Final Year Project

Computer Vision for a Real-Time Physical Traffic Testbed

Aaron Hurst

30th October, 2017

Word count: 7,970

School of Electrical, Electronic and Computer Engineering
The University of Western Australia
Australia

Dr Tim French
School of Computer Science and Software Engineering
The University of Western Australia

Dr Ghulam Mubashar Hassan
School of Computer Science and Software Engineering
The University of Western Australia

Contents

List of Figures

List of Tables

Abstract

Road transportation will soon experience a revolution with the introduction of autonomous vehicles. Many factors are unclear in this revolution and demand extensive research. Public and regulatory acceptance of autonomous vehicles will also be critical to a successful transition to this technology.

This research contributes to a larger project to develop a versatile, low-cost testbed for traffic and autonomous vehicle research. Its tangible and engaging nature also lends this testbed to use as a tool for public engagement.

The focus of this research is the computer vision system required to track multiple small vehicles in the testbed in real time. Multiple computer vision algorithms have been tested to determine their efficacy for this application. Several criteria, in particular processing time, accuracy and robustness to occlusion and lighting variation, have been used to evaluate the performance of these algorithms. The constraint of running the computer vision from a Raspberry Pi 3 was also imposed.

Overall, colour histogram comparison using χ^2distance has emerged as the most effective approach, with a hue matching approach also viable. More complex and cutting-edge tracking algorithms were found to be either too slow for real-time use or unable to track vehicles moving at reasonable speeds.

Chapter 1

Introduction

Autonomous vehicles are widely predicted to establish a major position in the auto-mobile market within the next few decades [1, 2, 3]. This will incite a vast array of changes to driving patterns, vehicle usage and government policy [4]. Direct benefits stemming from autonomous vehicles are expected to be in the order of hundreds of billions to trillions of dollars annually in the United States [3]. Positive impacts include a massive reduction in road accidents, higher fuel-efficiency and innumerable hours of driving time made available for more productive activities [3]. However, negative impacts are also foreseen, including a potential dip in safety during the transition to autonomous vehicles [5] and greater congestion due to increased travel demand [4].

In addition to continued research, regulatory and public acceptance of autonomous vehicles will be key to realising their benefits. Tangible representations such as physical testbeds can both facilitate research in this field and provide a tool for stimulating public engagement with research and autonomous vehicles in general. This project contributes the computer vision required for such a testbed that has been developed in collaboration with two other students.

The task for the computer vision system is to track small remote-controlled cars within the testbed area. This provides a feedback loop to the software that controls the cars. The primary goal of this research is to investigate multiple computer vision algorithms and determine the most appropriate method for this testbed. Several prioritised design requirements and validation criteria have been established for this purpose. These requirements focus on processing speed, tracking accuracy and robustness. A key constraint is that the computer vision must run on a Raspberry Pi 3.

Four different categories of algorithms have been investigated in this research: colour histogram comparison, hue matching, single-object trackers and saliency detection. Histogram comparison and hue matching both met all of the design requirements and provide effective computer vision for the testbed. Overall, histogram comparison was deemed the most appropriate due to its superior performance against multiple criteria. In contrast, all single-object trackers and saliency detection algorithms investigated were all unable to meet at least one mandatory requirement.

The remainder of this paper is as follows. Chapter 2 provides a literature review of autonomous vehicles, traffic models and object tracking and detection in computer vision. Chapter 3 presents the approach taken in evaluating computer vision algorithms. Chapter 4 describes the algorithms

investigated in this research. Chapter 5 details the experimental results used to evaluate these algorithms. Chapter 6 provides a discussion of the results in context of the system requirements. Chapter 7 concludes the paper and provides suggestions for future work.

Chapter 2

Literature Review

This literature review provides a survey of autonomous vehicles (Section 2.1), traffic simulation and modelling with a focus on hardware testbeds (Section 2.2) and object detection and tracking methods relevant to this project (Section 2.3).

2.1 Autonomous Vehicles

Over the past 10-15 years, autonomous vehicle (AV) technology has advanced dramatically to the point where widespread adoption is now a question of 'when', not 'if' [1, 6, 3]. Indeed, AVs are predicted to account for "around 50% of vehicle sales, 30% of vehicles, and 40% of all vehicle travel by 2040" [1, p. 285]. Motivation for this enormous change is strong amid predictions of economic benefits in the United States of between US$0.2 and US$1.9 trillion annually by 2025 [3]. The potential for traffic accidents is also expected to fall substantially, given human error contributes to approximately 93% of crashes in the United States [7].

Many auto-makers and non-auto-makers alike are investing heavily in AVs [8, 9]. Notable among the non-auto-makers is Google, whose company Waymo has completed over three million miles of testing [10, 11]. A key initiative that stimulated the early development of AVs was the Defense Advanced Research Projects Agency (DARPA) challenges. The first DARPA 'Grand Challenge', in 2004, saw 17 teams attempt a 225 km course through challenging and unfamiliar terrain [12]. All competitors failed in less than 12 km [13]. One year later, teams attempted a similar course with five successfully completing it and only one failing to complete more than 12 km [13]. A subsequent 'Urban Challenge' in 2007 saw six out of eleven teams successfully complete an on-road course requiring obedience to traffic rules and interaction with other vehicles [14, 15]. Overall, DARPA's initiatives inspired substantial advancements in autonomous vehicle technology.

Some, however, are less optimistic about the benefits of autonomous vehicles. For example, it has been argued that due to behavioural changes and interaction between autonomous and human-driven vehicles, autonomous vehicles may not improve overall safety [5]. The cost of additional technology required for autonomous driving may inflate car prices more than the cost savings to consumers, reducing demand [4, 16]. Ethical and legal challenges facing autonomous vehicles also

loom as barriers to adoption [4, 17]. Tying all these issues together is the issue of public acceptance. Indeed, Manyika et al. state that the benefits of autonomous vehicles will only materialise if "regulators approve autonomous driving and the public accepts the concept" [3, p. 78]. Other authors support this conclusion [1, 2, 16, 18].

2.2 Traffic Simulation & Modelling

As discussed above, the impacts of AVs are still debated and uncertain. This uncertainty extends to multiple facets of traffic behaviour [19]. Key topics in this space include how AVs will impact road safety and traffic congestion [5, 4], as well as their routing behaviours within the road network [1].

These uncertainties require research. Conventionally, traffic research is done using simulation software packages [19, 20, 21]. Widely used examples include MATsim [22], Aimsun and VISSIM [23]. However, hardware testbeds are also used to good effect. One of the more developed examples was created by researchers at the University of California, Los Angeles [24, 25, 26, 27]. The original architecture of this system consisted of a driving area, multiple small vehicles, cameras and separate computers for tracking the vehicles and controlling their behaviour (see Figure 2.1) [24]. Key benefits of this system are its low cost and small footprint (1.5x2.0 m) [27]. However, it also requires multiple desktop computers or custom-built cars with microprocessors and multiple sensors [24, 27]. Cameras are also mounted 2.6 m above the testbed, which limits its usability [24].

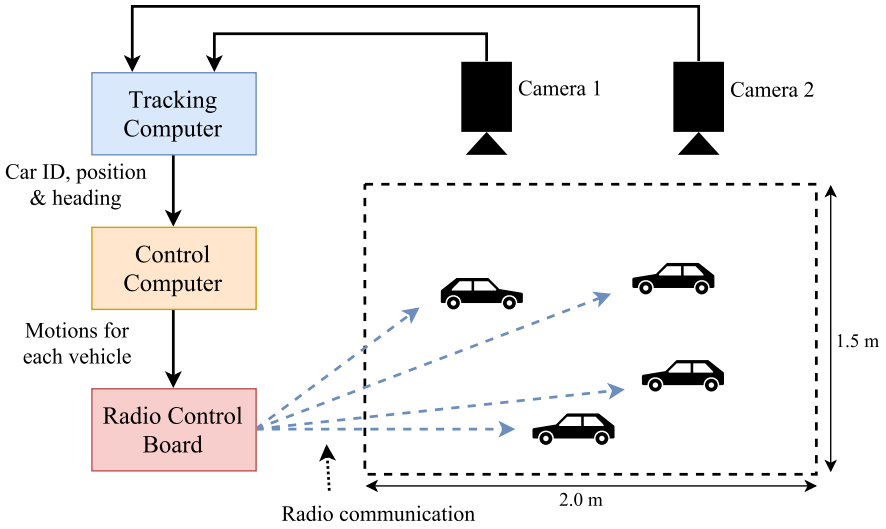


Figure 2.1: Original UCLA vehicle testbed architecture showing the driving area, two cameras for tracking, the tracking and control computers (both 3.0 GHz PCs) and radio control board [24].

Other researchers have used much larger robotic vehicles for similar projects, which require room-sized spaces [28, 29]. One recent project used small Zen-Wheels micro-cars (58x30 mm) in a similarly sized testbed [30]. However, this project was not completed and experienced significant challenges, in particular with the computer vision software.

Overall, in addition to allowing for hardware prototyping, the key benefit of hardware testbeds is the ability to assessing the impact of 'real-world' challenges such as sensor noise and communication issues on traffic protocols and algorithms [27, 31].

