NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DIOGO PIRES REGO MAYMONE

TRAFFIC RED LIGHT VIOLATION DETECTION UTILIZING IMAGE PROCESSING

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

TRAFFIC RED LIGHT VIOLATION DETECTION UTILIZING IMAGE PROCESSING

DIOGO PIRES REGO MAYMONE

Adviser: José Manuel Matos Ribeiro da Fonseca
Associate Professor with Aggregation, NOVA University Lisbon

Examination Committee

Chair: Luís Filipe Lourenço Bernardo
Associate Professor with Aggregation, NOVA University Lisbon

Rapporteur: André Damas Mora
Assistant Professor, NOVA University Lisbon

Adviser: José Manuel Matos Ribeiro da Fonseca
Associate Professor with Aggregation, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2023

Traffic red light violation detection utilizing Image Processing

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.10.10) [1].

To everyone who helped make this thesis possible

Acknowledgements

I am profoundly thankful to Professor José Fonseca of the NOVA School of Science and Technology for his unwavering guidance and mentorship throughout my thesis journey. The support provided by him has been instrumental in shaping the trajectory of my research. I also extend my sincere gratitude to the NOVA School of Science and Technology for fostering an environment conducive to academic growth and exploration. Additionally, I wish to express my appreciation to Professor André Mora for his valuable assistance, which significantly contributed to the development of my work.

In this moment of reflection, I am deeply grateful to my family and friends, whose constant encouragement and support have been the cornerstone of my academic endeavors. Their unwavering belief in me has provided the motivation to reach new heights. I would also like to extend my gratitude to the 99% and Ginkgo Biloba institutions for their contribution to this research.

The culmination of this thesis would not have been possible without the collective backing of these individuals and entities. Their support has shaped my academic journey in immeasurable ways, and for that, I am eternally thankful.

"A lesson without pain is meaningless. That's because no one can gain without sacrificing something. But by enduring that pain and overcoming it, he shall obtain a powerful, unmatched heart. A fullmetal heart. " (Edward Elric)

Abstract

Red-light violations are a major safety concern, as they can lead to serious accidents and fatalities. In the US, in 2020, an estimated 116,000 people were injured in red light running crashes and 928 people were killed in these same crashes. Half of those killed were pedestrians, bicyclists, and people in other vehicles who were hit by the red light runners.

Beyond the human toll, these violations also exact a substantial economic toll due to the costs associated with crashes and the resultant injuries. Consequently, the importance of developing effective strategies for detecting and preventing red light violations cannot be overstated, given the potential to enhance overall traffic safety.

This thesis aims to present a first look at an alternative solution to the problem of red light violations using a mobile setup, and image processing technologies and techniques. The core objective of this approach is to detect and identify instances in which vehicles break the rules as it happens, using a mobile camera setup that can be more easily set up and removed than the already existing fixed systems. This system aims to identify the specific vehicle and capture a picture and a short video of the violation, which later can be used to extract important information for further action. While this implementation is at a prototype level and only works with pre-recorded videos, the program's effectiveness was assessed through practical experimentation involving real-life traffic videos captured from various viewpoints, alongside genuine traffic light scenarios subjected to diverse lighting conditions and angles.

At the core of this approach lies the use of image processing and detection techniques. The former equips us to identify occurrences of traffic signals displaying a red light (as well as green and yellow), while the latter facilitates the monitoring of object motion, offering insights into movement and direction. By using these two techniques we are able to accurately detect when a red light is on, and by analyzing a specified prohibited ROI and direction we can detect vehicles trespassing in this zone. Moreover, to enhance the robustness of the system, we employ a creative approach to adaptively detect the traffic light color, which is pivotal for accurate infraction detection.

To handle the varying brightness conditions that transpire throughout the day, we

utilize brightness detection as opposed to color detection when analyzing the traffic lights. However, achieving adaptability in this context without using complex ML or AI techniques necessitates a more creative approach. Thus, we implement a K-means clustering technique that operates on the basis of a traffic light color cycle. This dynamic process enables us to pinpoint the two main clusters that correspond to the light being on and off, independent of the surrounding brightness. By harnessing this adaptative approach, we create a more resilient method for detecting traffic lights under varying illumination conditions, thus enhancing the overall robustness and accuracy of the system.

The outcomes indicate the feasibility of infraction detection using this approach, which can potentially be implemented in real-time and provide a more accessible means of addressing the issue. We hope that by catching and preventing red light violations, we can make the roads safer and reduce the number of accidents and injuries. This thesis aims to contribute to the field of red light violation detection, creating a better and safer environment for drivers and pedestrians on the roads.

Keywords: Red-Light Violation, Image Processing, Image Detection, Optical Flow, Traffic Violations, K-Means Clustering

Resumo

As infração de sinais vermelhos constituem uma preocupação significativa em termos de segurança, uma vez que podem resultar em acidentes graves e em fatalidades. Nos EUA, no ano de 2020, estima-se que tenham ocorrido cerca de 116.000 feridos em acidentes relacionados com a infração de semáforos vermelhos, tendo havido igualmente 928 vítimas mortais nestes mesmos acidentes. Metade das vítimas mortais eram peões, ciclistas e ocupantes de outros veículos que foram atingidos por condutores que desrespeitaram a sinalização.

Para além do impacto humano, essas infrações também acarretam um custo significativo do ponto de vista económico, devido aos custos associados aos acidentes e aos ferimentos resultantes. Desta forma, é de extrema importância desenvolver métodos eficazes para detetar e prevenir tais infrações, com o objetivo de melhorar a segurança rodoviária.

Esta tese tem como objetivo apresentar uma primeira abordagem alternativa para o problema das infrações de semáforos usando um setup móvel e recorrendo a tecnologias e técnicas de processamento de imagem. O objetivo central desta abordagem é detetar e identificar casos em que os veículos quebram as regras no momento que esta ocorre, utilizando um setup movél que pode ser mais facilmente montado e removido do que os sistemas já existentes. O nosso sistema visa identificar o veículo em questão e capturar uma fotografia, bem como um breve vídeo da infração, permitindo posteriormente a extração de informação pertinente para ações futuras. Embora esta implementação esteja a um nível de protótipo e só funcione com vídeos pré-gravados, o programa foi avaliado através de testes práticos que envolveram o uso de vídeos de trânsito em tempo real captados de diversos ângulos, juntamente com situações reais de semáforos sujeitos a condições de iluminação e ângulos variados.

No centro desta abordagem encontra-se a utilização de técnicas de processamento e de deteção imagem. O primeiro permite-nos identificar situações em que os sinais de trânsito exibem a luz vermelha (bem como verde e amarela), enquanto o segundo facilita a monitorização do movimento de objetos, fornecendo informações sobre a direção e o deslocamento. Através da combinação destas duas técnicas, conseguimos detetar com

precisão quando o sinal está vermelho e, ao analisar uma área restrita especificada e a respetiva direção, conseguimos identificar veículos que transgridem a zona proibida. Além disso, para melhorar a robustez do nosso sistema, adotamos uma abordagem criativa para detetar a cor do semáforo de forma adaptativa, um aspeto crucial para a deteção precisa de infrações.

Para lidar com as variações de brilho que ocorrem ao longo do dia, optamos por detetar brilho em vez de cor ao analisar os semáforos. No entanto, para alcançar adaptabilidade neste contexto sem recorrer a técnicas de ML ou IA complexas, é necessária uma abordagem mais criativa. Assim, implementamos uma técnica de K-means clustering que opera na base de um ciclo completo de cores do semáforo. Este processo dinâmico permite-nos identificar os dois principais clusters que correspondem à luz acesa e apagada, independentemente do brilho. Através desta abordagem adaptativa, o método torna-se mais resiliente para detetar semáforos sob condições de iluminação variável, aumentando a robustez e a precisão do nosso sistema.

Os resultados indicam a viabilidade da deteção de infrações utilizando esta abordagem, que pode potencialmente ser aplicada em tempo real e proporcionar um meio mais acessível de abordar este problema. Esperamos que, ao detetar e prevenir infrações de em sinais vermelhos, possamos tornar as estradas mais seguras e reduzir o número de acidentes e ferimentos. Esta tese tem como objetivo contribuir para o campo da deteção de infrações de sinais vermelhos, criando um ambiente rodoviário melhor e mais seguro para condutores e peões.

Palavras-chave: Infração do sinal vermelho, Processamento de imagem, Detecção de imagem, Optical Flow, Violação de regras rodoviárias, K-Means Clustering

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AI Artificial Intelligence *(p. vii)*
ATS American Traffic Solutions *(p. 18)*

BGR Blue, Green, Red *(p. 6)*

CMYK Cyan, Magenta, Yellow, Key *(p. 6)*
CNN Convolutional Neural Network *(pp. 22, 24)*

DBSCAN Density-Based Spatial Clustering of Applications with Noise *(p. 65)*

EUA Estados Unidos das América *(p. ix)*

GMM Gaussian Mixture Model *(p. 65)*

HSL Hue, Saturation, Lightness *(p. 6)*
HSV Hue, Saturation, Value *(pp. xiii, 6, 7, 26, 41)*

IA Inteligência Artificial *(p. x)*

KNN K-Nearest Neighbor *(pp. 8, 9)*

LK Lucas-Kanade *(pp. xiii, 12–14, 64)*

ML Machine Learning *(pp. vii, x)*
MOG2 Mixture of Gaussians 2 *(pp. xiii, 8, 9, 32, 33, 64)*

RGB Red, Green, Blue *(pp. xiii, 6, 7, 26–28)*
ROI Region of Interest *(pp. vi, xiii, xiv, 1, 2, 26, 39, 40, 45–47, 49–52)*

UI User Interface *(pp. 39, 49)*

US United States *(pp. vi, 1)*

YOLOv5s You Only Look Once version 5 small *(p. 22)*

1 Introduction

The objective of this thesis is to propose an alternative solution to the problem of red light violations using a mobile setup, and image processing technologies and techniques. Our motivation for this work stems from the desire to improve road safety and reduce the negative impacts of red light violations on society.

Traffic red light violations are a major safety concern on roads and highways worldwide. These violations can lead to accidents and injuries and are a significant contributor to traffic congestion and delays [2]. In the US, in 2020, an estimated 116,000 people were injured in red light running crashes and 928 people were killed in these same crashes. Half of those killed were pedestrians, bicyclists, and people in other vehicles who were hit by the red light runners [2]. Traditional methods for detecting and stopping red light violations, such as traffic cameras and law enforcement officers, are either extremely expensive or are limited in their ability to accurately and consistently identify offenders.

To address this problem, a solution that utilizes image processing to detect when a red light is on, and image detection to detect when a vehicle ignores the red light, is proposed.

Image processing involves the analysis of images and/or video footage to identify and classify objects and patterns, in this case, the existence of a red light signal. There are a number of different techniques to achieve this, yet the final decision was to use a combination of brightness detection and k-means clustering to enhance the system's robustness. This approach not only allows us to accurately discern the presence of a red light, but it also addresses challenges posed by varying illumination conditions throughout the day. The dynamic adaptability introduced by k-means clustering enables our system to reliably differentiate between the "on" and "off" states of the traffic light, regardless of surrounding brightness levels. [3].

Image detection and more specifically Optical Flow, is a technique for tracking the movement of objects in a scene, allowing for the detection of vehicle movement, speed, and direction. This technique is especially interesting and useful for this approach, as the goal is to detect vehicles moving in a certain direction, in a defined area where they shouldn't be, which is exactly what was implemented. Our approach prompts the user to draw an ROI and direction vector, consequently flagging only the vehicles moving in that

general direction, inside that chosen ROI [4].

By combining these two approaches, the aim was to create a system that accurately and consistently detects when a red light is on, and in that situation detects if any vehicle ignores it. This system has the potential to be implemented in real-time and improve road safety by identifying and penalizing red light violators, reducing the number of violations by introducing a more apparent way of detecting said violations, and deterring drivers from committing them [5].

In the following sections of this thesis, a detailed description of the already existing solutions and implementations of red light detection systems will be provided, both with and without the use of image processing, as well as their effectiveness and feasibility within our scope. It will also be expanding more into some methods and algorithms used for image detection and Optical Flow and discussing our implementation of these.

1.1 Objective

As stated before, the main objective of this thesis was to develop a solution for detecting traffic red light violations using image processing techniques that is robust and reliable enough, but also lightweight, mobile, and inexpensive. The importance of this project stems from the limited availability of affordable and lightweight solutions currently available for this specific problem.

1.2 Material

The code developed for this thesis is available online to those interested in the practical aspect of this solution, as well as the dataset used to test it. Both the Python code and the dataset used for this solution can be found here: https://github.com/diogomaymone/Red_Light_Violation_Detection_System

1.3 Structure of the thesis

This body of work comprises not only the current introduction but also five additional chapters, organized as follows:

- **Theory** – This section offers an all-encompassing exploration of the fundamental principles and concepts regarding traffic red light violation detection, image processing and detection, more specifically brightness detection, k-means clustering, movement/direction detection using Optical Flow, among other techniques.
- **State of the Art** – Within this chapter, a deep dive into the existing literature on traffic red light violation detection is undertaken. Specifically, the focus is directed toward solutions that use image processing and detection, like our own.

- **Practical Approach** – This segment details the methodology used in this study for the creation of the proposed traffic red light violation detection solution.
- **Findings** – Presented within this chapter are the obtained outcomes derived from the tests conducted to gauge the efficacy of the proposed solution.
- **Concluding Reflections and Future Work** – In the last chapter, a synthesis of the primary discoveries and deductions is given. Noteworthy contributions stemming from the proposed solution are explained, alongside suggestions for potential trajectories of future work aimed at refining and completing the solution proposed.

2

Theory

This chapter delves into the fundamental principles and key concepts underlying the domains of red light detection and image processing. This section provides an in-depth look at key components such as brightness detection, k-means clustering, and Optical Flow based movement/direction detection, among other techniques. This chapter helps the reader understand some of the technologies and techniques used throughout this thesis by delving into these fundamental elements.

2.1 Image processing

Image processing is the manipulation and analysis of digital images using computational techniques and algorithms. It plays a crucial role in many fields, including computer vision, robotics, medicine, security, and in our specific case, traffic safety, and it has a wide range of applications, including image and object recognition, pattern recognition, image enhancement, and movement detection, among others [6].

There are many different techniques and algorithms that can be used for image processing, and these techniques differ in their complexity, accuracy, and computational requirements, and the choice of which to use depends on the specific application and the resources available [7].

For the vehicle detection portion, various image detection and processing techniques such as sparse Optical Flow (Lucas-Kanade), background subtraction, good feature tracking (Shi-Tomasi corner detection), and morphological operations, among others are used to detect when a vehicle ignores the red light, so, these techniques will be the main focus of this section, where we will explore what they are as well as how they work.

2.1.1 Color Space Conversion

Color spaces are numerical representations of colors that can be manipulated and processed. These are essentially mathematical models that define a specific spectrum of colors suitable for display or printing. A predetermined collection of primary colors is

designated within each color space, serving as the foundation for generating all other colors within that specific color space [8].

Some of the most commonly used color spaces are:

- **RGB (Red, Green, Blue):** Employed for computer monitors and TVs.
- **CMYK (Cyan, Magenta, Yellow, Key):** Utilized for printing purposes.
- **HSV/HSL (Hue, Saturation, Value/Lightness):** Commonly applied in image processing and computer vision applications.

Each color space has its own advantages and is used for different purposes. In our case, the video input is in RGB but we used HSV for its image processing and computer vision applications because it separates the hue information from the saturation and value/lightness information, making it easier to manipulate these attributes independently [8].

