
3068.323	56
2816.821	61
2863.768	60
2816.821	61
2863.768	60
3181.965	54
3014.493	57
3242.002	53
2684.783	64
3181.965	54
2771.388	62
2816.821	61
3124.111	55
2771.388	62
2727.398	63
2603.426	66
2816.821	61
2962.519	58

TABLE XVI: 3 Meters, Trial 2

Detected Distance (mm)	Cone Height (px)
2912.307	59
3124.111	55
3242.002	53
3014.493	57
3242.002	53
3242.002	53
3242.002	53
3242.002	53
3181.965	54
2727.398	63
3242.002	53
2684.783	64
2962.519	58
2771.388	62
2771.388	62
3436.522	50
3436.522	50
3242.002	53
2771.388	62
2962.519	58
2863.768	60

TABLE XVII: 3 Meters, Trial 3

Detected Distance (mm)	Cone Height (px)
3181.965	54
2816.821	61
2863.768	60
3242.002	53
2863.768	60
2863.768	60
2816.821	61
2727.398	63
2962.519	58
2863.768	60
2771.388	62
3369.139	51
2727.398	63
3068.323	56
3735.350	46
3181.965	54
2863.768	60
2912.307	59
2816.821	61
2962.519	58
2684.783	64
2684.783	64
2912.307	59
2863.768	60
3181.965	54

TABLE XVIII: 3 Meters, Trial 4

Detected Distance (mm)	Cone Height (px)
3181.965	54
2816.821	61
3369.139	51
2962.519	58
3242.002	53
2912.307	59
3181.965	54
2564.568	67
2771.388	62
2816.821	61
3242.002	53
2962.519	58
3242.002	53
3124.111	55
2863.768	60
2962.519	58
3181.965	54
2863.768	60
9545.894	18
2771.388	62
3181.965	54
2863.768	60
3181.965	54

TABLE XIX: 3 Meters, Trial 5

Detected Distance (mm)	Cone Height (px)
2727.398	63
2727.398	63
2962.519	58
2643.478	65
3242.002	53
2771.388	62
2771.388	62
2912.307	59
3369.139	51
2771.388	62
2727.398	63
2816.821	61
3369.139	51
2771.388	62
3242.002	53
2912.307	59
2863.768	60
3014.493	57
3506.655	49
3242.002	53
2863.768	60