2.3 Object Detection & Tracking

One of the most common methods used for tracking vehicles in testbed-like environments is to apply distinctive markers to each vehicle and write software for detecting these markers in each video frame [25, 29, 30, 31, 32]. This effectively uses fast object *detection* for the purpose of *tracking*. For example, most teams in early RoboCup soccer competitions used colour markers to distinguish and orient their team's robots [31].

A particularly successful implementation of this approach was able to track ten robots at up 80 frames per second using a 656x494 pixel camera and 3 GHz processor [32]. The authors also demonstrated their algorithm to be robust against lighting intensity changes, but less so against light spectrum changes. Notably, identical markers were used for all robots, meaning that individual robots could not be identified directly from their marking. Using unique markers could potentially improve this model, but would limit the maximum number of robots [32].

The previously mentioned project by Suter [30] used black-and-white markers resembling QR-codes. However, despite the use of a high-resolution camera, the markers were difficult to resolve due to their small size and intricate design. Additionally, little consideration was given to the constraints of real-time operation, which is critical for a physical testbed.

For detection of colour markers, two relevant methods are hue matching and histogram comparison. Hue matching is the simpler of the two and involves scanning the frame for pixels whose hue value matches a specified range. Various authors have experimented with this approach, finding it to be generally effective, but vulnerable to detecting background objects and lighting variation [32, 33].

Histogram comparison was first demonstrated in 1991 by Swain and Ballard and is still well regarded in recent publications [35]. Reasons for its continued use include its computational efficiency and robustness to occlusion, illumination and object rotation [34]. The methodology is to calculate the colour histogram over some region and compare this to a library of known histograms [34]. Common metrics for histogram comparison include intersection [34] and chi-squared distance [36].

Visual saliency (to a human observer) is another method used for object detection [37]. Methods based on predicting what object is most salient to an observer or where an observer will look in an image are known as salient object detection and fixation prediction methods, respectively [38]. Typically, saliency-based models are only applied to static images [39], though some models have been applied to multiple image situations [40], video [41] and object tracking [42]. However, this is considered a "non-trivial" task [37, p. 20].

More general single-object tracking algorithms are abundant in the literature [43, 44, 45, 46]. Many deal with challenging problems including occlusion, arbitrary objects, background clutter, scale variation, viewpoint change and object deformation [43]. Some have also been developed for specific applications such as unmanned aerial vehicle-based tracking [47, 48]. The current trend in this field of research is a move toward deep learning approaches, which are hailed as improving robustness [43]. However, this generally comes at the cost of decreased processing speed [43].

Chapter 3

Design Process

This chapter discusses the high-level goals for the hardware traffic testbed (Section 3.1), the testbed architecture (Section 3.2), requirements for the computer vision system (Section 3.3) and the evaluation framework used for the computer vision (Section 3.4).

3.1 Project Goals

The testbed that this research contributes to has been designed and built in collaboration with two other students – Ray Barker and Ridge Shrubsall. Each student was responsible for different components of the overall system, as described in Section 3.2. The author was responsible for the testbed computer vision. The high-level goals for this testbed as a whole focus on its utility for demonstrating and testing multi-vehicle traffic scenarios and algorithms. Goals specific to the computer vision for this testbed include:

1. Ability to support up to eight cars simultaneously,
2. Capable of operating reliably indoors subject to uncertain lighting conditions, and
3. Capable of supporting cars moving at up to 0.3 m/s.

Additional goals relevant to other components of the testbed system include:

1. Ability to simulate different driving environments,
2. Low system cost,
3. Easily reproducible, and
4. Allow for convenient transportation of the testbed.

3.2 System Architecture

A block diagram illustrating the sub-systems and interfaces in this testbed is shown in Figure 3.1. This architecture forms a complete control loop. Beginning with the camera (Raspberry Pi Camera V2), images of the driving surface (1.2x0.9 m) are captured and sent to the computer vision system,

which runs on a Raspberry Pi 3. The computer vision system processes each image and determines the kinematic state of each car – its position, velocity and orientation. This information is passed to the car controller program, which in turn passes a filtered environment description to agent models for the cars. The agents then determine appropriate actions for each car, which are returned to the controller program. Finally, the controller translates these actions into driving commands and transmits these to the cars via bluetooth. Simultaneously, the controller also sends the driving environment as an image to the projector, which displays this over the driving surface for visual representation of the environment.

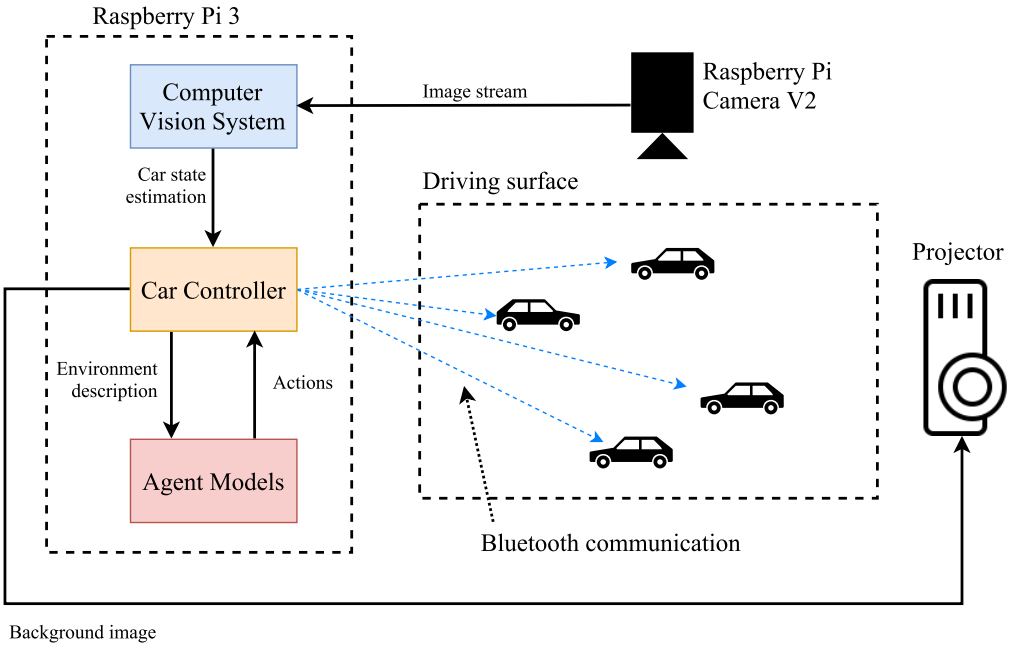


Figure 3.1: Testbed system architecture for the testbed showing key sub-systems and interfaces.

As mentioned, each student working on this project was responsible for specific components of the testbed. Ray Barker designed and constructed the physical component of the testbed and developed the car controller software. Ridge Shrubsall focused on the bluetooth communication between the controller and the cars. The author designed and implemented the computer vision system. A photograph of the testbed as currently constructed in shown in Figure 3.2.

The computer vision system is necessary since the cars do not respond perfectly to instructions they receive. As such, a feedback mechanism is required [31, 30]. Global vision was selected for this role since the cars themselves – "ZenWheels micro cars" – are unable to provide feedback [49] and attaching adequate sensing and communication equipment to each car was considered a greater engineering challenge.

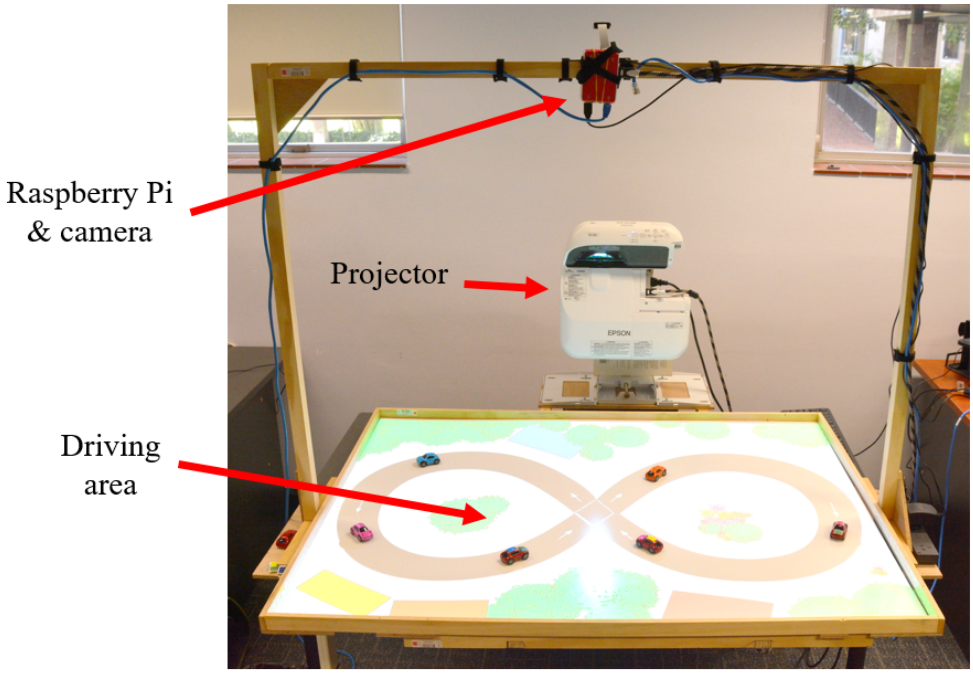


Figure 3.2: Testbed as constructed.