Color space conversion is the process of changing the representation of a color from one basis to another. In image processing, color space conversion is used to represent colors in ways that make certain calculations more convenient or to identify colors more intuitively [9]. The conversion from BGR to HSV is done through a mathematical transformation that maps the values of the blue, green, and red color channels in the BGR color space to the hue, saturation, and value components in the HSV color space [10]. In the case of 8-bit and 16-bit images, R, G, and B are converted to the floating-point format and scaled to fit the 0 to 1 range [10].

$$V \leftarrow max(R,G,B)$$

$$S \leftarrow \begin{cases} \frac{V-min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G-B)/(V-min(R,G,B)) & \text{if } V = R \\ 120+60(B-R)/(V-min(R,G,B)) & \text{if } V = G \\ 240+60(R-G)/(V-min(R,G,B)) & \text{if } V = B \\ 0 & \text{if } R = G = B \end{cases}$$

Figure 2.1: RGB to HSV conversion [10]

If $H < 0$ then $H \leftarrow H + 360$. On output $0 \leq V \leq 1, 0 \leq S \leq 1, 0 \leq H \leq 360$

The values are then converted to the destination data type:

- 8-bit images: $V \leftarrow 255V, S \leftarrow 255S, H \leftarrow H/2$ (to fit to 0 to 255)
- 16-bit images: $V < -65535V, S < -65535S, H < -H$
- 32-bit images: H, S, and V are left as is

2.1.2 Brightness Detection and Calculations

In the context of this thesis, brightness detection, and calculation, more specifically mean brightness calculation, refers to the image processing technique for quantifying the average intensity or brightness of an image or a specific region of interest. This calculation returns one value that represents the overall brightness level of the pixels in the selected area.

Converting an image to a suitable color space where the intensity information is more easily accessible is sometimes needed for mean brightness calculations. In our case, the Hue, Saturation, Value (HSV) color space was used, which was accomplished using previously described techniques, as it is more appropriate than RGB, which is not ideal for assessing brightness independently from color. The Value (V) representing the brightness of each pixel in the image can then be extracted and used to calculate the mean brightness value [9].

The reason for choosing brightness detection instead of color detection when analyzing traffic light colors will be explained in greater detail later in this thesis. However, it's important to highlight that this decision played a crucial role in enhancing the robustness of our detection method and reducing its sensitivity to changes. To achieve this, we also incorporated K-means clustering to provide adaptability, which we'll discuss further in the next subsection.

2.1.3 K-Means Clustering

K-means Clustering is a type of unsupervised learning algorithm used to categorize data into a set number of clusters. The algorithm works by assigning each data point iteratively to the cluster with the closest mean, then recalculating the mean of each cluster until convergence [11].

As mentioned before, the K-Means Clustering algorithm works by identifying similar observations in a dataset and grouping them into distinct groups. The first step involves randomly assigning each data point to an initial cluster and computing the centroid for each cluster. A centroid is the cluster's central point. Notably, certain variations of the process allow the user to explicitly define the initial clusters [11] [12] [13]. The algorithm then proceeds as follows:

- **Assignment Phase:** Each observation is evaluated by the algorithm by assigning it to the cluster with the closest centroid. The Euclidean distance between a data point and the centroid of a cluster, which must be shorter than the distances to the centroids of other clusters, determines the "closeness" criterion.
- **Centroid Recalculation:** When a cluster adds or removes a data point, the K-Means Clustering algorithm recalculates the cluster's centroid.
- **Iterative Refinement:** The algorithm iteratively repeats these steps until it can no longer reassign data points to a nearby cluster.

When the algorithm is finished, all clusters have the smallest within-cluster variance, ensuring their compactness. Clusters with low variance and size imply that the data points contained within them are very similar. While there are differences in characteristics within each cluster, the algorithm works to reduce this variability [11] [12] [13].

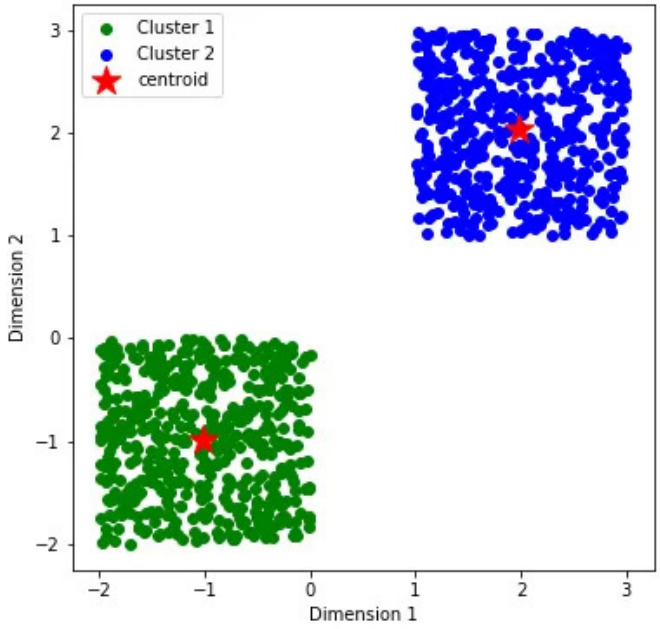

Figure 2.2: Example of two clusters and their respective centroids [14]

K-means Clustering is a key technique in our implementation, and it is critical in accurately determining the state of the traffic lights. K-means assists the program in determining whether the traffic lights are lit or not by categorizing the extracted brightness values into two distinct groups. This adaptability to changing lighting conditions allows the program to a make more robust analysis. This allows for quantitative analysis, adaptation to different lighting conditions, automatic decisions, and an overall increase in the program's reliability in detecting traffic light colors.

2.1.4 Background Subtraction

Background subtraction is a fundamental technique in computer vision used to separate moving objects in a video sequence from a stationary or slowly changing background. It is commonly employed in applications such as object detection, surveillance, and motion analysis [15].

In our study, we compared two different background subtraction algorithms: K-Nearest Neighbors (KNN) and Mixture of Gaussians 2 (MOG2). These algorithms are only a subset of the larger selection available, all of which aim to improve result accuracy in a variety of scenarios. Following a thorough evaluation, it became clear that the MOG2 algorithm

consistently produced superior results within our dataset. Notably, it performed well in terms of noise reduction. Also, MOG2 outperforms KNN in terms of speed, which is important in the context of our thesis. Additionally, MOG2 has higher accuracy in dealing with difficult factors such as shadows and abrupt environmental changes, which can occur when analyzing regions with high movement [16] [17] [18] [19].

MOG2 stands for Mixture of Gaussians 2, an algorithm designed for Background/ Foreground Segmentation based on Gaussian Mixture models. It draws its inspiration from two seminal papers by Z. Zivkovic: "Improved adaptive Gaussian mixture model for background subtraction" in 2004 [20], and "Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction" in 2006 [21].

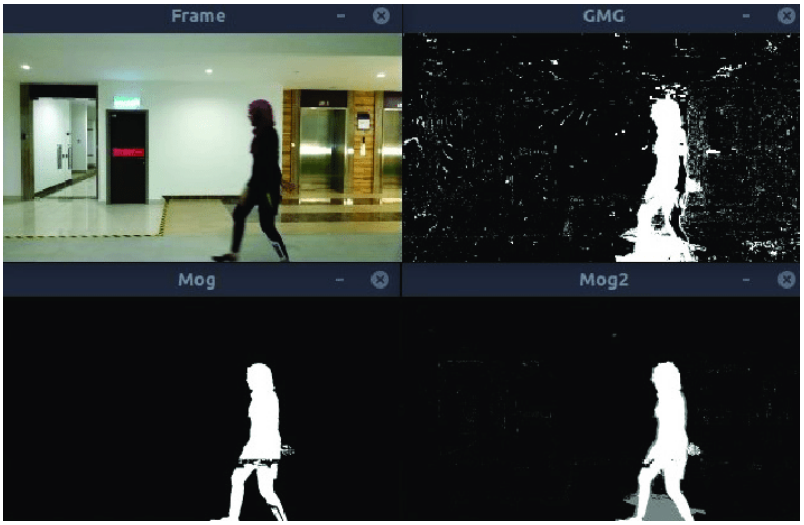

Figure 2.3: Comparison between background subtraction techniques, including MOG2 [22]

A defining trait of MOG2 lies in its ability to autonomously determine the optimal count of Gaussian distributions for every pixel. This innovative capability heightens MOG2's adaptability to dynamic environments, particularly those influenced by shifting illuminations [20] [21].

The operational principle of MOG2 involves representing each pixel within an image as a blend of K Gaussian distributions, where K signifies the count of Gaussian distributions employed for pixel modeling. The mixture's weights symbolize the temporal proportions that these color patterns endure within the scene. Notably, colors with longer-lasting stability are considered probable background elements. MOG2 employs an adaptable mechanism to ascertain the suitable number of Gaussian distributions for each pixel, thereby enhancing its proficiency in managing scenarios characterized by dynamic backgrounds and alterations in illumination conditions [20] [21].

2.1.5 Corner Detection

Corner detection is a technique used in image processing to identify locations within an image that have distinct variations in brightness or significant curvature along the edge curve. Its primary goal is to improve image processing efficiency and experimental accuracy by preserving important image properties. This method enables reliable feature matching in image applications and is widely used in tasks such as motion detection, image registration, video tracking, 3D reconstruction, and object recognition, to name a few [23].

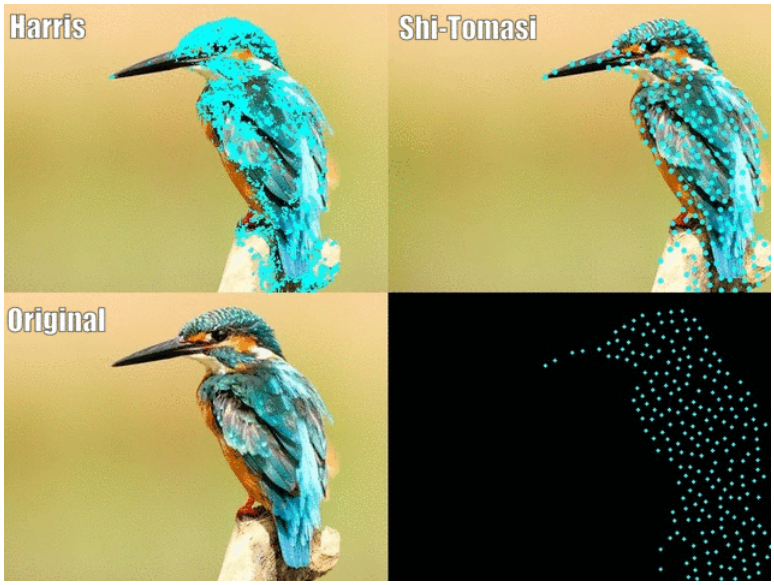

Figure 2.4: Comparison between corner detection techniques [24]

Shi-Tomasi corner detection is a technique employed in computer vision for identifying corners within images. It builds upon the Harris corner detection technique, offering improved performance and rotational invariance. Unlike the Harris scoring function, Shi-Tomasi utilizes the smaller of the two eigenvalues of the structure tensor matrix. This approach has been empirically proven to yield more accurate outcomes compared to its predecessor. The algorithm involves subtracting the determinant of a matrix M from its trace and comparing the result against a predetermined threshold. If the smaller eigenvalue exceeds this threshold, a robust corner point is identified. This method also addresses certain limitations of the Harris algorithm, making it particularly effective in tracking moving objects. In the context of Shi-Tomasi Corner Detection, when considering a window (W) positioned at coordinates (X, Y) and having a pixel intensity I(X, Y), the pertinent formula is as follows: [23] [25]:

$$f(X,Y) = \sum (I(X_k, Y_k) - I(X_k + \Delta X, Y_k + \Delta Y))^2 \tag{2.1}$$

where

$$(X_k, Y_k) \in W \tag{2.2}$$

Corner detection can be used in the context of our thesis to identify key points on vehicles that can be tracked over time. It is possible to determine the motion and direction of the vehicles by tracking these points. Shi-Tomasi corner detection is particularly useful in this context because it is more accurate than other corner-based algorithms and can be used in conjunction with other algorithms to improve tracking performance, such as Optical Flow [26] [27].

2.1.6 Optical Flow

Optical Flow characterizes the apparent movement of objects and edges in a scene due to relative motion between the observer and the scene. It is used in conjunction with sequential images, such as video frames, to estimate point velocities and predict their positions in subsequent frames. It is widely used in robotics for tasks such as motion detection, object segmentation, and stereo disparity measurement because it represents brightness pattern velocities in an image. Using local Taylor series approximations of the image signal and partial derivatives in spatial and temporal coordinates, the methods calculate motion between two images at times t and t+dt [28] [29] [30]. OpenCV provides further insights into Optical Flow [29].

Consider a pixel *I(x,y,t)* in the first frame. It moves by a distance *(dx,dy)* in the next frame taken after dt time. So since those pixels are the same and intensity does not change, we can say:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.3}$$

Then we take the Taylor series approximation of the right-hand side, remove common terms, and divide by dt to get the following equation:

$$f_x u + f_y u + f_t = 0 \tag{2.4}$$

, where:

$$f_x = \frac{\delta f}{\delta x}; f_y = \frac{\delta f}{\delta y}; u = \frac{dx}{dt}; v = \frac{dy}{dt} \tag{2.5}$$

"The Optical Flow equation" is the name given to the above equation. Within this equation, we find f_x and f_y representing image gradients, while f_t represents the gradient across time. The variables (u, v) are, however, unknown. This equation cannot be solved directly because it contains two unknown variables. As a result, various methods for addressing this challenge have been developed, one of which is Lucas-Kanade, a specific variant of Sparse Optical Flow. In the following section, we'll go over Lucas-Kanade in greater depth.

There are two main types of Optical Flow, Dense Optical Flow and Sparse Optical Flow. Dense Optical Flow computes the Optical Flow vector for every pixel in a frame, ensuring that the entire frame is considered. This thorough approach can produce more

accurate results, but it often results in slower processing speeds due to its extensive nature. Conversely, sparse Optical Flow targets specific "interesting features" within an image, such as edges or corners of objects. Instead of analyzing the entire frame, it strategically chooses a limited set of pixels, focusing on those that highlight crucial features. The motion of these selected pixels, represented by their velocity vectors, is tracked consistently across frames. To maintain this consistent tracking, the extracted features are passed through the Optical Flow function from one frame to another. Given that these characteristics of sparse Optical Flow fall in line with the scope of our thesis and objective, we opted to combine it with the previously mentioned Shi-Tomasi corner detection, aiming for a more accurate and efficient yet lightweight, and simpler approach [29] [31].

There are various implementations of both sparse and dense Optical Flow, such as Lucas-Kanade, Farneback, Horn-Schunck, Block-matching (also known as the Lucas-Tomasi method), and many others. A widely used sparse Optical Flow method is the Lucas-Kanade method, which we will analyze more in-depth as it was the chosen method for this thesis.

The Farneback method is another widely used technique. This method uses a dense Optical Flow approach to estimate the movement of each pixel in an image. It employs a pyramid structure to gauge flow across various scales, allowing it to handle large motions and fast-moving objects. In comparison to Lucas-Kanade, the Farneback method is more resilient to noise; however, it has a higher computational cost [32]. The Farneback method, like the Lucas-Kanade method, has found applications in traffic safety and violation detection. However, due to the aforementioned considerations, we have decided against using it.

Finally, another important technique to introduce is template matching. Template matching is not an Optical Flow method, however, it has similar uses to Optical Flow, and like the previous methods, it has seen some use in a few approaches regarding traffic safety and traffic violation detection, hence why we thought it would be important to mention. This method compares a small image segment known as the "template" (as shown in Figure 2.6) to the original image at various positions. The algorithm identifies the image location where the template shares the closest resemblance in appearance (as shown in Figure 2.7), which can be applied to Optical Flow by systematically sliding the template across the image and evaluating its similarity at various positions, [33]. This technique has improved noise and occlusion resistance, however, when confronted with large displacements, it is less precise than both Lucas-Kanade and Farneback, making it a less desirable choice for our particular implementation.