3.3 Computer Vision Requirements

The design requirements for the computer vision system are listed in Table 3.1 along with their background and priority. Most requirements are mandatory for the testbed to function effectively, however some are available for trade-off. Accurate detection (R2) is mandatory, however false positives are considered more critical than false negatives due to their effect on the controller. As such, it is acceptable to trade a reduced false positive rate for an increased false negative rate.

Tolerance of different lighting conditions (R6), high car speed (R7) and background independence (R8) are non-mandatory. However, improving tolerance of different lighting conditions is considered most valuable among these three requirements. This is because tolerance of different lighting conditions will allow the testbed to be transported and used more widely and thereby increase its versatility. High car speed is prioritised next since this will allow for more visually interesting and realistic demonstrations. Background independence is prioritised last since a simple background can suffice for most uses.

The system is also subject to the constraint that it must run on a Raspberry Pi 3. However, the rigid and uniform shape of the cars and fixed camera field of view simplify the complexity of the task.

Table 3.1: Computer vision system requirements

ID	Description	Background	Priority
R1	Real-time operation	Image frames must be processed sufficiently fast for the controller to provide fluid and realistic control of the cars. This must be achieved independent of the number of cars being tracked.	Mandatory
R2	Accurate detection	All cars must be detected consistently in each frame with few false positives (misclassification) or false negatives (failed detection). A low false positive rate is considered more important since the controller will send incorrect instructions in this case. False negatives will result in no instruction and are therefore less disruptive.	Mandatory
R3	Accurate state estimation	Car state estimations – position, velocity and orientation – must be sufficiently accurate for the controller to provide relevant instructions to the cars.	Mandatory
R4	Occlusion tolerant	It is a common event that cars become occluded, possibly by removing them from the testbed. The computer vision must be able to cope with this situation and recover from it by resuming tracking when the car returns to view.	Mandatory
R5	Car orientation independent	Car orientation is uncertain at run-time.	Mandatory
R6	Tolerant of lighting variation	This testbed is intended to be transportable and will therefore encounter a variety of lighting conditions.	1
R7	High car speed	Higher vehicle speeds allow for more realistic and engaging demonstrations.	2
R8	Background independent	The projector will be used to display different traffic scenarios on the driving surface.	3

3.4 Evaluation Framework

To facilitate evaluation of computer vision algorithms, the requirements listed in Table 3.1 have been assigned clear validation criteria, which are detailed in Table 3.2.

Requirement R2 allows for a false negative (failed detection) rate of up to 5%, but a false positive (misclassification) rate of only 0.5%. This reflects the importance placed on avoiding disruptive misclassification events.

Requirement R3 requires the position estimates for the cars to be accurate to within 2 mm. This measurement is comparable to what other authors have achieved [32, 24] and represents approximately ±3.4% of the car's length.

Requirements R6 and R8 are both questions of degree. That is, extreme changes in lighting conditions or projected driving environment will certainly affect the computer vision performance. As such, 'reasonable' will be interpreted as within the scope of normal testing/demonstration activities.

Table 3.2: Computer vision system validation criteria

ID	Description	Validation Criteria
R1	Real-time operation	The computer vision algorithm must process at least 5 frames per second while tracking 1-8 cars simultaneously. This threshold was experimentally found to be adequate for low-speed control.
R2	Accurate detection	True positive detection rate for each car is ≥95% while the false positive detection rate is <0.5%.
R3	Accurate state estimation	Car position estimate is accurate to within ±2 mm and velocity measurement noise is <5 mm/s.
R4	Occlusion tolerant	Correctly reports failed detection when car is occluded and re-commences tracking within 1 second of the car returning to the camera's field of vision.
R5	Car orientation independent	Performance is unaffected by car orientation.
R6	Tolerant of lighting variation	Performance is unaffected by reasonable changes in external lighting conditions.
R7	High car speed	Successfully tracks cars moving at 0.3 m/s.
R8	Background independent	Performance is unaffected by reasonable changes in the projected driving environment or removing it (no projection).

Chapter 4

Design Outputs

Six different computer vision algorithms have been investigated in this research project: hue matching, histogram comparison, two single-object trackers and two saliency detection algorithms. Each was implemented in C++, with most also making heavy use of OpenCV [50]. This chapter provides a summary of the implementation and experimental development of each algorithm (Sections 4.1 to 4.4) as well as the peripheral components of the computer vision system (Section 4.5).

4.1 Hue Matching

As shown in Figure 4.1, six of the cars used in this testbed have distinct, highly saturated colours. Hue matching aims to exploit this by scanning video frames for regions of pixels that match each car's known colour. A matched *region* of the correct size is then interpreted as a positive detection. Investigation of this technique was motivated by its simplicity and effectiveness for simple applications as demonstrated in the literature [32, 33, 51].

Figure 4.1: Six different coloured cars used in the testbed

The structure of this algorithm is as follows. First, an image is obtained from the camera and converted from Red-Green-Blue colour space to Hue-Saturation-Intensity (HSI). In this colour scheme, hue represents the colour of a pixel, saturation represents the intensity of the colour and intensity represents the pixel brightness. The HSI image is then scanned for pixels whose saturation

and intensity are above certain thresholds and whose hue matches the expected range for a given car. This produces a binary image or *mask* of matching pixels. Note that this process is completed independently for each car.

Once matching is complete, the resulting mask in cropped (to remove the testbed borders) and dilated. Dilation is required since it is common for the hue matching process to produce a binary image with discontinuous regions over the cars. This would make detection based on finding large *continuous* regions difficult. An example illustrating the use of dilation to 'fill in' such discontinuities is shown in Figure 4.2. It was found that applying a single iteration of dilation to the mask using a 3x3 kernel produced optimal results.

(a) No dilation (b) Dilation with 3x3 kernel

Figure 4.2: Illustration of the effect of applying dilation to the binary mask produced through hue matching. In 4.2a there are significant discontinuities over the car footprint. In 4.2b, post dilation, these discontinuities have been eliminated.

Next, all regions of matched pixels are analysed. The largest detected region within certain size limits is interpreted as a positive detection. If no suitable region is found, the algorithm reports failure. If one is found, its centroid is used to estimate the car's position.

4.1.1 Calibration

Calibration of the ranges of hue values considered in the matching process for each car was completed based on numerous images taken under a variety of lighting conditions. Ranges were selected to simultaneously encompass the greatest number of correct matching pixels and the fewest incorrect matching pixels. The chosen ranges for the six cars shown above are depicted in Figure 4.3.

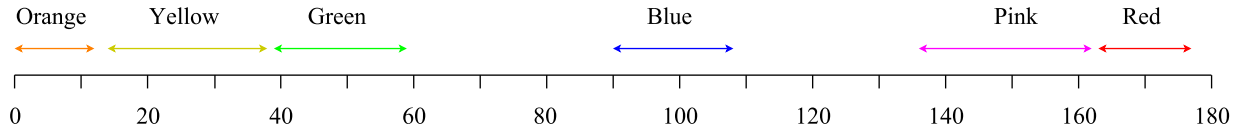


Figure 4.3: Calibrated hue ranges for the six different coloured cars.

4.1.2 Shutter Duration

The effectiveness of this algorithm was also found to be highly dependent on the brightness of the captured image. That is, by appropriately overexposing the image cars could be detected far more accurately. This effect is demonstrated in Figure 4.4 which shows the elimination of background elements (a 'racetrack' in this case) from the mask as shutter duration increased. Hence, an automated shutter duration selection function was developed to determine the minimum shutter duration to achieve background elimination.

Figure 4.5 illustrates the working of this function. As the shutter duration is increased, the number of pixels in the matched binary mask decreases gradually until a 'cut-off' point, after which the background is not detected. The shutter duration is then set just beyond this cut-off.

(a) 4.62 ms (b) 5.94 ms (c) 7.26 ms

Figure 4.4: Effect of shutter duration on binary mask produces in the hue matching process with one car in frame.

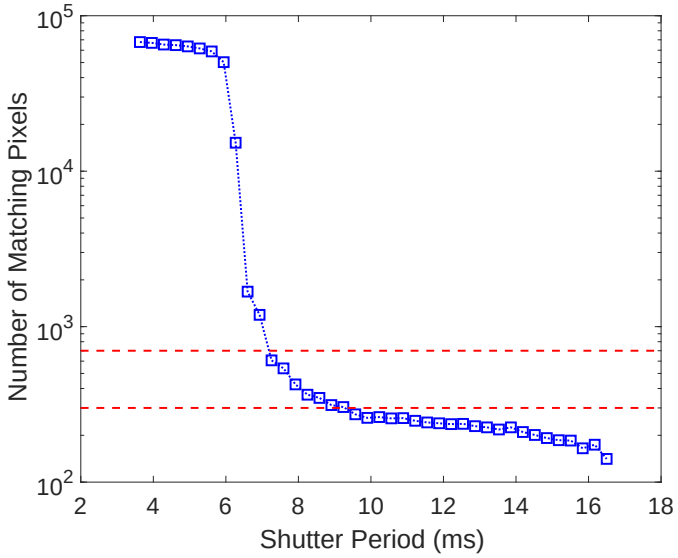
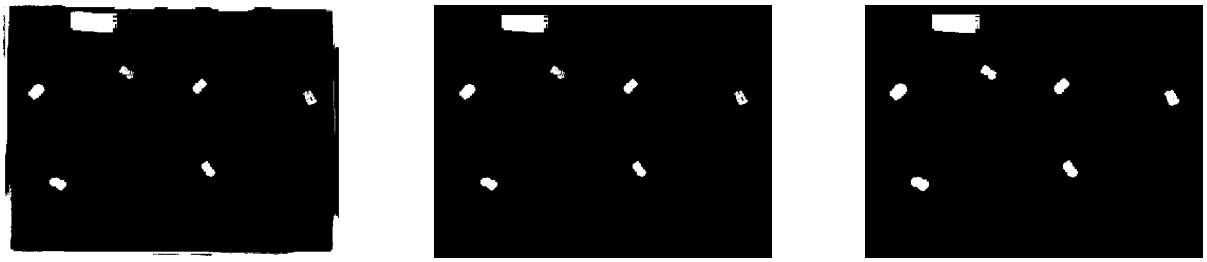


Figure 4.5: Effect of shutter duration on the number of matched pixels in the binary mask. Dashed red lines highlight the target band for one car.

4.2 Histogram Comparison

As discussed in the literature review, detecting objects based on their colour histogram is a well-established and reliable technique. Given a set of different coloured cars, this technique could therefore be expected to yield positive results.

The structure of this algorithm is as follows. First, a 'global binary mask' is generated by matching only those pixels with saturation and intensity values above certain thresholds. This filters out dark or pale regions of the frame and highlights the cars. An example is shown in Figure 4.6a. Cropping and dilation are then applied as per Section 4.1. The effect of these operations is illustrated in Figures 4.6b and 4.6c, respectively.



(a) Original mask (b) Mask after cropping borders (c) Mask after applying dilation

Figure 4.6: Example global mask with six cars. A region of the background (projected image) is also detected; however, its large size means it is ignored.

Next, a histogram of hue values is calculated over each car-sized region. Note that the global mask is used to determine *where* to calculate the histograms; the original image, in HSI colour-space, is used for the actual calculation. In this way, a list of *observed histograms* is assembled. This list is then compared with a library of *prototype histograms* that are each associated with a particular car. For each car, the observed histogram most closely matching its prototype histogram is interpreted as a positive detection, provided the match is at least as good as some minimum threshold.

Histogram comparison is achieved using the χ^2 distance, which uses the following equation [36]:

$$\chi^2 \text{ distance} = \sum_i \frac{(C_i - P_i)^2}{P_i} \tag{4.1}$$

Where C_i is the i^{th} element in an observed histogram and P_i is the i^{th} element in a prototype histogram. Smaller χ^2distances correspond to more closely matching histograms. Histogram intersection [34] was also investigated as a comparison metric; however, it was found to be less accurate.

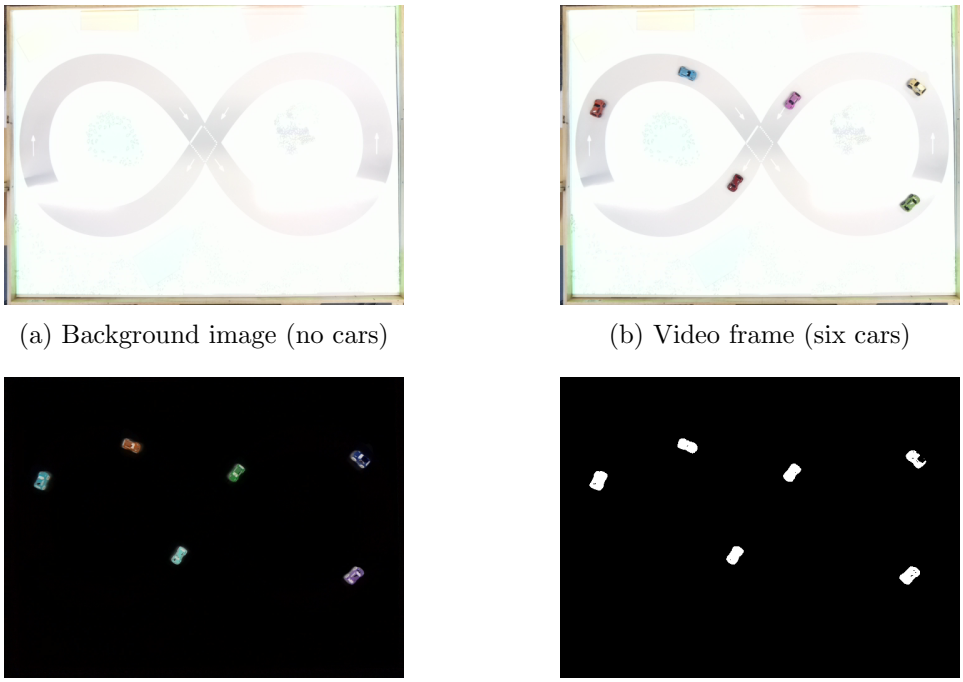
Prototype histograms were obtained by taking several thousand images of each car under different lighting conditions. Hue histograms were then calculated for each image and averaged to produce a single prototypical histogram for each car. A large bin width (width = 10, N = 18) was chosen so that the histogram would be more invariant to shifts in ambient light spectrum.

4.2.1 Background Subtraction

Initially, despite the aforementioned choice of wide histogram bins, this approach performed poorly when subject to lighting and background variation. In particular, projecting multi-coloured backgrounds caused the observed histogram of a given car to be shifted such that it no longer matched the correct prototype and occasionally matched another car's prototype, resulting in misclassification.

To address this problem, background subtraction was investigated. That is, instead of calculating histograms directly from image frames, histograms were calculated using the *difference* between the current frame and a reference 'background image' containing no cars.

The global binary mask was also calculated differently in this approach. That is, the mask was calculated by converting the difference image to grayscale and then binary by applying a threshold. Figure 4.7 illustrates this process of comparing a given frame (4.7b) to the background image (4.7a), producing a difference image (4.7c) and binary mask (4.7d).



(a) Background image (no cars)

(b) Video frame (six cars)

(c) Difference image

(d) Binary global mask

Figure 4.7: Background subtraction process. Background image (4.7a) and video frame (4.7b) are compared to obtain a 'difference image' (4.7c) which is converted to grayscale and then binary by applying a threshold (4.7d).

Prototype histograms were re-calibrated for this approach using 3,000 images per car across bright, dark and coloured lighting. The resulting prototypes are presented in Figure 4.8. As shown, most have little overlap, with the main exception being orange and red, which overlap in the 0-40 range.

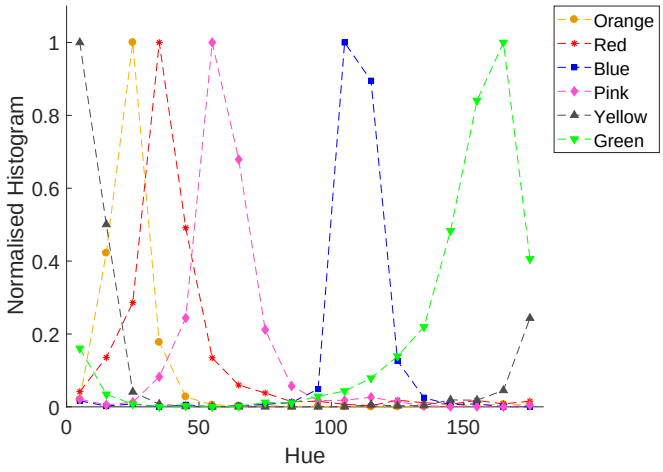


Figure 4.8: Prototype histograms for the six different coloured cars.

When further investigated, it was found that while the orange prototype matched all test instances of observed histograms for the orange car, the red prototype also matched 67% of these at a χ^2distance threshold of 4.5, which was empirically found to produce optimal results overall.

In practice, however, this did not harm performance since observed histograms for the orange car always matched the orange prototype *more closely* than the red prototype. This result is illustrated in Figure 4.9 which shows the χ^2distances for 1,000 observed histograms of the orange car calculated using both the red and orange prototypes. As shown, the orange prototype uniformly produces a smaller χ^2distance and therefore avoids misclassification.

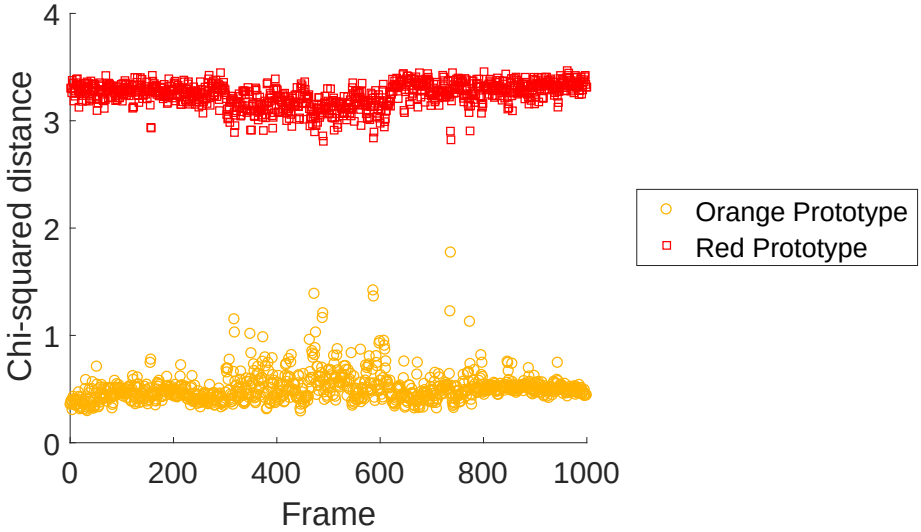


Figure 4.9: χ^2 distances for 1,000 observed histograms of the orange car compared to red and orange prototypes. As shown, the orange prototype produces a smaller χ^2distance in every case.