As mentioned before, the Lucas-Kanade (LK) method, which is based on sparse Optical Flow techniques, is notable for its conceptual clarity and computational efficiency. The LK method is based on the assumption that the flow in a local neighborhood around a given pixel remains essentially constant. This means that it anticipates minor and consistent displacement of the image contents between two consecutive frames within

Figure 2.5: An example of Farneback Optical Flow, taken from [29]

Figure 2.6: An example of a template, taken from [34]

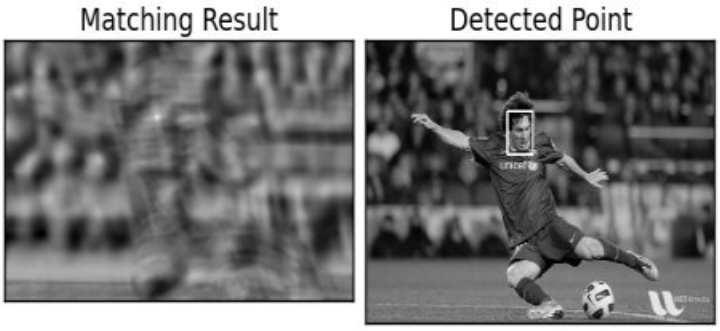

Figure 2.7: An example of Template Matching, taken from [34]

the vicinity of the pixel in focus. As a result, the basic Optical Flow equations are solved using the least squares criterion for each pixel in this neighborhood. This collective approach of collecting information from several nearby pixels, allows the LK method to frequently disentangle the inherent ambiguity associated with the Optical Flow equation, making it less susceptible to image noise than methods that assess points in isolation. When combined with a pyramidal framework, the LK method extends its reach, tracking even larger displacements [29]. This method's widespread acceptance, bolstered by its resilience and well-tested performance, gains additional precision when combined with Shi-Tomasi corner detection. They form an efficient combination of feature identification and meticulous tracking across frames, which is why we opted to use this approach. The following is an explanation of how the Lucas Kanade method operates, provided by OpenCV [29]:

The Lucas-Kanade method hinges on the principle that adjacent pixels move similarly. This method zeroes in on a 3x3 patch around a point, suggesting that the nine pixels

in this area move in unison. When observing these points, one can deduce the motion values *(fx, fy, ft)*. Consequently, we face the challenge of resolving nine equations with just two unknowns, resulting in an over-determined situation. To navigate this, the method employs the least squares fit approach. The resulting solution can be represented as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix} \tag{2.6}$$

From a user's perspective, the Lucas-Kanade method appears simple: input specific points and receive the associated Optical Flow vectors. Yet, there's an inherent challenge with this approach: it's primarily equipped to manage small motions between frames. Large motions can complicate matters. The solution to this problem is utilizing pyramids. As we progress upwards in the pyramid, minor motions get eliminated, and what were once large motions are scaled down to more manageable sizes. Leveraging the Lucas-Kanade method at this juncture allows for accurate determination of Optical Flow and its related scale.

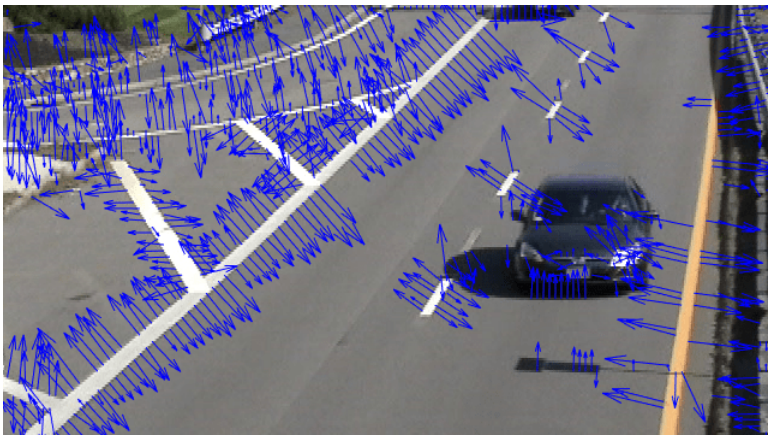

Figure 2.8: Plot of LK Optical Flow Vectors [35]

2.2 Red Light Cameras

Red Light Cameras are generally automated enforcement systems that use cameras to detect vehicles that run red lights. These systems are typically activated when a vehicle enters the intersection after the traffic signal has turned red. When a violation is detected, the system usually takes a photograph and/or a video of the vehicle and its license plate, so that it can be appropriately handled by law enforcement later [36].

Studies have shown that red light cameras can be effective in reducing the number of red light violations and the resulting accidents and injuries [5]. However, their effectiveness can vary depending on factors such as the location of the cameras, the duration of the yellow signal, and the presence of other traffic calming measures, among others.

Red Light Cameras have also been the subject of controversy and criticism, with some arguing that they are more focused on generating revenue than improving safety and

that they may disproportionately impact low-income drivers. There are also concerns about the accuracy of the systems, putting their fairness into question [37]. Our approach aims to answer some of these concerns, which we will expand on in later sections. There are two main types of red light cameras when it comes to their mobility, fixed red light cameras and mobile red light cameras.

2.2.1 Fixed Red Light Cameras

Fixed red light cameras are permanently installed at intersections to consistently monitor traffic adherence to signals. Positioned at known problem intersections, their main advantage is that they provide a consistent deterrent to red light violations, thereby improving safety. However, their stationary nature confines them to a single location, preventing them from adapting to emerging traffic concerns elsewhere [5].

2.2.2 Mobile Red Light Cameras

On the other hand, mobile red light cameras are equipped on vehicles, trailers, or other locations, allowing them to monitor various locations based on their need. In our opinion, their versatility lets authorities adapt to evolving traffic situations. Though their sporadic presence may discourage regular violators in different spots, the absence of constant surveillance might diminish the persistent impact seen with fixed cameras. Nonetheless, given the direction of our thesis, we chose to focus on this type of camera.

2.3 Chapter Conclusions

In conclusion, this chapter offered a thorough overview of the fundamental principles, key concepts, and fundamental techniques underlying the domains of red light detection and image processing. By looking into the various principles, algorithms, and techniques presented in this chapter, readers can gain a better understanding of their complexity, accuracy, and computational requirements, all of which are important factors in the development of our proposed solution for detecting traffic red light violations.

The following chapter, titled "State of the Art," will provide an in-depth review of the existing literature on traffic red light violation detection, with a special emphasis on image processing and detection solutions similar to ours. This examination of the current state of the art will provide useful insights into the strengths and weaknesses of current approaches, influencing the evolution of our solution.

3

State of the Art

In this chapter, we will discuss the current state of some of the technologies used in the project, as well as some techniques that we found to be especially important to delve deeper into.

3.1 Red Light Cameras

In this section, we will continue to delve into the use of Red Light Cameras as a method for detecting and deterring red light violations. We will start by providing an overview of the current state of red light camera technology. We will then delve into some implementations and existing research on the topic, where we will examine three papers that we found particularly relevant to our research.

3.1.1 Current State of the Technology

In this subsection, we will explore how and where red light cameras have been implemented, several different approaches and implementations as well as their effectiveness and feasibility.

Some examples of countries that use red light cameras include [38]:

- **United States:** Red Light Cameras are used in many parts of the United States, although their use has declined in recent years, with some cities and states banning their use or ending their programs.
- **Australia:** Red Light Cameras are used in many states and territories in Australia, including New South Wales, Victoria, and Queensland.
- **Canada:** Red Light Cameras are used in several provinces in Canada, including Ontario, Quebec, and British Columbia.
- **United Kingdom:** Red Light Cameras are used in many parts of the United Kingdom, including England, Scotland, and Wales.

- **Germany:** Red Light Cameras are used in some cities in Germany, including Berlin, Munich, and Frankfurt.
- **Singapore:** Red Light Cameras are widely used in Singapore as part of the country's efforts to improve road safety.

As evident from the before examples, Red Light Cameras remain a widely used tool for enforcing traffic laws in many countries around the world. In the United States specifically, the use of red light cameras has declined in recent years, with some cities and states banning their use or ending their programs, citing disproportionate ticketing of low-risk offenses and bad financial model [39]. However, red light cameras are still in use in many parts of the country and the world and continue to be deployed in new locations.

There has been significant research and development in the field of red light camera technology, both by private companies and unaffiliated individuals, as demonstrated by the numerous papers and studies published on the subject, some of which we will analyze in a further section. Several major companies are operating in this field, and these companies implement these systems in various countries around the world, including the United States, the United Kingdom, Australia, and Singapore, among others, as mentioned before [38].

Some examples of these companies include American Traffic Solutions (ATS), Conduent, Verra Mobility, Redflex Traffic Systems, Swarco, Siemens, and Nextec. These companies are usually contracted by cities to implement their technologies and receive a percentage of the ticket money [40]. There is limited information available on the technologies and systems used by companies in this field, as these companies may be hesitant to disclose their methods due to concerns about the potential loss of profit. Therefore, in this report, we will examine red light camera implementations and studies published by third parties, including those that utilize image processing and those that do not.

3.1.2 Implementations and Existing Research

In this section, we will review and discuss some previous research and published papers on the topic of red light violation detection systems. We will present and analyze a few relevant examples of systems that use image processing, similar to the approach we are proposing, as well as systems that utilize other techniques that were not within the scope of our research or not feasible for our specific objective, but that we believe are relevant and worth discussing.

In [41], **Saha et al.(2010)** use image processing and a background subtraction technique to detect moving objects in changing lighting and background conditions. They use this technique to identify and track moving objects by subtracting the background from the current image, allowing for the detection of intrusions and moving objects.

The way this "background" is obtained, is by using an adaptive technique to gradually modify the background image by starting with five initial images. If the difference

between any successive image and the background image is found to be below a predefined threshold value, the image is considered a background image and added to the background image database. Upon modification to the database, a new image is generated by averaging the last five images and used as the current background until the next calculation.

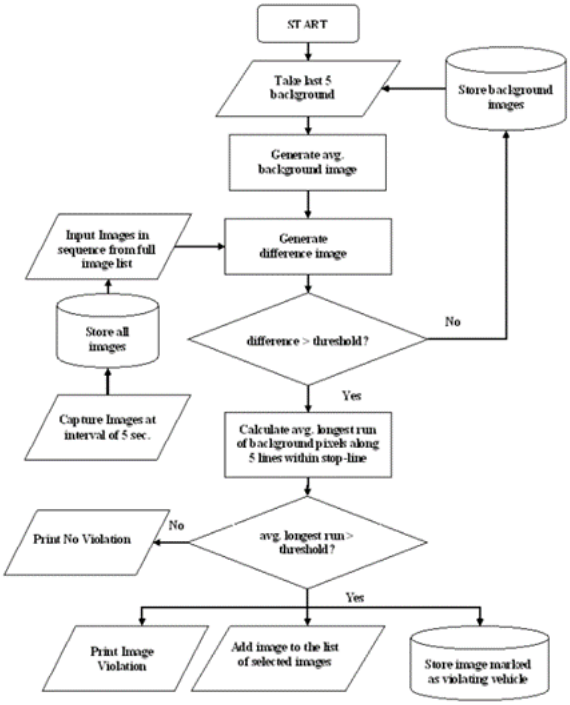

Figure 3.1: Scheme flowchart, taken from [41]

When an image is added, a difference image is created and analyzed by calculating the mean gray value of its pixels (Fig.3.2). If the mean value is above a predefined threshold, the difference image is considered a non-background image, potentially indicating the presence of an object. The white stop-line appears black in the difference image if it is not covered by a vehicle, allowing the system to determine occlusion by examining the pixels along the lines. If any non-black pixels are detected, occlusion is assumed.

To address issues like noise or pedestrians, the total number of continuous non-black pixels along the lines is considered instead of isolated non-black pixels. Additionally, the longest run of non-background pixels along the lines is also analyzed and their mean is calculated, taking into account that a few people may sporadically occlude the stop-line, but a vehicle will likely occlude it continuously over a longer distance. A Flowchart of the complete system is available in Fig.3.1.

This approach is beyond the scope of our research due to the use of three cameras and may not be as easily implemented as our proposed solution. It may also result in false positives in situations with many pedestrians and it is common for drivers to slightly encroach the stop line without violating the red light. Therefore, it is more appropriate to focus on analyzing the area in front of the stop line rather than the line itself.

Figure 3.2: Subtraction of the vehicle image from the mean background image, taken from [41]

In [42], **Polhan P et al.(2016)** an approach similar to the one being proposed in our paper is utilized. The authors employ image processing techniques, including image detection to detect when a red light is on and the creation of a "Forbidden Region" to identify vehicles that enter the region when the light is red. Optical Flow is then utilized to detect red light violations. This approach utilizes a single camera attached to a PC, which is positioned to monitor a controlled intersection. This approach is closely aligned with the scope of this research, as it is similar to the proposed solution.

An operator must place the camera in a stable position that allows it to capture a clear view of the intersection and at least one traffic light. The software then employs image detection to identify potential traffic light candidates within the field of view. It is then up to the operator to confirm the detection of the correct traffic light.

The traffic light detection algorithm goes through a few steps. First, it identifies "red" areas in the image by comparing the high intensities in the red channel with the intensities in the other two channels. However, because typical traffic scenes contain large white areas that also show high red intensities, these are eliminated from consideration, this being the reason the authors did not attempt to identify lights fully automatically. Instead, they select red light candidates by shape and size and present them to the operator one by one. The process is terminated when the operator confirms the detection of a candidate or requests another image.

Once a traffic light is detected and selected by the operator, the next step is to define the "forbidden region" - the area that should not be entered by vehicles after the light has turned red. The process of identifying this region is designed to be quick and easy for the operator, requiring only two mouse clicks to complete. This step is critical for the system to correctly identify red light violations, as it establishes the boundaries of the area where vehicles should not be present after the light turns red.

After the forbidden region is established, the operator then aligns the link between

the camera at the intersection and the monitoring computer. The authors suggest that to ensure smooth transmission, the antennae at both ends should be placed at a reasonable height above the ground. Even in busy city areas, they found that transmitters and receivers placed 2-3m or more above the ground were sufficient. The software then verifies that there is sufficient bandwidth to transmit images of red light violators in real time.

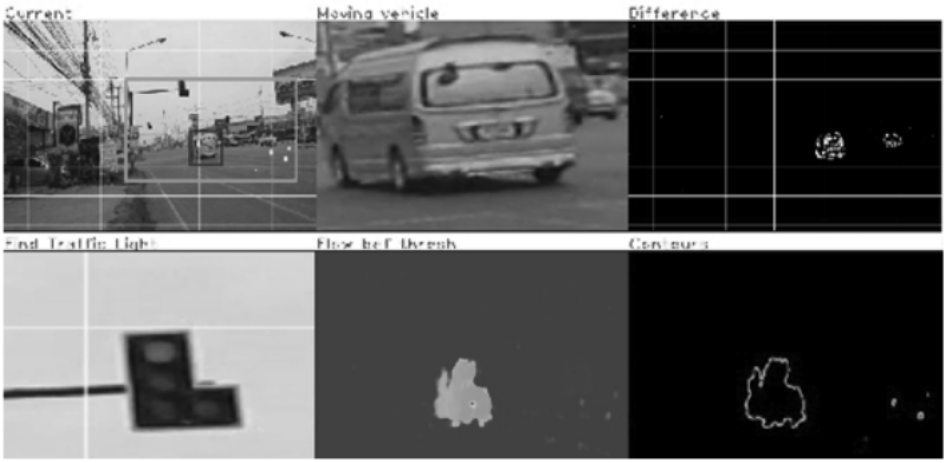

Figure 3.3: Screenshot after detecting a red light runner, taken from [42]

Next, the program enters monitor mode, where it uses Optical Flow to detect moving objects in the exclusion zone when the traffic light is red, automatically. Candidate algorithms for this mode were frame differences and Optical Flow, which ended up being used. The frame differences method, while relatively fast, is susceptible to interference from moving objects such as trees or even traffic lights affected by the wind. This requires significant processing to remove noise and non-vehicle movements, which can be challenging and time-consuming. Ultimately, the authors found that this method did not meet their real-time constraints and rejected it in favor of using Optical Flow techniques, which are more robust and reliable for identifying red light violations.

The use of Optical Flow is computationally complex, especially when applied to high-resolution images, so to mitigate this challenge, the authors make use of the forbidden region defined earlier to reduce the area that must be processed. Additionally, Optical Flow enables the determination of the speed and direction of moving vehicles, which allows for the rejection of vehicles that may have stopped unintentionally in the forbidden region. For this method, the authors employed the Farneback Optical Flow routine from the OpenCV library, which when applied to high-resolution images, may struggle to accurately assign a uniform velocity to every pixel of a moving vehicle. However, despite this limitation, the algorithm is still able to effectively identify the moving vehicle (Fig.3.3) and provide a clear image of the red light violator.

Our analysis of this paper was more in-depth as their approach closely aligns with the one we are proposing. The techniques and hardware used in their study fall within the scope of our research and the insights gained from this paper were valuable in

the development of our solution. The method described in the paper is simple yet effective, utilizing a combination of image processing techniques such as, but not limited to Optical Flow. Additionally, the hardware requirements are minimal compared to other implementations, requiring only one camera and a computer, with the assistance of an operator to ensure smooth operation.

The third and final paper we will examine in-depth is [43] by **Thao et al.(2017)**, which employs a distinct approach compared to the previous two studies. Although it falls within the scope of our research, we believe that our approach is more appropriate for the problem at hand in our specific situation. Nevertheless, the findings from this paper were still relevant to our investigation.

In this study, **Thao et al.** use image processing to detect when a red light is on, similar to many other papers in this field, however, they take a unique approach by utilizing machine learning, specifically convolutional neural networks (CNNs), to address the problem of red light violation detection.

The system uses a camera to capture traffic at intersections with traffic lights and pre-processes the images to detect the stop line and the location of the traffic lights. Image processing techniques such as filtering, calculating the area of uniform areas, and contour drawing are used to detect the road markings and determine the location and changing state of the traffic lights. These parameters are then used to decide whether a vehicle has committed a traffic violation. A system configuration, as well as a flowchart of its workings, can be seen in Fig.3.4 and Fig.3.5 respectively.

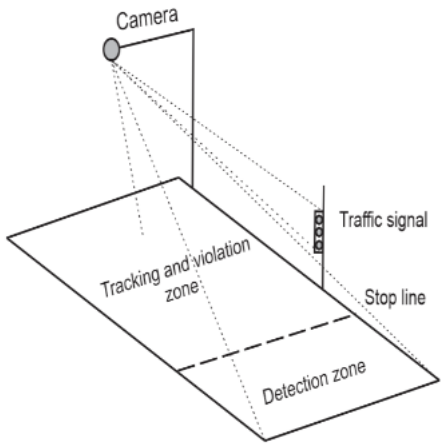

Figure 3.4: System configuration, taken from [43]

The identification of violations is performed on a server, which uses the stop line on the road and the traffic signal color to generate different zones. If a vehicle is inside the violating zone while the traffic light is yellow or red, it is detected as violating. The system employs deep learning techniques with pre-trained models to recognize the vehicle that the camera captured, and in the project, the authors used the modified YOLOv5s pre-trained model for the detection of violating vehicles.

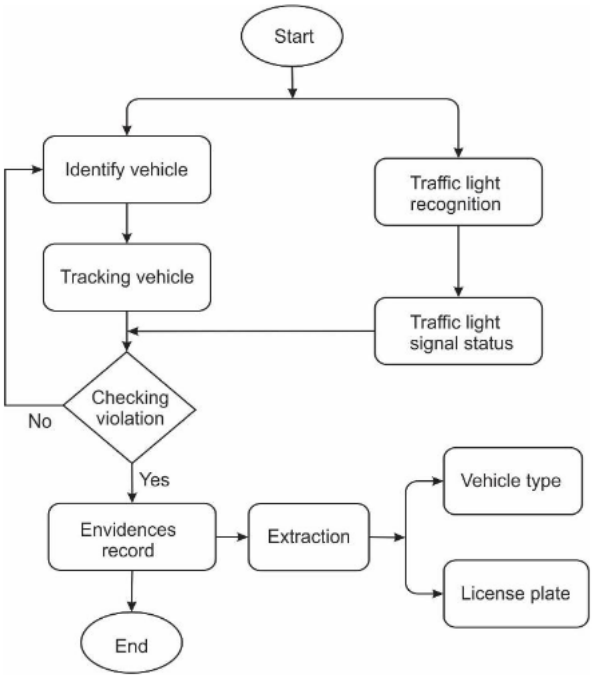

Figure 3.5: Algorithm of violation detection, taken from [43]

Once a vehicle violates, the following frames will be used to identify the vehicle's license plate, and the image is given to the camera as input, where it is further processed to compare with traffic signal change and check for violation. To detect a vehicle license plate, the authors apply machine learning to border the plate combined with the optical character recognition technique, which is used to recognize the extracted characters to get the violated vehicle's information from the plate that is bordered.

As stated before, this paper provides valuable insights into the use of machine learning and convolutional neural networks for red light violation detection. While our approach may differ, this solution was still within our scope, and the findings from this paper were still important to our investigation as they provide a different perspective on the problem and offer potential solutions that we can learn from.

The three papers we analyzed in-depth, [41], [42], and [43], offer different and valuable approaches to the problem of red light violation detection, and although we have read and analyzed many more papers, these three were selected for their unique and relevant perspectives on the problem.

In addition to the papers we analyzed in-depth, several other approaches and variations have been proposed in the literature. For example, in [44], a similar approach to the first paper we analyzed in this section is used, with a heavy focus on background subtraction and extraction methods. However, this paper also introduces the use of blob analysis to differentiate between pedestrians and vehicles.

Another interesting approach is presented in [45], where the camera and computer are embedded in the vehicle, rather than on a pole or traffic light. While this approach

is outside of the scope of our research and difficult to implement on a larger scale, it is still noteworthy. Additionally, [46] proposes an extremely complex system that not only detects red light violations but also speed limit infractions. This system makes use of a camera embedded in the car and GPS tracking to identify the offender.

In contrast, [47] utilizes motion detection sensors, and image processing is only used to extract the car plate. This approach is different from [48], which calculates the distance between the post and the vehicle in real time to detect when the vehicle breaks a certain threshold.

As stated before, while these papers propose different approaches to detecting red light violations, they all demonstrate the importance of keeping an open mind and considering various methods when tackling this problem. By considering a variety of different approaches, we gained insights into the challenges and limitations that may arise when doing a project like this one and used this knowledge to improve our solution, such as using background subtraction, which we had not considered at the beginning.

3.2 Image processing and detection

In this section, we will continue exploring the concept of image processing as well as some image detection techniques, and their current state in the industry. We will mention some techniques and methods used to analyze and extract information from images, and how they were applied to the problem of detecting red light violations in other papers. Additionally, we will discuss its current state in the industry, some challenges faced, and future prospects.

3.2.1 Current State of the Technology

Image processing and image detection technologies play pivotal roles across various industries, including healthcare, manufacturing, security, and notably, traffic safety. Their significance has been driven by advancements in algorithms, machine learning, and deep learning methodologies. One clear testimony to their influence is the application of image processing in red light cameras, a technology leveraged to enhance road safety and efficiency [49]. Within this scope, image processing efforts revolve around the creation of novel algorithms and techniques geared toward object detection, classification, and tracking. Furthermore, there's a rising interest in employing image processing for innovative applications like intelligent transportation systems [50] and connected, automated vehicles, marking this as a swiftly evolving technological realm.

Parallelly, image and object detection have undergone significant refinement over recent years. The incorporation of convolutional neural networks (CNNs), for instance, has empowered the field with superior pattern recognition capabilities in images and objects [51]. This enhancement has found applications in diverse sectors, encompassing security, surveillance, autonomous vehicles, and traffic safety. Specifically, within traffic

safety, image detection augments the precision in identifying red light violations [5]. However, as its adoption becomes ubiquitous, certain challenges arise. Concerns related to the continuous monitoring of citizens, intruding on their privacy and civil liberties, have been voiced [52]. Additionally, while the technology bolsters accuracy, it isn't without flaws. Instances of false detections have occasionally led to unwarranted ticketing and fines [37], highlighting areas demanding further refinement.

3.2.2 Implementations and Existing Research

In this section, we will examine and evaluate prior research and published papers on the topic of image processing and image/object detection, with a specific focus on traffic light detection. As image/object detection is a broader topic than red light cameras alone, we will not delve as deeply as in the previous section. Nevertheless, we will present and analyze a range of methods related to this problem, whether they are/were directly relevant to our research or not, but deemed to be noteworthy and worthy of discussion. As previously stated, the insights gained from each paper we analyzed were valuable in helping us achieve our ultimate objective.

In **Polhan et al. (2016)** [42], an operator-assisted approach is presented, where the operator is required to position the camera in a stable location to capture the intersection and at least one traffic light. The software then runs a script to detect multiple traffic light candidates and prompts the operator to select the desired one. While this approach requires human intervention during setup, it increases the accuracy of the system by eliminating the possibility of the program detecting the wrong traffic light or object.

The specific algorithm or technique used in the traffic light detection process is not detailed in the paper, however, it is stated that it involves identifying "red" areas by comparing high intensities in the red channel of the image with intensities in the other two channels. The process also eliminates white areas that may show high red intensities and eliminates red-colored cars, which are often present in the scene, making this the main reason why this part of the program is so difficult to fully automate while maintaining a high level of accuracy. Lastly, after comparing the intensity of the red channel with the others, the program selects red light candidates by shape and size, presents these candidates to the operator one-by-one, and terminates this task when the operator accepts a candidate or repeats the process on another image.

While there is not a lot of detail provided on the techniques used to detect traffic lights, the authors highlight the importance of having an operator aid in the process of traffic light detection. Additionally, this approach aligns with our research, as we also incorporated human input in a few portions of the setup of our system. The idea of having a human aid in the set-up and selection of traffic light candidates is appealing to us as it can eliminate potential errors that may be introduced by relying solely on an automated process while increasing its accuracy.

In [53], **Wonghabut et al. (2018)** delve deeper into the methods and algorithms used to detect traffic lights, as it is the primary focus of their research. The goal of their paper is to address the challenges of detecting red lights in video frames due to fading or dimming of the light, obscuration by large vehicles, and flare. To address these issues, they propose a software technique based on the HSV (Hue, Saturation, Value) color model that can eliminate the difficulties in red light detection and identify all colors of a traffic light.

The authors developed two distinct techniques to enhance the accuracy of the traffic light detection system, however, these methods were found to be more effective when used together than when used independently, so, the final method is a combination of the two.

In the first approach presented in the paper, the video frames are converted to grayscale images, and then, a threshold is applied to yield binary images. The three colors of traffic lights are then compared to predefined patterns to determine which one is on. If more than one color of the traffic light is found to be on, the threshold value is adjusted and the process is repeated.

The second approach utilizes the stability of color in the HSV model by converting video frames in RGB to the HSV color space (Fig.3.6) and comparing the hue value to determine the color of the traffic light. This method takes advantage of the fact that in the HSV model, the color remains consistent despite changes in lighting conditions, with only the brightness value changing.

$$H \begin{cases} 60 \times \dfrac{G-B}{max-min} & \text{if } max = \boldsymbol{R} \\ 60 \times \dfrac{B-R}{max-min} + 120 & \text{if } max = \boldsymbol{G} \\ 60 \times \dfrac{R-G}{max-min} + 240 & \text{if } max = \boldsymbol{B} \end{cases}$$

$$S = \frac{max-min}{max} \times 100$$

$$V = max \times 100$$

Figure 3.6: Converting RGB to HSV, taken from [53]

The third approach combines the previous two methods to further improve the accuracy of traffic light detection. The process begins by converting the video frames from RGB to HSV and comparing the hue values to identify the color of the traffic light (method 2). The image is then converted to grayscale, and a threshold is applied to yield a binary image. Next, the three traffic light colors are compared to a predefined pattern to determine which one is on (method 1). A flowchart of these approaches can be seen in Fig.3.7.

It is important to mention that similar to [42], every method used in this paper relies on an ROI (Region of Interest) to more accurately identify the traffic light. Additionally, each method presented in the paper is followed by a conditional analysis to handle ambiguity in the detection. This is accomplished through the use of a predefined pattern, as illustrated in Fig.3.8, to determine the output (green, yellow, or red) of the traffic light, which can then

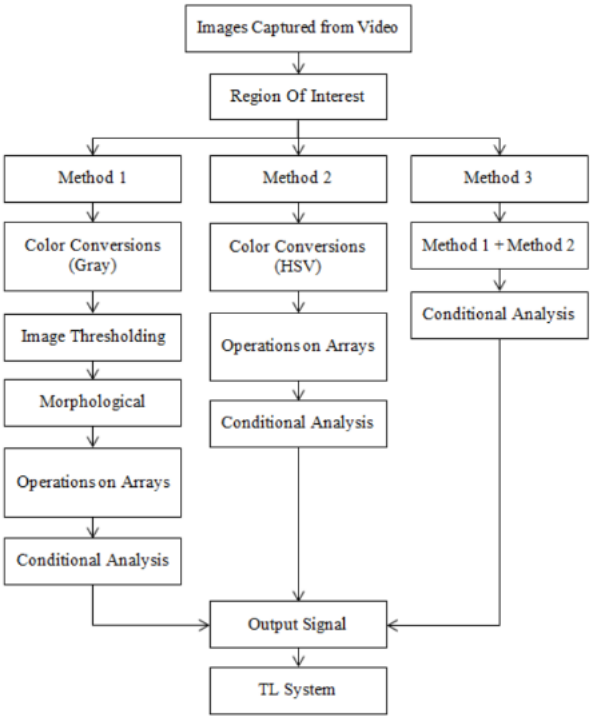

Figure 3.7: Flowchart of the proposed methods, taken from [53]

be used in our specific case for traffic light violation detection by triggering an automated detection routine when the output is "red".

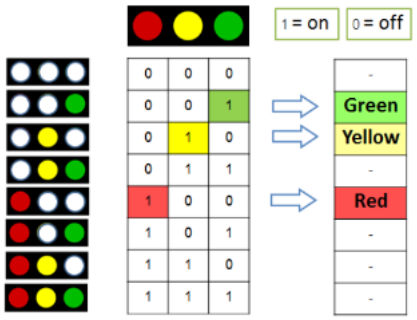

Figure 3.8: Decision pattern, taken from [53]

This approach, as presented in the paper, appears to be effective and has high accuracy rates in detecting traffic light colors. The authors claim up to 100% accuracy when detecting just red and green, and 96% accuracy when identifying all three colors. This approach was used as inspiration for some techniques we implemented in our final implementation.