4.2.2 Colour Markers

In order to extend the number of distinguishable cars beyond the six cars shown in Figure 4.1, the use of unique colour markers was investigated. As discussed in the literature review, this approach has been demonstrated successfully by multiple authors. Figure 4.10 shows the three combinations of markers tested, each consisting of two colour squares.

Figure 4.10: Colour markers used to distinguish cars.

Calibration of the prototype histograms was completed as per Section 4.2. However, the results were quite poor. For example, the prototype for the car with blue and green markers matched observed histograms for the car with green and pink markers 64% of the time and while the car with yellow and pink markers 19% of the time. Similarly poor results were observed for the other combinations.

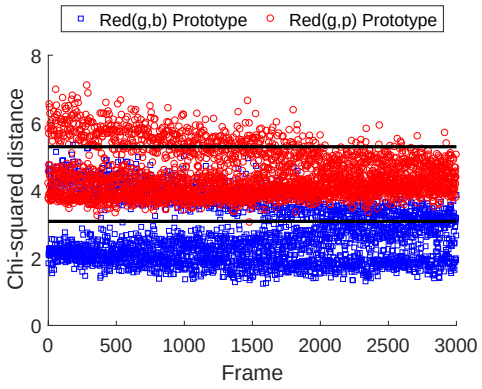
Unlike the orange and red prototypes discussed previously, it was not the case that the correct prototype reliably produced closer matches. Figure 4.11 shows two such cases. Overall, these results indicate that the use of colour markers to distinguish cars for the purpose of histogram comparison is ineffective.

4.3 Single Object Trackers

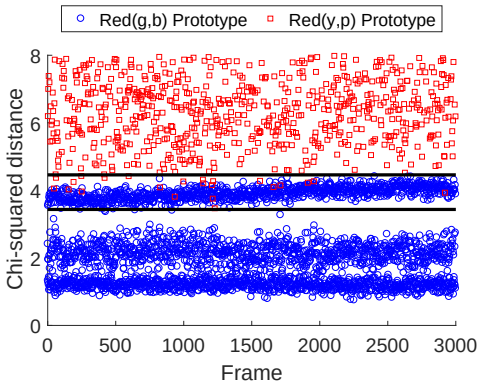
Two single-object tracking algorithms were investigated in this research: Kernalized Correlation Filters (KCF) [52] and Struck [53].

4.3.1 Kernalized Correlation Filters

Kernalized Correlation Filters (KCF) was selected for investigation due to its reasonably high accuracy, very high speed [43, 52] and API support within OpenCV. KCF uses a tracking-by-detection approach and on-line learning of a classifier for discriminating between the object of interest and the image background [52]. In regard to speed, KCF is able to process in excess of 170 frames per second on a 2.7 GHz Intel i7 CPU [43]. Overall, Hare et al. describe KCF as "the current state-of-the-art" [53, p. 2098].



(a) Observed histograms for car with green and pink markers (Red(g,p))



(b) Observed histograms for car with green and blue markers (Red(g,b))

Figure 4.11: χ^2 distances for 3,000 observed histograms of the car with green and pink markers ("Red(g,p)") (4.11a) and green and blue ("Red(g,b)") (4.11b) compared to different prototype histograms. Areas of overlap between correct matches (blue) and incorrect matches (red) are highlighted by horizontal black lines to emphasise that it is not possible to separate the χ^2distances produced for correct versus incorrect matches.

When tested in this research, KCF achieved 30 frames per second on the Raspberry Pi. This represents a decrease by a factor of approximately six relative to the aforementioned 170 frames per second due to the change in hardware. In tracking, however, KCF proved to be unreliable for fast moving cars. This is likely due to the finite search radius used when attempting to locate the target object in subsequent frames [52]. If a car moves sufficiently fast, it will move beyond this search radius, which will cause tracking to fail. This effect was observed by Galoogahi et al. [43] in that higher frame rates – with correspondingly smaller motion between frames – resulted in substantially higher performance for KCF. Enlarging this search radius was investigated. However, even at its maximum setting, car speeds above 0.25 m/s still caused tracking to fail.

4.3.2 Struck

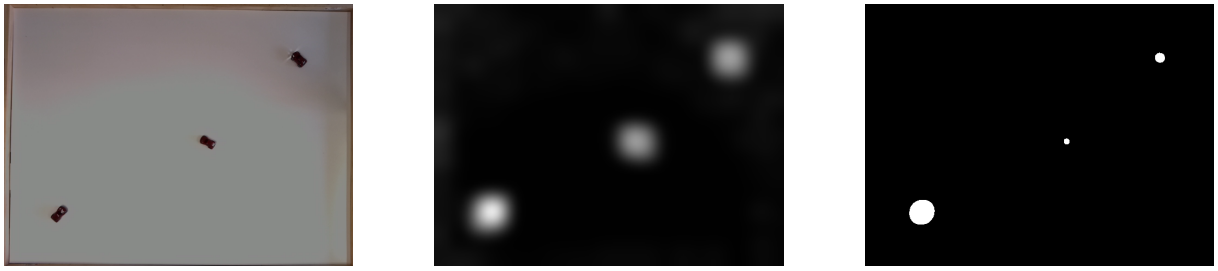
Struck was selected for investigation due to its high accuracy, reasonable speed [44, 53] and open-source C++ code [54]. Like KCF, this algorithm uses tracking-by-detection and on-line learning in which a support vector machine learns to classify the object of interest [53]. In a comprehensive review of object tracking, Wu et al. [44] found Struck to be among the most accurate algorithms while also achieving above 20 frames per second on a 3.4 GHz Intel i7 CPU.

When tested on the Raspberry Pi tracking a single car, 3.2 frames per second was achieved. This represents a similar decrease to that observed with KCF. It also fails requirement R1. Additionally, this speed could be expected to decrease significantly if multiple cars were tracked, requiring multiple trackers to run in parallel.

4.4 Saliency Detection

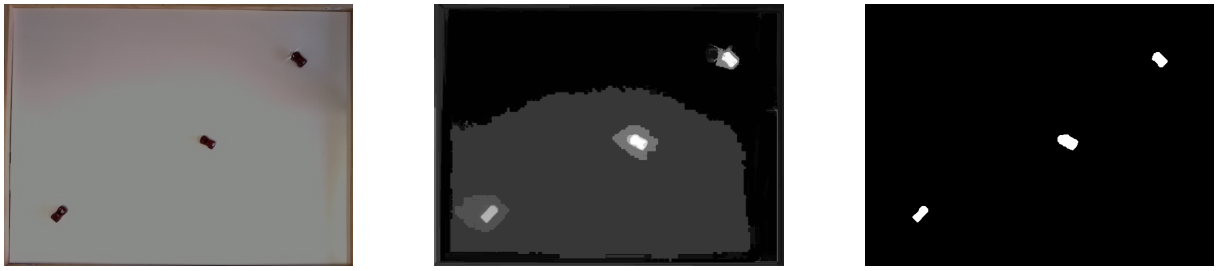
As described in the literature review, visual saliency can also be used to detect objects within an image [39]. Two algorithms from this field have been investigated. The first of these, Discriminative Region Feature Integration (DRFI), calculates a saliency map by segmenting the image, computing certain features over these regions and calculating saliency scores based on these features [55]. The second, Boolean Map based Saliency (BMS), derives saliency maps using a feature called "surroundedness", which measures the degree to which a region in enclosed [56]. Both of these algorithms performed near the top of their classes in a recent benchmark [39].

Preliminary experiments indicated that BMS was moderately effective at detecting cars in the testbed. However, this was somewhat sporadic and did not produce uniform results for each car, as shown in Figure 4.12. Results for DRFI were moderately better, as shown with the same source image in Figure 4.13. However, both algorithms were critically limited by processing speed. BMS required 1-2 seconds to process a single 640x480 pixel image on the Raspberry Pi. DRFI required a similar amount of time to process the same image on a 2.2 GHz quad core laptop. Both results are unacceptable for tracking purposes.



(a) Original image (b) Output of BMS algorithm (c) Binary output

Figure 4.12: BMS algorithm output with three cars. Significantly different binary masks are produced depending on their position within the frame.



(a) Original image (b) Output of BMS algorithm (c) Binary output

Figure 4.13: DRFI algorithm output with three cars. Reasonably even masks are produced for each car.