The third and final paper we will discuss in this section is [54]. In this paper, **Omachi M et al.(2009)** use a very different approach from the two previously mentioned studies. A unique approach is proposed, to detect traffic lights using a single in-vehicle camera. They convert the color space from RGB to normalized RGB and then detect candidate regions for traffic lights by identifying pixels that match the colors of said lights (green, yellow,

and red). They then detect edges from the remaining pixels, and finally, use the Hough transform method to detect circles representing traffic lights. A complete flowchart of this approach can be seen in Fig.3.9.

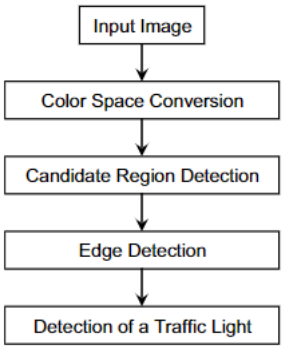

Figure 3.9: Flowchart of the approach, taken from [54]

According to the authors, a lot of techniques were proposed for detecting general objects from a "scene image". One of the most fundamental techniques is template matching, in which a region that is most similar to a given template is selected from the input image, using a criterion such as the normalized cross-correlation or the sum of squared differences. In this case, to achieve a stable detection, it was proposed to convert the color space from RGB to normalized RGB, as it has been reported to be robust under illumination changes. The conversion to normalized RGB is performed by passing the original red, green, and blue values through a function (Fig.3.10), which outputs the normalized values of R, G, and B, as seen in Fig.3.11.

$$\begin{cases} R = G = B = 0 & (\text{if } s = 0) \\ R = \frac{r}{s},\ G = \frac{g}{s},\ B = \frac{b}{s} & (\text{otherwise}) \end{cases},$$

where

$$s = r + g + b\,.$$

Figure 3.10: RGB to normalized RGB, taken from [54]

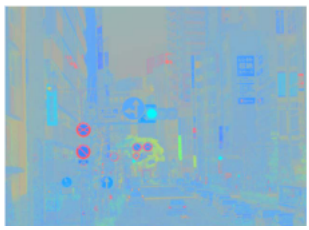

Figure 3.11: Normalized RGB, taken from [54]

The extraction of candidate regions for a traffic light is done by analyzing the normalized RGB image. Pixels are evaluated based on their R, G, and B values, and those that meet certain conditions are considered candidates. These conditions are determined by observing the images of traffic lights and are defined as:

(R > 200 and G < 150 and B < 150)
or (R > 200 and G > 150 and B < 150)
or (R < 150 and G > 240 and B > 220).
The connected regions that are formed by these selected pixels are considered potential traffic light candidates, as seen in Fig.3.12.

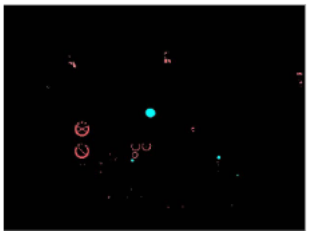

Figure 3.12: Candidate regions, taken from [54]

The process of detecting edges in the candidate regions is accomplished by applying a Sobel filter to the image. This filter is applied only to the candidate regions that have been previously extracted, enabling the detection of edges within these specific areas. This approach is relatively straightforward, as it only focuses on the regions that have been identified as potential traffic lights, as seen in Fig.3.13.

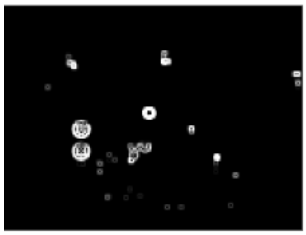

Figure 3.13: Edges, taken from [54]

To detect the traffic light, the authors use the Hough transform to detect a filled circle that represents a traffic light. Using the previously calculated edge of the image, the program then uses a simple equation of a circle to detect circular edges. Each pixel on a detected edge is then used to vote in a three-dimensional parameter space. The set of parameters that receives the most votes is considered to be the position and size of the traffic light. To improve the accuracy of the detection and eliminate the detection of open circles, the previously extracted candidate regions are utilized. By utilizing these regions, the detection process can focus on areas of the image that are more likely to contain a traffic light, resulting in a more efficient and accurate detection. An example of the detection of a filled circle can be seen in Fig.3.14, as well as an example of a detected traffic light in Fig.3.15.

According to the authors, the proposed method in this paper demonstrates the ability to detect a traffic light promptly with improved accuracy compared to traditional methods. They suggest that future research could focus on enhancing the accuracy by considering the location of the detected area within the image. While the approach is intriguing, it is

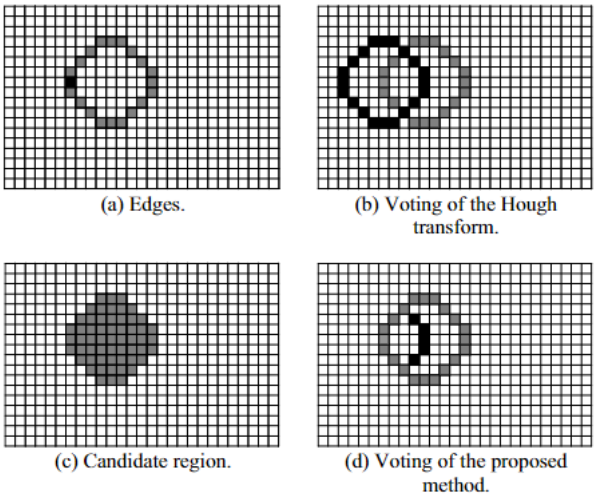

Figure 3.14: Detection of a filled circle, taken from [54]

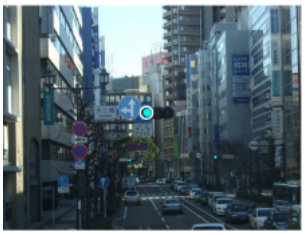

Figure 3.15: Detected traffic light, taken from [54]

somewhat outside the scope of our research and lacks the level of experimentation and results found in the previous paper.

The three papers we analyzed, [42], [53], and [54], offer different and valuable image processing/detection techniques aimed at the problem of traffic-light detection, and although we have read and analyzed more papers, these three were selected for their unique and relevant perspectives on the problem. In this section, we did not delve into the idea of vehicle detection on its own as we want to analyze it separately, more specifically when using Optical Flow and background subtraction.

3.3 K-Means

In this section, we'll review the concept of K-means clustering and how it applies to our implementation. We will highlight approaches and methodologies that use K-means clustering in various ways. We will look at its current position in the industry, its challenges, and its potential future advancements. Although there are some references to papers where K-means has been successfully applied in different contexts, its application in traffic safety, particularly in red light violation detection, remains limited. Therefore, we will not delve deeply into any specific papers on this topic.

3.3.1 Current State of the Technology

K-means clustering, which originated in signal processing, has cemented its position as a critical method in unsupervised learning and vector quantization and is widely used in data science and machine learning disciplines [55]. Its primary utility is in partitioning data into distinct clusters based on inherent similarities, making it a popular clustering algorithm. Modern improvements, such as K-means++, have addressed initial centroid selection sensitivities, while methodologies such as the Elbow Method and Silhouette Analysis help determine the optimal cluster count [55]. Other notable algorithms, such as Affinity Propagation, have begun to shape the field when discussing the broader spectrum of clustering techniques, offering alternatives to the well-established K-means[56].

K-means clustering has demonstrated significant proficiency in the specialized realm of image processing, particularly in tasks such as image compression and segmentation [55]. In our implementation, we came up with a creative use of K-means clustering. It is synergized with other image processing techniques, where we use it to divide brightness values into two distinct clusters, allowing us to determine whether or not traffic lights are activated. This methodology not only provides a lightweight implementation but also ensures adaptability and robustness, allowing it to navigate the complexities of changing light conditions.

The combination of K-means clustering and traffic safety is still largely unexplored territory. Modern intersections with high-resolution data systems have the potential to integrate K-means clustering to revolutionize real-time traffic management and signal control [57]. However, it is critical to emphasize the current lack of extensive research on the application of K-means in red light violation detection. As the field develops, there is a clear need for more in-depth research to uncover this potentially transformative synergy.

K-means clustering is an unsupervised learning cornerstone with a wide range of applications, from tailored image processing solutions to potential traffic safety innovations. As more nuanced challenges in data clustering and image processing emerge, the evolution and adaptation of K-means and its contemporaries will be critical.

3.4 Background Subtraction

In this section, we will delve deeper into the Background Subtraction technique and its synergistic potential when combined with Optical Flow, particularly for detecting vehicular movement and infractions. We will begin by providing an overview of Background Subtraction's current technological landscape. Following that, we will look at research papers that have explored innovative solutions for traffic monitoring and safety with the use of Background Subtraction.

3.4.1 Current State of the Technology

Background subtraction has firmly established itself as a critical domain in image processing, particularly in applications such as real-time surveillance, object detection, and tracking. Its methodologies have undergone significant transformations over the years, both conventional and cutting-edge, in pursuit of a generalized framework suitable for real-time applications. Deep neural network integration has heralded a renaissance in the field in recent years, with these networks providing unparalleled performance enhancements [58]. Deep learning has increased the efficacy of these methods by addressing the fundamental challenges inherent in background subtraction. Furthermore, the combination of various features has been identified as a means of improving traditional background subtraction techniques. To aid in the evaluation and progression of these algorithms, several benchmark video datasets have been introduced [58].

Within the context of these advancements, MOG2 (Mixture of Gaussian 2) stands out as a well-known method. MOG2's strength lies in its ability to distinguish between moving objects and shadows, making it particularly suited for static camera applications like ours. It is based on the idea of using a combination of Gaussian distributions to model each background pixel [59]. It is critical to emphasize that the effectiveness of background subtraction, particularly MOG2, serves as the foundation for subsequent higher-level video analytical tasks, emphasizing its importance in surveillance systems and human identification [59].

Background subtraction is used extensively in the field of traffic safety. Given the dynamic nature of traffic scenes, where both the environment and the objects are constantly changing, robust and accurate detection is critical [60]. The primary goal here is to track traffic patterns, detect anomalies, and, most importantly, ensure road safety [60]. Advanced background subtraction techniques are particularly useful in red light violation detection, a subset of traffic safety. These systems, by identifying vehicles that violate traffic rules, serve as critical tools in ensuring adherence to traffic regulations, ultimately contributing to safer roads.

Nevertheless, as with any evolving domain, background subtraction, including MOG2, has its challenges. While significant progress has been made, the dynamic nature of outdoor environments, particularly those densely populated by humans, introduces complexities that require further research [59]. With the promise that deep learning brings to the forefront and ongoing improvements in algorithms such as MOG2, the trajectory for background subtraction in image processing as well as its integration into critical sectors like traffic safety remains positive.

3.4.2 Implementations and Existing Research

In this section, we will delve into the field of Background Subtraction, with a specific focus on its usage in traffic safety and violation detection. We will be analyzing x papers that have used background subtraction in their methodologies. Although we utilize the

MOG2 background subtraction method in our implementation, our discussion will not be limited to this. We will also focus on other techniques mentioned in these papers, as they have provided useful insights. By examining these methods, we gained an understanding of their benefits and drawbacks, which was important when considering the possibility of their application.

In [41] by **Saha et al. (2010)**, the authors delve deeply into the mechanism of background subtraction as a foundation for detecting vehicles that violate the red light signal. Their methodology is insightful and suggests a combination of careful observation and creative application.

The primary goal of background subtraction in this implementation, as stated, is to distinguish between immutable elements in the scene (such as roads or buildings) and mutable entities (such as vehicles). This distinction is critical because it serves as the foundation for detecting changes, particularly moving vehicles, in an otherwise static frame. The paper effectively recognizes the dynamic nature of background changes caused by lighting changes resulting from natural sunlight shifts and artificial streetlights. As explained in a prior section, the authors address the issue of lighting variations by using an adaptive technique to update the background image regularly. The rolling update method, which starts with five initial images and refines them based on the differences between subsequent images, ensures that the system is always tuned in to the most recent environmental conditions. This aspect of their methodology is especially noteworthy because it takes into account real-world unpredictability.

They refine their methodology beyond vehicle detection to detect the precise position of the vehicles, specifically if they have crossed the delineated stop-line. This method is creative because it does not rely on only detecting the vehicle itself, but rather the occlusion of the stop-line. The logic is straightforward but effective: if the line is obscured, it is most likely due to a vehicle crossing it, indicating a potential violation. This approach is efficient and reduces computational complexity, as it doesn't require intricate vehicle detection algorithms.

While the approach presented in this paper is comprehensive, it raises concerns about its robustness in situations involving multiple vehicles or when confronted with an obscured stop-line caused by wear or environmental factors. Furthermore, as previously stated, this approach may be beyond our primary focus due to the possibility of false positives, particularly in situations with a high volume of pedestrians. It's not uncommon for drivers to go over the stop line but not break the red light sign. Thus, focusing on the area preceding the stop line may yield more accurate results than emphasizing the line itself.

In the second paper presented by **Prasantha et al. (2008)** [48], the technique of "Background Subtraction" or "Background Removal" is heavily emphasized. This method, which is firmly rooted in the realms of vehicle detection and red light violation, draws inspiration from the practicality of using an empty road as a reference. The method seeks

to identify vehicular presence through subtraction by comparing a static image of an empty road with a dynamic image, potentially full of vehicles. However, there is a flaw to this approach: the method's sensitivity to changes in lighting and environmental variations. Such inherent variations may skew the background, resulting in false detections, and highlighting the approach's fragility.

Because of this flaw, an alternative approach is proposed to strengthen their approach: the "inter-frame difference method." The strength of this method comes from comparing successive frames, essentially subtracting a current frame from the one before it. As a result, the regions that have displayed changes - essentially the moving objects - are highlighted. The success of this technique lies in its ability to avoid the pitfalls of environmental changes that affect the background subtraction method. While both approaches have distinct advantages and have a legitimate place in the overall framework, the paper favors the inter-frame differencing method due to its resilience and adaptability.

As previously stated, this paper provides important insights when comparing various methodologies for detecting red light violations. The emphasis on Background Subtraction highlighted potential issues, particularly its sensitivity to changes in environmental conditions. On the other hand, the alternative inter-frame difference method provides a response to the identified flaws, demonstrating its adaptability in the face of changing scenarios. While the advantages and disadvantages of both approaches are well articulated in this paper, guiding our understanding, we eventually leaned toward background subtraction, believing it aligns more effectively with our specific objectives, based on our research.

While these papers demonstrate various techniques for vehicle detection, they both emphasize the importance of being open to new ideas and evaluating multiple approaches when addressing this issue. We recognized the complexities and potential difficulties associated with such projects by examining a range of strategies. This new knowledge influenced the development of our solution, ultimately leading us to use the background subtraction method, as mentioned before.

3.5 Optical Flow

In this section, we will continue exploring the concept of Optical Flow, especially as a method for recognizing when a vehicle has infringed a red light. We will begin by providing an overview of the current state of the technology in the industry. We will then delve into two papers that have proposed different approaches to the problem of traffic safety, using Optical Flow.

3.5.1 Current State of the Technology

In the current landscape, Optical Flow techniques are widely used in a variety of applications such as video compression, object tracking, and image stabilization [61]. More

specifically, in the field of traffic safety and violation detection, Optical Flow methods have several ways of being implemented, for example, Lucas-Kanade and Farneback have been used to track vehicles and pedestrians in traffic scenes, while template matching has been used to detect traffic lights [42] [62]. These methods have been shown to be effective in many cases, but they also have their limitations.

One of the main challenges with using Optical Flow for traffic safety and violation detection is the high computational cost. The Lucas-Kanade and Farneback methods, for example, both rely on solving complex differential equations, which can be computationally expensive. Additionally, these methods can be sensitive to noise, which can lead to inaccurate results. Another challenge is the presence of occlusions, where in real-world traffic scenes, vehicles and pedestrians can occlude each other, which can make it difficult to accurately track their movements. Additionally, the presence of glare and reflections on vehicles can also make it difficult to detect traffic lights. Many of these concerns have been mentioned in previous papers such as [41], [42], and [53]. Another social limitation that techniques and technologies such as Optical Flow suffer are issues regarding privacy and civil liberties of individuals, as cameras are being used to monitor people and vehicles [52].

Despite these challenges, Optical Flow methods continue to be widely used in the field of traffic safety and violation detection, due to their effectiveness in many cases. However, it is important to keep in mind the limitations of these methods and to always consider the specific requirements of the problem at hand when choosing which method to use.

3.5.2 Implementations and Existing Research

In this section, we will explore several papers on the topic of Optical Flow, with a specific focus on its usage in traffic safety and violation detection. We will be examining two papers in particular and evaluating their methods and results. At this time, we have decided to use the Lucas-Kanade method in our implementation, however, we will still focus and analyze methods like Farneback and template matching as they have been widely used and have shown promising results in previous studies. Through these evaluations, we gained a deeper understanding of the strengths and limitations of these methods, as well as potential areas for further research.

Firstly, and as presented before, [42] **Polhan P et al.(2016)** use Optical Flow to detect traffic violations that occur within a designated "forbidden area". By implementing this forbidden area, they can address one of the main challenges associated with Optical Flow, which is its computational complexity. To perform this task, they utilize the Farneback method, which is known for its ability to handle large motions and fast-moving objects effectively. Additionally, the Farneback method is found to be more resilient to noise compared to the Lucas-Kanade method, which was one of the factors that led to its selection for use in the study.

Pixels that are moving faster are encoded with greenish colors, and a contour is drawn

around these fast-moving pixels. Areas that are too small to be a vehicle are rejected as noise, while larger areas are identified as potential red light runners. By extracting an image of the offending vehicle from the current scene image, it can be transmitted to a monitoring post. This method accurately identifies a moving vehicle, even when the Optical Flow algorithm struggles to assign a uniform velocity to every pixel of the vehicle, which happens especially often on images of higher quality, allowing for a clear and complete image of the red light runner.

These results and decisions present an interesting and seemingly effective way to use the Farneback Optical Flow method to help solve the problem.

In [62], **Parashar N et al.(2015)** use Optical Flow, more specifically the Lucas-Kanade method as an important step in their solution, but in a different way from the previously mentioned paper, since their main objective is to detect and control the flow of traffic as opposed to detecting red light runners.

According to the authors, the purpose of using Optical Flow in their system is to provide statistical traffic flow information. The role of Optical Flow in their system is deemed essential and critical to the final performance, therefore, it is important to consider application-specific assumptions when choosing an algorithm for Optical Flow estimation from the various available options. Optical Flow calculates the motion between two image frames taken at different times at every position. These methods are referred to as differential, as they are based on the Taylor Series' approximation of the image signal, utilizing partial derivatives in relation to both spatial and temporal coordinates.

According to the authors, the decision was made to use the Lucas-Kanade method for solving the problem at hand, as opposed to other commonly used methods such as the Horn-Schunck method. The application of Optical Flow in our approach is distinct from the problem being addressed by the authors of this paper, however, it presents a noteworthy usage of the Lucas-Kanade method in the field of traffic safety, providing valuable insight for our research.

As apparent from our research and the papers mentioned, Optical Flow has been used successfully in traffic safety and detection, hence why we used this technique in our implementation.

3.6 Chapter Conclusions

This chapter conducts an in-depth review of the current landscape of the technology and concepts used for the detection and prevention of red light violations. This research included a thorough review of research papers and practical implementations that all use image processing and related methodologies to recognize moving objects and pinpoint instances of red light violations. Furthermore, when addressing this complex issue, the chapter emphasized the importance of maintaining an adaptable mindset and embracing diverse strategies. By delving into a wide range of approaches, invaluable insights into

the nuanced challenges and constraints that can arise during the course of such a project have been gained. Our proposed solution and approach were improved as a result of this in-depth research.

In the next chapter, titled "Practical Approach," we will look at the methodology used in this study to create the envisioned traffic red light violation detection solution. A detailed breakdown of the coding and techniques used in the solution's implementation will be provided, along with a brief analysis of the obstacles and constraints encountered along the way. This chapter is intended to provide a practical understanding of our proposed solution and its suitability for real-world deployment scenarios.

4

Practical Approach

In this chapter, we delve into the complexities of our approach and the implementation designed to detect red light violations. In keeping with our original goal, we worked towards creating a more straightforward, lightweight, mobile, and simple-to-install system that tried not to sacrifice accuracy or robustness. Recognizing the limitations imposed by time constraints, we are currently in the prototype stage of our work. Our solution is made up of two main components that work together to create a complete implementation, one to detect the active traffic light color and one to detect vehicle movement in a particular direction. As a result, this chapter is divided into "Shared Functions and Imports", "Traffic Light Detection", "Vehicle Detection and Capture", and "UI and other Visual Components".

4.1 Shared Functions and Imports

This section delves into the imported libraries as well as the implementation of important functions that are used for both the "Traffic Light Detection" portion as well as the "Vehicle Detection and Capture" portion. These functions serve as a starting point for both the main functionalities of the code.

1. **Import Libraries:** The script begins by importing essential Python libraries, including OpenCV (`cv2`), NumPy (`np`), and the typing module for type annotations. Libraries such as Collections - Seque and SciPy - KDTree are also imported and used for the "Vehicle Detection and Capture" portion, which we will delve into later.

2. **Mouse Click Event Handler:** The `unified_click_event` function is set up to manage mouse clicks within the OpenCV window. This function updates global lists with ROI points for each different traffic light color, vehicle detection area, and vehicle direction, and visually guides the user by drawing lines and displaying informative messages on the frame.

3. **ROI Selection:** The `unified_select_roi` function handles the ROI selection process, calling `unified_click_event` and awaiting user input to establish all the ROIs. This function returns the drawn polygon as well as the coordinates of all its points.

4. **Video Initialization:** The script opens the specified video file, displays its first frame, and prompts the user to draw the ROIs for each corresponding traffic light color as well as the vehicle detection area and direction.

4.2 Traffic Light Detection

This section delves into the implementation of the Traffic Light Detection portion, clarifies the primary components of our chosen solution, and investigates our reasoning for choosing specific techniques over alternative approaches.

First, let's clarify the main objective of this first half of our solution prototype. The primary objective of this part of the implementation is to detect and identify the status of traffic lights (Red, Green, Yellow) in a given video file. The program uses OpenCV and Python to accomplish this.

Now let's take a more extended look at its main components:

1. **Global Variables:** Global variables are initialized to store the Region of Interest (ROI) coordinates for Red, Green, and Yellow traffic lights as empty lists. Additionally, a *path* variable is declared, specifying the input video file's path.

2. **Reading and Processing Frames:** One by one, the script isolates and stores the frames containing each traffic light color based on the previously defined ROIs. These cropped frames are then categorized into separate lists according to color. This is done separately from the main loop as we need to analyze at least one traffic light color cycle to obtain the brightness information, which will be used in combination with K-Means Clustering to determine the traffic light colors.

```
while True:
    # Read a single frame and store the return value in 'ret'
    ret, frame = cap.read()

    # Check if a valid frame is read
    if ret:
        # Crop the areas corresponding to each color ROI from the read
    frame
        red_frame = np.array([l[rxMin:rxMax] for l in frame[ryMin:ryMax
    ]])
        ...
    else:
        # Break the loop if no more frames are available
        break

    # Append the cropped areas to the respective lists for future
    analysis
    red_frames.append(red_frame)
    ...
```

Listing 4.1: Reading and Processing Frames Loop

3. **Brightness Calculation:** The `get_brightness` function calculates the mean brightness of a given frame, obtained in the previous process. It converts the color space of the frame to HSV as it is easier to manipulate the brightness value this way.

```python
def get_brightness(frame: np.ndarray) -> float:
    try:
        # Convert the color space of the frame to HSV
        hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)
        # Extract the V channel (Brightness)
        v_channel = hsv[:,:,2]
        # Compute the mean brightness of the V channel
        brightness = cv.mean(v_channel)[0]
        return brightness
    except Exception as e:
        print(f"An error occurred in get_brightness: {e}")
        return 0.0  # Return a default value
```

Listing 4.2: Brightness Calculation Function

4. **Extracting Brightness Values:** The `extract_brightness_values` function calculates and logs brightness values for a series of frames, sorted by color.

```python
def extract_brightness_values(red_video_frames, green_video_frames,
    yellow_video_frames):
    # Initialize lists to store mean brightness values for each color
    red_brightness_values = []
    ...

    # Calculate mean brightness for each frame for Red light and store in
     list
    for rframe in red_video_frames:
        rbrightness = get_brightness(rframe)
        red_brightness_values.append(rbrightness)

    # Calculate mean brightness for each frame for Green light and store
    in list
    ...

    # Calculate mean brightness for each frame for Yellow light and store
     in list
    ...

    return red_brightness_values, green_brightness_values,
    yellow_brightness_values
```

Listing 4.3: Extracting Brightness Calculation Function

5. **K-means Clustering for Brightness Values:** The `cluster_brightness_values` function employs k-means clustering to identify two distinct cluster centers, which signify the ON and OFF states of each traffic light color, which can change based on various

conditions. This creative solution introduces a level of adaptability and robustness to our implementation while still maintaining a relatively simple and lightweight approach.

```
def cluster_brightness_values(brightness_values: List[float]) -> Tuple[float, float]:
    try:
        # Create an array for the input to k-means clustering
        data = np.array(brightness_values, dtype=np.float32).reshape(-1, 1)

        # Number of clusters
        K = 2

        # Define criteria and apply k-means()
        criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 100, 0.001)
        attempts = 10
        flags = cv.KMEANS_RANDOM_CENTERS

        ret, labels, centers = cv.kmeans(data, K, None, criteria, attempts, flags)

        # Extract the individual cluster centers
        center_1, center_2 = centers.ravel()

        return center_1, center_2
    except Exception as e:
        print(f"An error occurred in cluster_brightness_values: {e}")
        return 0.0, 0.0  # Return default values
```

Listing 4.4: K-means Clustering for Brightness Values Function

6. **Main K-means Function:** The `k_means` function wraps around the clustering function, determining specific brightness clusters for Red, Green, and Yellow lights as well as their centers, which are later organized into upper or lower centers based on their values.

```
def k_means(red_frames, green_frames, yellow_frames):
    # Extract mean brightness values from frames of each color
    red_brightness_values, green_brightness_values, yellow_brightness_values = extract_brightness_values(red_frames, green_frames, yellow_frames)

    # Find the cluster centers for each color by applying k-means on their brightness values
    rcenter_1, rcenter_2 = cluster_brightness_values(red_brightness_values)
    ...

```

```
    return rcenter_1, rcenter_2, gcenter_1, gcenter_2, ycenter_1,
    ycenter_2
# Run k-means function to get cluster centers for brightness for each
    color
red_center1, red_center2, green_center1, green_center2, yellow_center1,
    yellow_center2 = k_means(red_frames, green_frames, yellow_frames)
```

Listing 4.5: Main K-means Function

```
# Classify the centers as 'upper' or 'lower' based on their values to
    help in determining if a light is ON or OFF
upper_red_center, lower_red_center = max(red_center1, red_center2), min(
    red_center1, red_center2)
...
```

Listing 4.6: Classifying the K-means Centers

7. **Light Status Determination:** The main loop continuously reads frames and utilizes the brightness cluster centers calculated earlier to identify which light is active at any given moment. Once the light is identified as red, the program enters the vehicle detection routine, only exiting it when the light is identified as anything other than red again. The program then terminates and performs cleanup functions either when the video file concludes or when the user presses 'q', which can be done at any stage of the program.

```
while True:
    try:
        # Read a single frame from the VideoCapture object.
        ret, frame = cap.read()

        # Check if a frame has been successfully captured.
        if ret:
            # Crop out the ROIs for Red, Green, and Yellow lights from
    the frame.
            red_frame = frame[ryMin:ryMax, rxMin:rxMax]
            ...

            # Calculate the mean brightness for each of the cropped
    frames.
            red_brightness = get_brightness(red_frame)
            ...

            # Determine if Red, Green, or Yellow light is ON.
            # This is done based on the cluster centers and relative
    brightness levels.
            red_on = abs(red_brightness - upper_red_center) < abs(
    red_brightness - lower_red_center) and \
                     abs(red_brightness - upper_red_center) < abs(
    green_brightness - upper_green_center) and \
```

```
                    abs(red_brightness - upper_red_center) < abs(
    yellow_brightness - upper_yellow_center)
            ...

            # Identify which light is ON based on the boolean conditions
    above, and annotate it on the frame.
            if red_on and not (green_on or yellow_on):
                which_light = "RED"
            ...

            # Determine font scaling and text position based on the frame
     dimensions.
            ...

            # Add text annotation to the original frame and display it.
            ...

            # Update the vehicle detection flag based on light status
            if which_light == "RED" and prev_light != "RED":
                run_vehicle_detection = True  # Start vehicle detection
    when light turns red
            elif which_light != "RED" and prev_light == "RED":
                run_vehicle_detection = False  # Stop vehicle detection
    when light is not red

            prev_light = which_light  # Update previous light status

            # Conditionally run the vehicle detection code
            if run_vehicle_detection:
            ...

            # Exit the loop if 'q' is pressed.
            if cv.waitKey(int(FPS)) & 0xFF == ord('q'):
                ...
        else:
            # If a frame is not captured successfully, exit the loop.
            ...

    except KeyboardInterrupt:
        print("Interrupted by user. Exiting.")
        break

    except Exception as e:
        print(f"An error occurred: {e}")

# Release the VideoCapture object and destroy all OpenCV windows.
cap.release()
cv.destroyAllWindows()
```

Listing 4.7: Light Status Determination Loop

4.3 Vehicle Detection and Capture

This section delves into the implementation of the Vehicle Detection and Capture portion, clarifies the primary components of our chosen solution, and investigates our reasoning for choosing specific techniques over alternative approaches.

First, let's clarify the main objective of this second half of our solution prototype. The primary objective of this part of the implementation is to detect and identify vehicles going in a certain user-defined direction, inside a user-defined area, in a given video file. The program mainly uses OpenCV and Python to accomplish this.

Now let's take a more extended look at its main components:

1. **Global Variables and Initialization:** Global variables are initialized to store the Region of Interest (ROI) and direction vector coordinates as empty lists. Additionally, several variables are declared to facilitate the snapshot and video snippet functionalities in vehicle detection. Further initializations include `bg_subtractor`, `feature_params`, `lk_params`, `frame_buffer`, `FPS`, and `numbFrames`, which serve the following purposes:

   - `bg_subtractor` is initialized for background subtraction, aiming to segregate moving objects from the background.
   - `feature_params` is a dictionary set up to house the parameters essential for feature detection.
   - `lk_params` is another dictionary, configured to contain the parameters for Lucas-Kanade optical flow.
   - `frame_buffer` is initialized as a deque used as a frame buffer to hold 2 seconds' worth of video frames.
   - `FPS` sets the frames per second for the video.
   - `numbFrames` is used to denote the total number of frames in the video.

   Each of these initializations performs a specialized role, contributing to various aspects of the vehicle detection system.