4.5 Peripheral Components

Peripheral components to the computer vision algorithm include interfacing with the camera, communication to the car controller algorithm and car state estimation. Interfacing with the Raspberry Pi Camera V2 was accomplished using the RaspiCam library [57]. Communication to the car controller was implemented using network sockets with the controller acting as the server and the computer vision system as a client. Information exchange is one-way with the computer vision system sending packets containing a timestamp and the current state of each detected object. The packets are structured as JSON strings using the format specified in Figure 4.14. Options for extensibility are included via the 'object type' entry and two spare entries at the end of the object field.

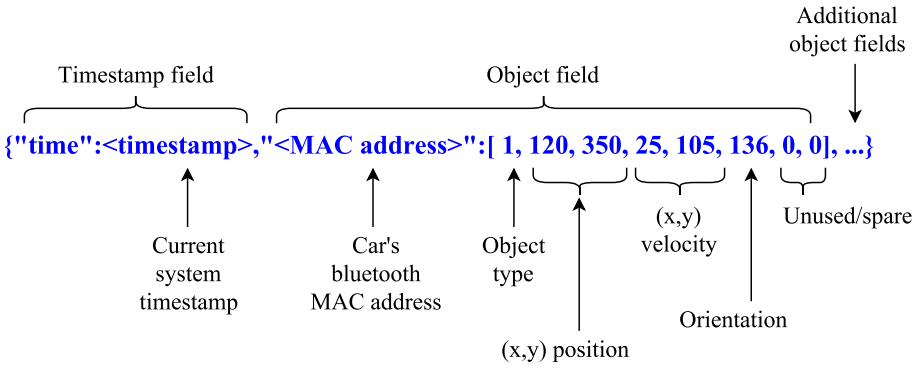


Figure 4.14: JSON string format for inter-program communication

For state estimation, car position – the output of the tracking algorithm – is first transformed from image coordinates to world coordinates by subtracting an 'origin' point and scaling appropriately. A scale factor of 1.93 was determined for pixel-to-mm conversion. Velocity is updated next by taking the difference between consecutive position measurements and dividing by the time interval according to the following equation:

$$\vec{v}_i = \frac{\vec{x}_i - \vec{x}_{i-1}}{t_i - t_{i-1}} \tag{4.2}$$

Where $\vec{x}_i$ and $\vec{x}_{i-1}$ are the current and previous car positions and t_i and t_{i-1} are the corresponding timestamps. Finally, orientation is estimated as the angle of the velocity vector.

Chapter 5

Results

This chapter details the validation experiments undertaken to evaluate each algorithm investigated with respect to the requirements defined in Section 3. Section 5.1 outlines algorithm speed performance (requirement R1). Section 5.2 presents the algorithm detection accuracy results (requirement R2). Section 5.3 describes state estimation accuracy (requirement R3). Section 5.4 addresses lighting and background variation performance (requirements R6 and R8).

5.1 Algorithm Speed

To compare the speed of each algorithm, three trials of 1,000 frames were conducted for one to six cars. The average speed results, in frames per second, are listed in Table 5.1.

Table 5.1: Algorithm speed comparison. Speeds shown in frames per second as the average of three 1,000 frame trials.

	Number of cars					
Algorithm	1	2	3	4	5	6
Hue Matching	17.8	12.9	10.0	7.7	7.2	6.0
Histogram Comparison	15.9	15.2	15.1	14.9	14.9	14.8
KCF	29.9	28.9				
STRUCK	3.2					
DRFI	<1					
BMS	<1					

Struck, DRFI and BMS were not fully tested since their speed failed to meet requirement R1 with even one car. KCF was not fully tested due to its inability to track fast moving cars. As such, all four were not tested further and are ignored for the remainder of this chapter. Chapter 4 provides additional discussion of these failings.

Hue matching and histogram comparison were both fully tested. Results for these algorithms are presented in Figure 5.1 with error bars representing one standard deviation above and below the mean. As shown, the speed for hue matching drops off far more rapidly than histogram comparison

as the number of cars increased. This is due to the nature of the computational tasks required for each additional car. Hue matching requires an additional matching operation in which the entire frame in scanned for pixels matching the car's known colour. Histogram comparison only requires an additional histogram calculation over a small region of the frame, which is *far* less computationally expensive. Consequently, hue matching suffers a much greater speed penalty with addition cars.

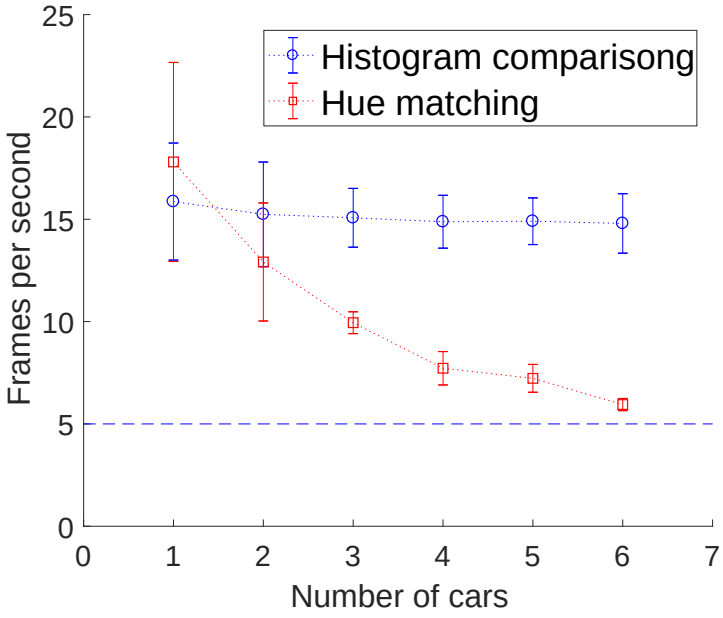


Figure 5.1: Hue matching and histogram comparison speed results with minimum 5 frames per second for requirement R1 shown.

5.2 Detection Accuracy

Testing of detection accuracy was completed for hue matching and histogram comparison. In this experiment, 6,000 frames showing all six cars under a variety of lighting conditions were tested with the percentage of correct detections for each car presented in Table 5.2. As shown, both algorithms achieve above 97% accuracy in all bit one case – hue matching with the orange car.

The reason for this anomaly is likely the proximity of the orange car's hue range to the 0-180 wrap-around point on the hue scale used by OpenCV (see Figure 4.3). If the observed hue of the orange car is shifted by non-neutral lighting, it may cause it to shift partially beyond the wrap-around point in the hue scale. If this occurs, hue matching algorithm may not detect the car.

Overall, though, accuracy comfortably exceeds the 95% benchmark set in requirement R2. Misclassification was also extremely rare with only a handful of cases observed across all data.

Table 5.2: Overall positive detection rates

Car	Hue Matching	Histogram Comparison
Orange	88.5	98.5
Red	100	100
Blue	100	98.1
Pink	97.7	99.5
Yellow	100	99.4
Green	100	100
Overall	**97.70**	**99.23**

5.3 State Estimation Accuracy

Measurement of state estimation accuracy was conducted by tracking stationary cars. This allows position accuracy to be measured via the standard deviation in position measurements and velocity noise to be measured via the average speed. Given the car is stationary, a perfect results would be zero in both cases.

Experimental results for these measures are presented in Tables 5.3 and 5.4. Each entry represents an average over three trials of 1,000 frames across different lighting conditions.

Table 5.3: State estimation accuracy: position uncertainty measured by position standard deviation (mm). Recorded values are the maximum of the standard deviations of x and y coordinates.

	Hue Matching		Histogram
Car	One Car	Six cars	Six cars
Orange	0.61	1.08	0.15
Red	0.36	0.72	0.14
Blue	0.34	0.28	1.25
Pink	0.45	0.25	0.23
Yellow	0.38	0.57	0.46
Green	0.65	0.60	0.32
Overall	**0.45**	**0.58**	**0.42**

As shown, hue matching and histogram comparison produce similar results with the exception being the performance of hue matching in average velocity with only one car (column 2 of Table 5.3). This is nearly three times greater than the corresponding measurement with six cars.

This result can be explained by considering the frame rate and how velocity is estimated (4.2). This formula takes the difference in position between frames and divides this by the time interval between them. For hue matching with a single car, the frame rate is approximately three times that for six cars. Consequently, the interval between frames is approximately one third, resulting

Table 5.4: State estimation accuracy: stationary average speed noise (mm/s)

	Hue Matching		Histogram
Car	**One Car**	**Six cars**	**Six cars**
Orange	11.22	6.13	1.24
Red	8.63	5.15	1.43
Blue	4.01	1.19	11.13
Pink	9.62	1.64	1.95
Yellow	9.14	3.22	5.46
Green	13.50	3.86	1.88
Overall	**9.35**	**3.53**	**3.85**

in the greatly amplified velocity noise observed in this experiment. In contrast, the frame rate for histogram comparison is relatively constant and the average velocity measured in this experiment did not vary noticeably with the number of cars.

5.4 Lighting & Background Variation

As mentioned above, the experiments in Sections 5.2 and 5.3 were conducted with a variety of lighting conditions. These include different projected backgrounds and ambient lighting. Overall, it was found that provided the overall lighting was not too dim, accurate detection and state estimation could be maintained in most cases. For example, Table 5.5 compares the detection accuracy results for neutral projected light and green light (one of the more extreme lighting scenarios). As shown, detection accuracy only drops significantly for the orange car in the hue matching approach. Performance drops for histogram comparison are never more than 3%.

Additionally, while there were no misclassifications under neutral lighting, green light introduced a small number of misclassifications. However, this was still comfortable below the 0.5% threshold set in requirement R2.

State estimation accuracy was unaffected by changes in lighting.

Table 5.5: Positive detection rate with different lighting/background

	Hue Matching		Histogram Comparison	
Car	Neutral	Green hue	Neutral	Green hue
Orange	99.8	77.3	100	97.0
Red	99.9	100	100	100
Blue	100	100	99.0	97.1
Pink	100	95.4	100	98.9
Yellow	100	100	99.9	98.8
Green	100	100	100	100
Overall	**99.95**	**95.46**	**99.81**	**98.64**

Chapter 6

Discussion

This chapter provides an evaluation of the results presented in Chapter 5 and the algorithms themselves against the requirements validation criteria specified in Section 3.4.

The results for algorithm speed (Section 5.1) show that only three algorithms – hue matching, histogram comparison and KCF – are able to achieve real-time tracking as per requirement R1 criteria. As this is a mandatory requirement, the remaining algorithms – Struck, DRFI and BMS – are rejected for this testbed. Note, however, that hue matching only meets requirement R1 by a small margin with six cars. Based on the graph in Figure 5.1, it is likely that it will not meet this requirement for seven or more cars.

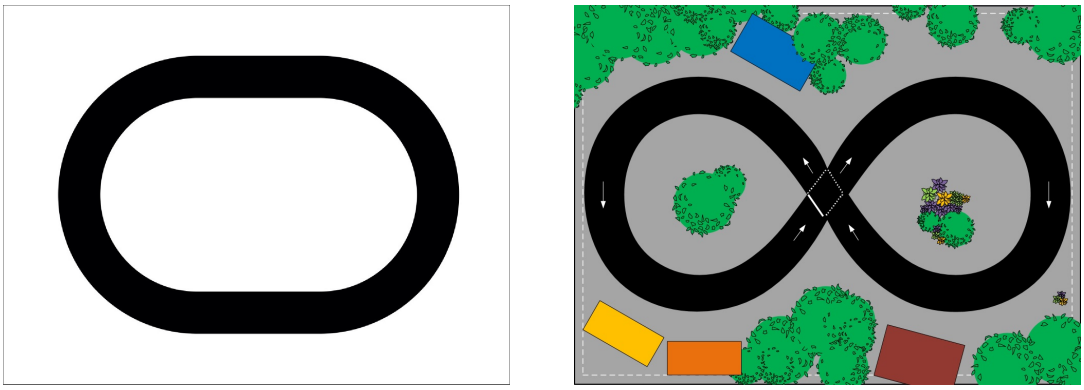
Detection accuracy results (Section 5.2) show that both hue matching and histogram comparison are able to meet requirement R2, with respect to both true and false positive rates. KCF was not assessed due to its inability to track fast moving cars (requirement R7).

State estimation accuracy was measured using two metrics. The low values observed for position standard deviation – 0.58 mm for hue matching and 0.42 mm for histogram comparison – demonstrate that both algorithms perform within the ± 2 mm position accuracy tolerance for requirement R3. That is, these algorithms are precise to within ± 0.58 mm and ± 0.42 mm, respectively, at one standard deviation. Satisfactory results were also attained for velocity estimation, with average velocity noise measurements of 3.35 mm/s and 3.85 mm/s both within the 5 mm/s tolerance.

However, with small numbers of cars, hue matching produced worse velocity noise results due to its increased frame rate in this case. With one car, its measured velocity noise of 9.35 mm/s nearly doubles the limit set in requirement R3.

The fourth set of results – lighting and background variation – are a question of degree. There will always be some lighting condition under which tracking fails. However, the results shown in Section 5.4 and the holistic performance of the testbed indicate that adequate performance with regard to requirements R6 and R8 has been achieved for hue matching and histogram comparison. With respect to holistic testbed performance, successful demonstrations have been completed with up to six cars using the two different background projections shown in Figure 6.1.

Histogram comparison, however, did experience less performance drop due to lighting changes and, unlike hue matching, was able to meet requirement R2 for each car under challenging lighting.



(a) Simple racetrack background (b) Complex figure of eight background

Figure 6.1: Two background images used for testing

Returning to the overall system goals (Section 3.1), support for up to eight cars was originally desired. Tracking has only been achieved for six cars at this stage due to only six different colour cars being available. Colour markers were investigated as an option for adding more; however, this did not prove workable.

The remaining requirements have been addressed as follows. Requirements R4 and R7 are met implicitly by all detection-based approaches (hue matching, histogram comparison, DRFI and BMS) due to these operating on a frame-by-frame basis. In contrast, Struck and KCF were unable to re-establish tracking following occlusion or track fast moving cars. Requirement R5 is met by all algorithms since none are affected in any way by car orientation.

Table 6.1 summarises the performance of each algorithm against the each requirement. Blank entries containing a '-' were not tested due to the algorithm having already failed a mandatory requirement. Overall, it can be seen that both hue matching and histogram comparison at least partially meet all requirements, with histogram comparison performing most highly. As mentioned, both of these algorithms provided effective computer vision for the testbed in practice.

Table 6.1: Requirements verification

ID	**Hue**	**Histogram**	**KCF**	**Struck**	**DRFI**	**BMS**
R1	Yes	Yes+	Yes	No	No	No
R2	Partial	Yes	-	-	-	-
R3	Partial	Yes	-	-	-	-
R4	Yes	Yes	No	No	Yes	Yes
R5	Yes	Yes	Yes	Yes	Yes	Yes
R6	Partial	Yes	-	-	-	-
R7	Yes	Yes	No	No	Yes	Yes
R8	Partial	Yes	-	-	-	-

Chapter 7

Conclusions

This research contributes to the development of a versatile, low-cost physical traffic testbed by providing an effective computer vision system. The utility of this testbed as a whole is twofold with it having applications in both research and public engagement.

In this research, six computer vision algorithms have been investigated for the task of tracking multiple small cars within the testbed. Two algorithms – hue matching and histogram comparison – were demonstrated to meet all design requirements, while the others – KCF, Struck, DRFI and BMS – failed to meet either real-time or accuracy requirements. Of the two successful algorithms, histogram comparison emerged as most efficacious due to its high computational efficiency, greater accuracy and robustness to lighting variation.

These results suggest that, on the whole, simpler approaches are more effective for this application, particularly considering the hardware constraint of running on a Raspberry Pi 3.

One limitation of the system developed is that only six cars can currently be tracked reliably. This is because only six different coloured cars are available at this time. The use of colour markers to distinguish cars and thereby extend the number of track-able cars was investigated, but did not produce workable results.

Overall, both this computer vision project and the development of the testbed as a whole in collaboration with two other students have been highly successful.

7.1 Future Work

While this project has been successful, multiple avenues exist for enhancement. Three suggestions are proposed here. First, car state estimation, particularly in regard to velocity noise, has room for improvement. In the literature, Kalman filtering is suggested as a standard tool for state estimation [32, 58]. Implementing this within the current computer vision framework would allow more precise state estimates to be produced, which would improve the performance of the car controller. This may become critical if high-precision control algorithms are to be used in future.

Second, alternative histogram comparison techniques could be investigated. Pele and Werman [36] present a "family" of such measures which are demonstrated to outperform current state-of-the-art approaches. Use of these more discriminative techniques could potentially improve the robustness of histogram comparison.

Third, the histogram comparison approach lends itself to machine learning techniques. In particular, it would not be a challenging problem to configure a neural network to learn the histograms of each car and classify them appropriately. Learning approaches typically perform well at computer vision tasks [43], suggesting that this may be a worthwhile endeavour.

References

[1] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies," *Journal of Modern Transportation*, vol. 24, no. 4, pp. 284–303, 2016.

[2] KPMG and Center for Automotive Research, "Self-driving cars: The next revolution," 2012. [Online]. Available: https://faculty.washington.edu/jbs/itrans/self_driving_cars%5B1%5D.pdf

[3] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs, "Disruptive technologies: Advances that will transform life, business, and the global economy," 2013.

[4] J. M. Anderson, K. Nidhi, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola, *Autonomous Vehicle Technology: A Guide for Policymakers*. Rand Corporation, 2016.

[5] M. Sivak and B. Schoettle, "Road Safety With Self-Driving Vehicles: General Limitations And Road Sharing With Conventional Vehicles," The University of Michigan, Tech. Rep., 2015.

[6] The Economist, "The driverless, car-sharing road ahead," 2016. [Online]. Available: http://www.economist.com/news/business/21685459-carmakers-increasingly-fret-their-industry-brink-huge-disruption

[7] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A*, vol. 77, pp. 167–181, 2015spu.