2. **Take snapshot:** The `take_snapshot` function is a simple helper function used to take vehicle snapshots.

```
1 def take_snapshot(frame, filename):
2     cv.imwrite(filename, frame)
```

Listing 4.8: Snapshot Function

3. **Calculate Density:** The `calculate_density` function is a simple helper function used to calculate the vector density of the detected vectors.

```python
def calculate_density(vectors):
    return len(vectors)
```

Listing 4.9: Vector Density Calculation Function

4. **Point Proximity:** The `is_close` function is a simple helper function used to check if two points are close to each other.

```python
def is_close(point1, point2, threshold=50):
    distance = np.sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)
    return distance < threshold
```

Listing 4.10: Point Proximity Calculation Function

5. **Detection Verifier:** The `check_near_recent_detections` function is a simple helper function used to check if a detection is near other recent detections.

```python
def check_near_recent_detections(recent_detections, current_vectors, distance_threshold=50):
    # Flatten the recent_detections and build a KDTree
    all_recent_points = [point for sublist in recent_detections for recent_vector in sublist for point in [recent_vector[0]]]
    if len(all_recent_points) == 0:
        return False
    kdtree = KDTree(all_recent_points)

    # Query the KDTree for each current point
    for current_vector in current_vectors:
        current_point = current_vector[0]
        # Query the tree for nearby points within the distance_threshold
        if len(kdtree.query_ball_point(current_point, distance_threshold)) > 0:
            return True
    return False
```

Listing 4.11: Detection Verifier Function

6. **Setup Tasks:** Additional setup tasks are executed, including the cropping of the vehicle detection ROI, the initialization of a warm-up phase for the background subtractor, and the identification of features to track.

```python
# Crop the first frame to the user-selected ROI
vehicle_frame = np.array([l[vxMin:vxMax] for l in first_frame[vyMin:vyMax]])

# Convert the cropped frame to grayscale and detect features
prev_gray = cv.cvtColor(vehicle_frame, cv.COLOR_BGR2GRAY)
prev = cv.goodFeaturesToTrack(prev_gray, mask=None, **feature_params)
mask = np.zeros_like(vehicle_frame)

```

```
# Warm-up phase
warm_up_frames = 20
for frameNum in range(warm_up_frames):
    ret, frame = cap.read()
    if not ret:
        print("Failed to read a warm-up frame.")
        exit()
    frame = frame[vyMin:vyMax, vxMin:vxMax]
    bg_subtractor.apply(frame)

# Initialize features for tracking after warm-up
ret, first_frame = cap.read()
if not ret:
    print("Failed to read the video.")
    exit()
```

Listing 4.12: Vehicle Detection Setup Tasks

7. **Red Light Violation Determination:** As previously stated, upon detecting a red light, the program transitions into the vehicle detection phase and remains in this state until the light turns green again. During this phase, the program continually reads video frames and employs both Lucas-Kanade Optical Flow and background subtraction techniques to identify vehicles traversing the designated prohibited area ROI, as mentioned before in this paper, these techniques were chosen for their balance between accuracy and computational complexity. The program uses the direction vector, which is manually drawn by the user, to evaluate and filter only the vehicles moving in the prohibited direction. This detection is activated only when a specific density of direction vectors is observed. In the event of a violation, the program captures both a snapshot and a brief video snippet of the vehicle running the red light. Finally, the program will terminate and perform necessary cleanup tasks either when the video concludes or if the user presses the "q" key to exit.

```
if run_vehicle_detection:
    if not ret:
        if 'out' in locals() and out.isOpened() and not video_finalized:
     # Check the flag here
            out.release()
        break

    user_direction_vectors = []
    if snapshot_cooldown_counter > 0:
        snapshot_cooldown_counter -= 1

    # Crop the current frame to the user-selected ROI
    vehicle_frame = frame[vyMin:vyMax, vxMin:vxMax]

    # Apply background subtraction and morphological operations
    ...
```

```
16
17      # Detect features and compute the optical flow for the current frame
18      gray = cv.cvtColor(vehicle_frame, cv.COLOR_BGR2GRAY)
19      prev = cv.goodFeaturesToTrack(gray, mask=fg_mask_binary, **
        feature_params)
20
21      # Check if any features were detected
22      if prev is not None and len(prev) > 0:
23          next, status, error = cv.calcOpticalFlowPyrLK(prev_gray, gray,
        prev, None, **lk_params)
24          if next is not None:
25              good_old = prev[status == 1]
26              good_new = next[status == 1]
27
28              # For each detected vehicle, determine its direction
29              for i, (new, old) in enumerate(zip(good_new, good_old)):
30                  ...
31
32                  # If the user has provided a specific direction, compare
        it
33                  if len(direction_points) == 2:
34                      ...
35                          # If angle difference is small, detection is True
36                          if angle_diff < np.radians(17):
37                              detection = True
38                      else:
39                          # If either vector is a zero vector, skip the
        current iteration.
40                          continue
41
42                  if detection:
43                      user_direction_vectors.append((new, old))
44
45              # Calculate density of vectors in the user drawn direction
46              density = calculate_density(user_direction_vectors)
47              # Add the current frame to the rolling buffer
48              frame_buffer.append(frame.copy())
49
50              #Handle snapshot and video recording
51              if density > density_threshold and snapshot_cooldown_counter
        == 0 and not is_video_recording:
52                  ...
53
54          # Update the previous grayscale frame and the set of good
        features for the next iteration.
55          prev_gray = gray.copy()
56          prev = good_new.reshape(-1, 1, 2)
```

Listing 4.13: Red-Light Violation Determination Loop

4.4 UI and other Visual Components

This section aims to highlight the user interface (UI) and other visual elements of the implementation that cannot be adequately conveyed through code or textual explanations alone. As previously mentioned, the system is currently in its prototype stage; therefore, the UI remains relatively simplistic yet strives to be user-friendly. The primary features showcased in this section include the interface for ROI (Region of Interest) selection, visual indicators for traffic light detection, and the program's output in the form of snapshots and video snippets, as already stated.

1. **ROI Selection:** As previously mentioned, the user is guided to draw a four-point polygon to encapsulate each distinct traffic light color, as well as the area designated for vehicle detection. Additionally, the user is prompted to sketch a line that will serve as a direction vector; vehicles moving in this direction will be specifically targeted for detection. On-screen text assists the user through these selections. Importantly, the user has the option to redraw the polygon should there be a misclick or any other error. Below are some figures illustrating portions of this selection process:

Figure 4.1: The user is prompted to start drawing the first ROI

Figure 4.2: The user is prompted to press any key after drawing the RED ROI

Figure 4.3: The user is prompted to select 1 more point to complete the GRN ROI

Figure 4.4: Exaple of a vehicle ROI

Figure 4.5: Exaple of a direction vector

2. **Visual Indicators:** Once the user has successfully selected all the ROIs and the direction vector, the program proceeds to analyze the color of the traffic light as the video plays. This ongoing analysis is made visible to the user through on-screen visual indicators. Below is a figure that illustrates this aspect:

Figure 4.6: Visual indicator that the traffic light is currently green

3. **Outputs:** After identifying a vehicle within the designated vehicle detection zone, moving in the direction specified by the user, the software proceeds to capture an image of the vehicle that ran a red light and generates a brief video clip showcasing the violation. Below are some figures that illustrate this:

Figure 4.7: Snapshot of a red light runner

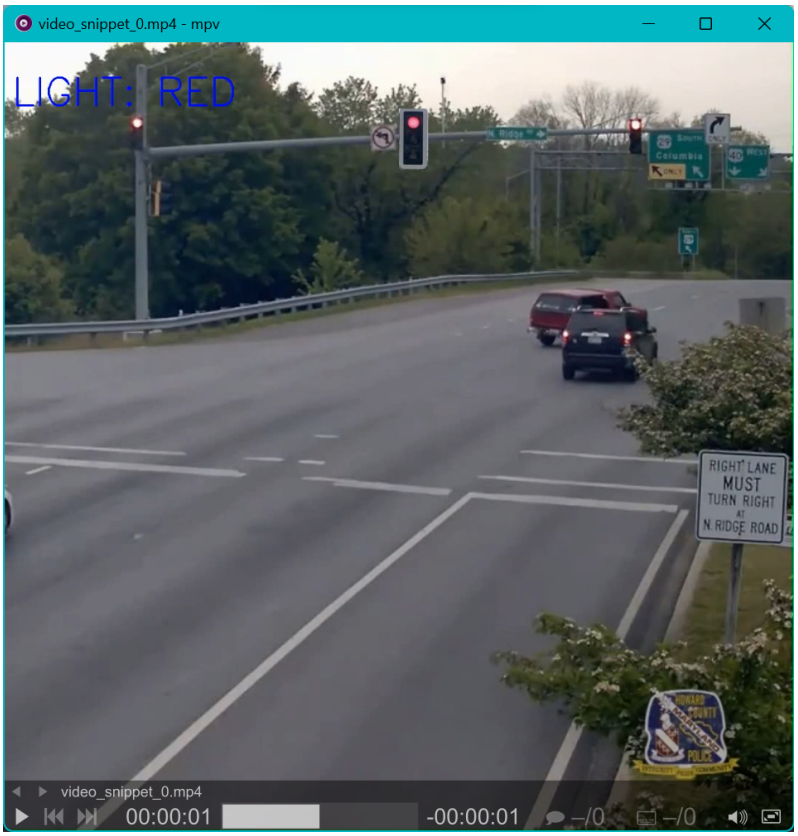

Figure 4.8: Brief video of the violation

4.5 Chapter Conclusions

In conclusion, this chapter outlined the methodology used in the development of our proposed traffic red light violation detection solution. The solution was created using OpenCV and Python, emphasizing its lightweight, mobile, and precise characteristics.

In the following chapter, appropriately titled "Findings," we will go over the outcomes and success rates derived from an assortment of tests. These tests have been chosen to assess the efficacy of each independent component within our solution, as well as its overall performance when put through real-world scenarios during complete implementation. However, due to data constraints, certain adjustments and artificial tests were required, resulting in a somewhat constrained pool of test data for analysis.

5

Findings

In this chapter, we will look at the results of our implementation and highlight the most important findings. It is critical to emphasize that the domain under investigation suffers from a significant lack of available video data pertaining to vehicles disregarding red traffic signals. As a result, we divided our implementation evaluation into three distinct phases, mirroring the ones discussed in the preceding chapter. These phases include an examination of traffic light color detection, an assessment of vehicle detection, and a simultaneous evaluation of both components. It's worth noting that the third phase, which involves the simultaneous evaluation of both parts, is the most difficult due to the previously mentioned scarcity of video data in this context.

Furthermore, it is critical to acknowledge that our ability to collect proprietary data for this study was severely limited. Filming intersections and roadways with traffic lights is a difficult endeavor due to the legal ambiguity, which restricts our ability to gather fresh video data. Additionally, the likelihood of capturing instances of vehicles running red lights was extremely low, complicating our data collection efforts even further.

Moreover, it is critical to recognize that our ability to collect appropriate data was further limited by the requirement that the videos used remain static. Any video instability has a significant impact on the accuracy of traffic light color detection and vehicle detection algorithms.

While our system consistently produced excellent and, on occasion, flawless results across a variety of tests, it is critical to note that these results were obtained in a controlled testing environment. In everyday, real-world situations, achieving absolute accuracy is extremely unlikely. We acknowledge that the quality of testing materials and data availability had a significant influence on these results.

5.1 Traffic Light Detection

As previously stated, due to a lack of available data, the two primary components of this solution were initially tested separately. The Traffic Light Detection component was tested using a variety of videos from various angles, quality levels, and distances to determine

its accuracy. It is critical to emphasize that the operation of the traffic light detection is dependent on video stability and the presence of at least one full cycle of the traffic light. In other words, for the detection to work properly, the video must show the traffic light changing colors throughout its cycle. While this requirement increased the difficulty of obtaining suitable video data, it also increased the detection system's robustness and precision.

The evaluation of the Traffic Light Detection component, conducted with the video data we managed to acquire, yielded highly favorable results. The program consistently identified the correct traffic light colors across all the videos, irrespective of the diverse conditions they presented. While results like these are only possible under perfect or close to perfect conditions and a controlled environment, our testing efforts have instilled confidence in asserting that the program accurately detects traffic light colors under ideal circumstances, as well as in scenarios characterized as good and regular conditions.

It is crucial to emphasize that the detection's effectiveness may be reduced when confronted with shaky videos or those lacking a complete traffic light cycle.

5.2 Vehicle Detection

The vehicle detection portion of the program, like the preceding section, underwent initial independent testing until it demonstrated consistent positive results. We rigorously evaluated the program's performance by utilizing a diverse set of video data featuring moving vehicles captured from various angles, distances, and directions. This extensive testing was required to ensure that the program could detect vehicles in a variety of real-world scenarios.

It is important to note that, in our implementation, vehicle detection is a more complex task than traffic light detection because it relies on optical flow and background subtraction techniques, which are highly sensitive to noise and camera movement. Another challenge was determining what should be classified as a vehicle and what should not. These criteria can vary significantly depending on the camera's angle and distance from the moving objects, influencing the average vector density of a moving vehicle. As a result, we had to establish an arbitrary threshold for this density to test with video data, taking into account the typical proximity of vehicles to the camera in various detection scenarios.

Our testing yielded very encouraging results, with the majority of vehicles being detected correctly in their direction. We recognize that results like these while encouraging, have to be taken cautiously due to the controlled environment they are under. Nonetheless, the program demonstrated the ability to filter out noise and ignore smaller objects, such as pedestrians or animals.

5.3 Full Implementation

The complete implementation of our solution encountered an even more pronounced challenge related to the scarcity of available video data. This problem was exacerbated by the fact that, as previously stated, our solution relies on videos that include a full cycle of traffic light colors as well as a vehicle running a red light. The rarity of such specific conditions, coupled with the impracticality of generating our own video data, led us to create and modify a set of test videos. These fabricated videos were created by combining segments from different traffic light videos that we discovered with excerpts from traffic light violation videos. While these videos are fabricated, we believe that using them does not significantly impair the quality of our results. The assertion is based on the proven accuracy of our traffic light detection, which comprises the artificially incorporated segments. Despite the use of edited videos, we are confident in the overall results of our findings.

In addition to traffic light and vehicle detection, the full implementation allows for the capture of a snapshot and a brief video snippet of the red light runner. These features were also tested using a variety of videos. We achieved good results, despite recognizing that they are not flawless. The snapshots are taken a few frames after a vehicle is detected to ensure a complete image of the vehicle, whereas the video snippet captures moments before, during, and after the violation. While these features are, as just mentioned, not without flaws, their overall success rate contributes to the overall effectiveness of our solution.

5.4 Success Rates

- **Traffic Light Detection:** As previously noted, we achieved excellent results during the testing of the traffic light detection component in isolation. We conducted testing using a diverse set of ten videos, each featuring distinct angles, distances, and varying levels of video quality. The metric used to determine success was the accurate detection of the correct color while that color was displayed on the traffic light and the transition between detected colors in synchronization with the video. The outcomes of these tests are summarized in 5.1.

- **Vehicle Detection:** Much like the evaluation of the traffic light detection component, we also rigorously tested the vehicle detection portion. Our assessment encompassed two distinct sets of data. The first set comprised six regular videos depicting vehicles in various scenarios, including interstates, streets, and other settings. The second set featured videos capturing vehicles running red lights from diverse angles, distances, and times of day. The metric used to determine success was the program's ability to detect all the vehicles on screen. False positives were mostly addressed, as the program only considered vehicle detection when a certain density of movement

vectors was met. Additionally, a separate test was conducted specifically to evaluate both vehicle detection and direction using the same videos, and the results of this test will be presented in a separate table. While the results obtained for vehicle detection were not flawless, they were highly commendable. The outcomes of these tests are summarized in 5.2 and 5.3.