[8] S. Abuelsamid, D. Alexander, and L. Jerram, "Navigant Research Leaderboard Report: Automated Driving - Assessment of Strategy and Execution for 18 Companies Developing Automated Driving Systems," 2017. [Online]. Available: http://62.nl.dealer-preview.co/SiteAssets/LB-AV-17-Navigant-Research_FINAL.pdf

[9] A. Davies, "Detroit is stomping Silicon Valley in the self-driving car race," 2017. [Online]. Available: https://www.wired.com/2017/04/detroit-stomping-silicon-valley-self-driving-car-race/

[10] BBC, "Google's driverless cars make progress," 2017. [Online]. Available: http://www.bbc.com/news/technology-38839071

[11] Waymo, "On the Road," 2017. [Online]. Available: https://waymo.com/ontheroad/

[12] R. Kurjanowicz, "The DARPA Grand Challenge: Past and Future," *Computer*, vol. 39, no. 12, pp. 28–29, 2006.

[13] K. Iagnemma and M. Buehler, "Editorial for Journal of Field Robotics – Special Issue on the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 655–656, 2006.

[14] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. A. David Anderson, S. Cacciola, P. C. Patrick Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D. V. Covern, and M. Webster, "Odin: Team VictorTango's Entry in the DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[15] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[16] T. Litman, "Autonomous Vehicle Implementation Predictions: Implications for Transport Planning," Victoria Transport Policy Institute, Tech. Rep., 2017.

[17] B. Wernle, "Autonomous vehicle explosion brings challenges," *Automotive News*, vol. 89, no. 6683, p. 10, 2015.

[18] A. Waytz, J. Heafner, and N. Epley, "The mind in the machine: Anthropomorphism increases trust in an autonomous vehicle," *Journal of Experimental Social Psychology*, vol. 52, pp. 113–117, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jesp.2014.01.005

[19] M. Bahram, Z. Ghandeharioun, P. Zahn, M. Baur, W. Huber, and F. Busch, "Microscopic traffic simulation based evaluation of highly automated driving on highways," in *Proceedings of the 17th International Conference on Intelligent Transportation Systems*, 2014, pp. 1752–1757.

[20] M. Pursula, "Simulation of Traffic Systems - An Overview," *Journal of Geographic Information and Decision Analysis*, vol. 3, no. 1, pp. 1–8, 2016. [Online]. Available: http://publish.uwo.ca/~jmalczew/gida_5/Pursula/Pursula.html

[21] E. Thonhofer, T. Palau, A. Kuhn, S. Jakubek, and M. Kozek, "Macroscopic traffic model for large scale urban traffic network design," *Simulation Modelling Practice and Theory*, vol. 80, pp. 32–49, 2018.

[22] M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, and K. Nagel, "MATSim-T: Architecture and simulation times," in *Multi-Agent Systems for Traffic and Transportation Engineering*. IGI Global, 2009.

[23] X. Y. Lu, J. Lee, D. Chen, J. Bared, D. Dailey, and S. E. Shladover, "Freeway micro-simulation calibration: Case study using aimsun and VISSIM with detailed field data," in *Proceedings of the 93rd Annual Meeting of the Transportation Research Board, Washington, DC*, 2014.

[24] C. H. Hsieh, Y. L. Chuang, Y. Huang, K. K. Leung, A. L. Bertozzi, and E. Frazzoli, "An Economical Micro-Car Testbed for Validation of Cooperative Control Strategies," in *Proceedings of the American Control Conference*, 2006, pp. 1446–1451.

[25] K. K. Leung, C. H. Hsieh, Y. R. Huang, A. Joshi, V. Voroninski, and A. L. Bertozzi, "A second generation micro-vehicle testbed for cooperative control and sensing strategies," in *Proceedings of the American Control Conference*, 2007, pp. 1900–1907.

[26] M. Gonzalez, X. Huang, D. S. H. Martinez, C. H. Hsieh, Y. R. Huang, B. Irvine, M. B. Short, and A. L. Bertozzi, "A Third Generation Micro-vehicle Testbed for Cooperative Control and Sensing Strategies," in *Proceedings of the International Conference on Informatics in Control, Automation and Robotics*, 2011, pp. 14–20.

[27] D. H. Martinez, M. Gonzalez, X. Huang, B. Irvine, C. H. Hsieh, Y. R. Huang, M. B. Short, and A. L. Bertozzi, "An Economical Testbed for Cooperative Control and Sensing Strategies of Robotic Micro-Vehicles," in *Proceedings of the International Conference on Informatics in Control, Automation and Robotics*, 2013, pp. 65–75.

[28] R. A. Cortez, J. M. Luna, R. Fierro, and J. Wood, "A multi-vehicle testbed for multi-modal, decentralized sensing of the environment," in *Proceedings of the International Conference on Robotics and Automation*, 2010, pp. 1088–1089.

[29] C. Yan and Q. Zhan, "Real-time multiple mobile robots visual detection system," *Sensor Review*, vol. 31, no. 3, pp. 228–238, 2011.

[30] J. D. Suter, "Miniature Vehicle Testbed for Intelligent Transportation Systems," Honours Thesis, Ohio State University, 2016.

[31] M. Asada, H. Kitano, I. Noda, and M. Veloso, "RoboCup: today and tomorrow - what we have learned," *Artificial Intelligence*, vol. 110, no. 2, pp. 193–214, 1999.

[32] M. Brezak, I. Petrović, and E. Ivanjko, "Robust and accurate global vision system for real time tracking of multiple mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 3, pp. 213–230, 2008.

[33] D. Lavy and T. Marshall, "Autonomous Navigation with NAO," 2015. [Online]. Available: http://www.bu.edu/vip/files/pubs/reports/DLTM15-06buece.pdf

[34] M. J. Swain and D. H. Ballard, "Color Indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–31, 1991.

[35] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel svms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 66–77, 2013.

[36] O. Pele and M. Werman, "The quadratic-chi histogram distance family," in *European Conference on Computer Vision*, 2010, pp. 749–762.

[37] A. Borji, M.-M. Cheng, H. Jiang, and J. Li, "Salient Object Detection: A Survey," 2014. [Online]. Available: http://arxiv.org/abs/1411.5878

[38] A. Borji, "What is a salient object? A dataset and a baseline model for salient object detection," *IEEE Transactions on Image Processing*, vol. 24, no. 2, pp. 742–756, 2015.

[39] A. Borji, M. Cheng, H. Jiang, and J. Li, "Salient Object Detection: A Benchmark," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5706–5722, 2015.

[40] Z. Liang, B. Xu, Z. Chi, and D. D. Feng, "Relative Saliency Model over Multiple Images with an Application to Yarn Surface Evaluation," *IEEE Transactions on Cybernetics*, vol. 44, no. 8, pp. 1249–1258, 2014.

[41] Y. Zhai and M. Shah, "Visual attention detection in video sequences using spatiotemporal cues," in *Proceedings of the ACM International Conference on Multimedia*, 2006, pp. 815–824.

[42] G. Zhang, Z. Yuan, N. Zheng, X. Sheng, and T. Liu, "Visual saliency based object tracking," in *Proceedings of the 9th Asian Conference on Computer Vision*, 2009, pp. 193–203.

[43] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for Speed: A Benchmark for Higher Frame Rate Object Tracking," 2017. [Online]. Available: http://arxiv.org/abs/1703.05884

[44] Y. Wu, J. Lim, and M.-h. Yang, "Object Tracking Benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, vol. 37, no. 9, pp. 1834–1848, 2015.

[45] D. A. Klein, D. Schulz, S. Frintrop, A. B. Cremers, D. A, S. Klein Dirk, F. Simone, and C. Armin B., "Adaptive real-time video-tracking for arbitrary objects," pp. 772–777, 2010.

[46] H. Shao, W. Na, and S. Huajun, "Object Tracking Method Based on SURF," *AASRI Procedia*, vol. 3, pp. 351–356, 2012.

[47] O. Samuelsson, "Video Tracking Algorithm for Unmanned Aerial Vehicle Surveillance," Masters Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2012.

[48] S. A. P. Quintero and J. P. Hespanha, "Vision-based target tracking with a small UAV: Optimization-based control strategies," *Control Engineering Practice*, vol. 32, pp. 28–42, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.conengprac.2014.07.007

[49] ZenWheels, "ZenWheels Micro Car," 2017. [Online]. Available: http://zenwheels.com/zenwheels-micro-car-red.html

[50] OpenCV Team, "About," 2017. [Online]. Available: http://opencv.org/about.html

[51] T. Bräunl, *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*, 3rd ed. Springer Science & Business Media, 2008.

[52] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[53] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 38, no. 10, pp. 2096–2109, 2016. [Online]. Available: ttp://dx.doi.org/10.1109/TPAMI.2015.2509974

[54] S. Hare, "struck," 2015. [Online]. Available: https://github.com/samhare/struck

[55] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li, "Salient object detection: A discriminative regional feature integration approach," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2083–2090, 2013.

[56] J. Zhang and S. Sclaroff, "Exploiting Surroundedness for Saliency Detection: A Boolean Map Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 5, pp. 889–902, 2016.

[57] R. M. Salinas and J. Larson, "RaspiCam: C++ API for using Raspberry camera with/without OpenCv," 2017. [Online]. Available: https://www.uco.es/investiga/grupos/ava/node/40

[58] R. Yao, Y. Zhang, Y. Zhou, and S. Xia, "Multiple small objects tracking based on dynamic Bayesian networks with spatial prior," *Optik-International Journal for Light and Electron Optics*, vol. 125, no. 10, pp. 2243–2247, 2014.