- **Complete Violation Detection:** Our testing efforts concluded with the complete Red Light Violation Detection, which incorporates both the traffic light color detection and the vehicle detection components. As mentioned before, these final tests presented a unique challenge in that we had to create artificial data by combining segments of red light violation videos with footage of a complete traffic light color cycle. Using this method, we were able to thoroughly assess the performance of our program as a whole. We tested five of these fabricated videos, and the results were very favorable. The metric used to assess success was the program's ability to detect the perpetrator, capture a snapshot at the appropriate moment, and record a video snippet of the violation. This testing approach confirmed the robustness and effectiveness of our entire Violation Detection system, even if tested under a controlled environment with fabricated data. The outcomes of these tests are summarized in 5.4.

5.5 Chapter Conclusions

Based on the test results, it is clear that our Violation Detection system has a commendable level of robustness and effectiveness in detecting red light violations. Notably, we had a 100% success rate in both snapshot and video snippet capture, indicating that our program correctly identified the transgressor, took a snapshot at the critical time, and recorded a video snippet of the violation. However, it is essential to recognize that these flawless results reflect our system's performance in the controlled testing environment. In real-world, day-to-day scenarios, achieving 100% accuracy is extremely unlikely, and we recognize that a lack of quality testing material and data influenced these results. Furthermore, while the vehicle detection results in a vacuum were not perfect, they still achieved a high level of effectiveness. The majority of false positives were successfully reduced, and the program demonstrated the ability to detect all vehicles in the field of view.

In the next and final chapter, titled "Concluding Reflections and Future Work," we will summarize the key findings of this study and suggest potential future research directions. We'll look at the study's limitations and potential improvements, investigate the practical applications of our Violation Detection system in real-world contexts, and identify areas for future research in this field.

Traffic Light Color Detection Success Rate			
Video	RED(%)	GRN(%)	YLW(%)
tl1	100,00	100,00	100,00
tl2	100,00	100,00	100,00
tl3	100,00	100,00	100,00
tl5	100,00	100,00	100,00
tl6	100,00	100,00	100,00
tl7	100,00	100,00	100,00
tl8	100,00	100,00	100,00
tl9	100,00	100,00	100,00
tl10	100,00	100,00	100,00
tl11	100,00	100,00	100,00
Total	100,00	100,00	100,00

Table 5.1: Success rates for all traffic light detection videos

Vehicle Detection Success Rate - Regular Videos		
Video	Detection(%)	Direction(%)
city	100,00	100,00
highway	96,67	100,00
highway2	100,00	95,00
istate	100,00	100,00
roundb2	100,00	90,00
traffic	95,23	85,71
Total	98,65	95,17

Table 5.2: Success rates for all regular vehicle detection videos

Vehicle Detection Success Rate - Violation Videos		
Video	Detection(%)	Direction(%)
red1	96,67	100,00
red2	100,00	100,00
red3	100,00	100,00
red4	100,00	100,00
red5	90,00	90,00
red6	100,00	100,00
red7	100,00	100,00
red8	100,00	100,00
red9	100,00	100,00
red10	100,00	100,00
Total	98,7	99,00

Table 5.3: Success rates for all vehicle detection violation videos

Red Light Violation Detection Success Rate		
Video	Snapshot(%)	Snippet(%)
test1	100,00	100,00
test2	100,00	100,00
test3	100,00	100,00
test4	100,00	100,00
test5	100,00	100,00
Total	100,00	100,00

Table 5.4: Success rates for snapshots and video snippets for red light violation detection

6

Concluding Reflections and Future Work

In this final chapter, we present our final thoughts on the work presented in this thesis and outline potential future research directions. We began this research with the goal of creating a lightweight, mobile, and inexpensive system for detecting traffic red light violations using image processing techniques. We demonstrated the feasibility and effectiveness of such a system using a combination of existing solutions and our own novel approach. In this chapter, we reflect on the main discoveries and deductions made throughout the study, as well as discuss the potential impact of our proposed solution on road safety. We also discuss potential future work directions aimed at refining and completing the proposed solution.

6.1 Concluding Reflections

The primary goal of our research, as stated before, was to create and test the viability of a lightweight, mobile, and cost-effective system for detecting traffic red light violations using image processing techniques, with the ultimate goal of enabling real-time detection. To achieve this goal, our research methodology included the development of a Python solution that incorporated a variety of image processing and detection methods. This proposed algorithm was tested using both real-world and synthetic datasets, yielding results that confirm the feasibility of future implementation and the use of this prototype as a foundation for a real-time traffic red light violation detection solution.

Furthermore, as demonstrated in our previous research [5], this study shed light on the potential applications of violation detection algorithms in improving road safety by identifying habitual offenders and incentivizing greater caution on the road. The proposed algorithm may also be extended to other aspects of road safety, acting as a reference or source of inspiration for the development of additional solutions and implementations aimed at fostering safer road environments. The algorithm we have proposed and all the datasets employed in this study have been made publicly available to facilitate future

comparisons and research endeavors, as outlined in the "Material" section of the first chapter. This open-access approach encourages fellow researchers to further develop and augment the capabilities of the proposed algorithm to cater to specific application requirements and contribute to the advancement of this field.

In conclusion, the findings of this dissertation make a significant contribution to the advancement of red light violation detection and provide valuable insights into the broader potential of traffic violation detection for improving road safety. Future research efforts can focus on fine-tuning the proposed algorithm for real-time applications, tailoring it to specific use cases, and investigating the integration of traffic violation detection into other domains of image processing and detection. The proposed algorithm and associated datasets' accessibility will undoubtedly encourage further research in this field, paving the way for the development of more advanced violation detection techniques and, ultimately, safer road environments.

6.2 Future Work

In this chapter, we evaluate the limitations of our current traffic red light violation detection system and propose potential future development paths. Furthermore, we highlight the domain's challenges and opportunities, providing recommendations to guide future research initiatives. The primary goal of this chapter is to provide a strategic list for researchers and engineers in order to promote the evolution of traffic red light violation detection technology, with the overarching goal of improving road safety and efficiency.

6.2.1 Real-Time Operation

The primary goal of this thesis, as previously stated, was to develop a real-time violation detection system, and this overarching goal remains the central focus of our project. At the moment, our prototype only functions with pre-recorded video input and does not accept live video feed. Bridging this gap in our implementation will necessitate a number of significant changes. Nonetheless, our prototype serves as a proof of concept, demonstrating the viability of developing a more streamlined and simplified solution to the red light violation detection problem. The current implementation provides a solid foundation for the development of a real-time iteration as many of the techniques and concepts used are still applicable. Given the constraints in data availability and the practical challenges of testing a real-time setup, we decided that developing a prototype designed for video input was the most practical approach. This method allows us to lay a solid foundation before moving on to the more challenging task of implementing real-time functionality with live video feed.

There still exists plenty of room for future work in achieving our ultimate goal of a fully operational real-time violation detection system. We can adapt the techniques used in our current implementation for real-time processing, such as optical flow, background

subtraction, and brightness detection. We can, for example, consider the modification of techniques such as k-means clustering to operate within timed cycles, ensuring an adaptable and robust system capable of functioning effectively under a variety of weather and lighting conditions.

Furthermore, it is essential to reevaluate and potentially revise certain techniques, acknowledging that some aspects of our current implementation may rely on pre-recorded video input rather than a live video feed. This requires a fundamental change in which we modify these techniques to the real-time nature of a live video feed, potentially necessitating the development of new algorithms or adaptations of existing ones.

In short, working toward a fully functional real-time violation detection system involves improving and adapting existing techniques, exploring new methods, and integrating these components into a cohesive system capable of performing reliably in real-world conditions.

6.2.2 Optimization

Multiple parameters and techniques can be adjusted to achieve superior results when attempting to improve the performance of an algorithm. Parameter selection decisions, as well as using appropriate techniques, have a significant impact on the algorithm's effectiveness. The following section delves into the various parameters and techniques that could be changed, as well as the methodologies available for optimizing them. Ultimately, the goal is to develop algorithms that are more efficient and precise in their operations.

6.2.2.1 Fixed Values

Our red light violation detection algorithm's use of fixed values has proven effective in a variety of scenarios. However, these fixed values, which include parameters such as the vector density threshold, snapshot cooldown, and optical flow and background subtraction settings, may not be universally optimal. Their adaptability is essential for improving the algorithm's precision and robustness, as factors like the number and density of movement vectors in the scene can have a big impact on its performance. These variables can vary significantly depending on camera distance and angle.

To address this need for adaptability, the algorithm can begin by balancing accuracy and optimization using well-established constants. Nonetheless, these fixed values should be dynamically adjusted based on scene-specific characteristics such as movement vector density and number. These adjustments may include the use of machine learning algorithms and other analytical techniques to effectively assess the scene. For example, if the camera is placed further away from the observed scene, causing vehicles to appear smaller, the algorithm would lower the vector density threshold for vehicle detection, assisting in the accurate distinction between vehicles and other objects. In contrast, in close-up camera scenarios where vehicles appear larger and more visible, the algorithm would increase these fixed values to improve precision and minimize false positives.

Another scenario in which adaptability is critical is in vehicle-heavy environments. In such cases, the algorithm may need to use a shorter snapshot cooldown to ensure timely detection of potential violations in scenes where more than one violation might occur.

6.2.2.2 Background Subtraction

The MOG2 background subtraction method was used in our implementation of the red light violation detection system. Extensive testing and research influenced this decision as explained in Background Subtraction. However, it is important to note that the applicability of this method may vary depending on the circumstances. As a result, more research and refinement of background subtraction techniques are required to determine the best approach for this implementation.

Furthermore, as discussed in Fixed Values, the background subtraction parameters used in our system produced satisfactory results. However, these parameter values may not always correspond to the needs of all scenarios. Experimenting with different parameter values or exploration of dynamic parameter values, tailored to adapt to varying environmental conditions, could be potential avenues for improving the system's robustness and accuracy in background subtraction. Background subtraction is a critical component of our solution, and optimizing it is critical for overall system performance.

6.2.2.3 Optical Flow

Optical flow, like background subtraction, is critical in the context of our red light violation detection system. There are several optical flow methods, and we specifically chose LK optical flow in our implementation due to its conceptual clarity and computational efficiency, as detailed in Optical Flow. However, as with our considerations for the MOG2 method in background subtraction, it is important to note that, while LK optical flow produced favorable results, it may not be suitable for all scenarios. As a result, there is room for future investigation and research to identify an optical flow method that achieves the best balance of accuracy and computational efficiency for our solution.

Furthermore, as described in Fixed Values, our implementation used fixed parameters for Lucas-Kanade optical flow, which resulted in very positive results. These fixed parameters, however, may not be universally applicable to every situation. To improve the robustness and accuracy of our system, we may want to experiment with different parameter values or implement adaptive parameter settings. This change could be especially significant because optical flow is the foundation of our vehicle detection component.

6.2.2.4 Clustering

It is critical to assess the suitability of the K-Means clustering algorithm for this task, as we did in the two previous subsubsections. In the future, it would be useful to conduct an analysis to compare its effectiveness to other well-established and widely used algorithms. In this solution, it is important to strike a balance between simplicity and accuracy. Below

we outline some clustering algorithms that may be suitable for brightness value analysis and could be used as alternatives to K-Means:

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** DBSCAN is good at grouping data points that are close to one another in high-density areas. What distinguishes DBSCAN from other widely used methods is its ability to identify clusters with irregular shapes while also effectively handling noisy data points. These features of DBSCAN are especially useful when dealing with scenarios involving variations in brightness or data distributions that do not neatly conform to a spherical shape, as assumed by the K-Means algorithm [63] [64].

- **Agglomerative Clustering:** Agglomerative Clustering is a hierarchical clustering technique that starts by treating each data point as a separate cluster and then gradually combines clusters based on a specific linkage criterion. This merging process is repeated until only two clusters remain, allowing a hierarchical representation of the data's structure to be obtained. Given these properties, Agglomerative Clustering could be used to merge clusters within brightness data iteratively until exactly two clusters are obtained. This approach can be particularly beneficial when the number of clusters is not predetermined, although in our specific case, this is not a requirement [65] [66].

- **Mean-Shift Clustering:** Mean-Shift Clustering is a non-parametric clustering algorithm that finds clusters by adjusting data points iteratively toward the mode or peak of the data distribution. This method is advantageous because it adapts to the shape and size of clusters and does not require the number of clusters to be specified in advance. Its effectiveness shines through when clusters have varying shapes or sizes, making it a useful and versatile tool in various situations [67] [68] [69].

- **Gaussian Mixture Model (GMM):** The Gaussian Mixture Model (GMM) is a clustering method based on a probabilistic model. It represents the data as a combination of Gaussian distributions, allowing for flexibility in accommodating different cluster shapes and orientations. In our case, applying GMM with two components to the brightness data could be advantageous. The resulting Gaussian distributions could effectively capture the traffic light's "on" and "off" states, making it an appropriate option for this application [70] [71].

- **Affinity Propagation:** Affinity Propagation is a graph-based clustering algorithm that excels at identifying and assigning data points to exemplars, which are data points that effectively represent clusters. This algorithm's ability to autonomously determine the number of clusters is notable, making it especially useful when there are distinct exemplars, in our case, for the "on" and "off" states of the traffic light. This feature translates well with many real-world situations, making Affinity Propagation a great option in such cases [72] [73] [74].

Ultimately, we demonstrated the feasibility and effectiveness of our proposed "Red Light Violation Detection" system prototype by combining existing solutions with our novel approach. We see this work as an important milestone and provide a possible roadmap for those interested in contributing to the improvement of our proposed solution. As we embark on this journey of progress, we invite fellow researchers and enthusiasts to actively participate in shaping the evolution of this technology. We eagerly await further growth and advancement in this field.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[2] IIHS. *Red light running*. 2022. URL: https://www.iihs.org/topics/red-light-running (cit. on p. 1).

[3] Y. Amit, P. Felzenszwalb, and R. Girshick. "Object Detection". In: *Computer Vision* (2020), pp. 1–9. DOI: 10.1007/978-3-030-03243-2{_}660-1. URL: https://link.springer.com/referenceworkentry/10.1007/978-3-030-03243-2_660-1 (cit. on p. 1).

[4] D. Fortun, P. Bouthemy, and C. Kervrann. "Optical flow modeling and computation: A survey". In: *Computer Vision and Image Understanding* 134 (2015-05), pp. 1–21. ISSN: 10773142. DOI: 10.1016/j.cviu.2015.02.008. URL: https://linkinghub.elsevier.com/retrieve/pii/S1077314215000429 (cit. on p. 2).

[5] C. Goldenbeld, S. Daniels, and G. Schermers. "Red light cameras revisited. Recent evidence on red light camera safety effects". In: *Accident Analysis & Prevention* 128 (2019-07), pp. 139–147. ISSN: 00014575. DOI: 10.1016/j.aap.2019.04.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S0001457518303610 (cit. on pp. 2, 14, 15, 25, 61).

[6] M. Petrou Costas Petrou and W. A. John Wiley and. *Image processing: the fundamentals*. 2010. URL: https://books.google.com/books?hl=en&lr=&id=w3BpSIxN9ZYC&oi=fnd&pg=PR23&dq=what+is+image+processing&ots=3MesfBZB2F&sig=wk4BlyR6OSA9Pe22QQ6XRnIkjIQ (cit. on p. 5).

[7] B Chitradevi, P Srimathi, and A. Professor. "An Overview on Image Processing Techniques". In: *International Journal of Innovative Research in Computer and Communication Engineering (An ISO* 3297.11 (2007). ISSN: 2320-9798. URL: www.ijircce.com (cit. on p. 5).

[8] L. Jorge. *The Science of Color: Understanding Color Spaces in Image Processing*. URL: https://medium.com/@livajorge7/the-science-of-color-understanding-color-spaces-in-image-processing-d0e238872a0c (cit. on p. 6).

[9] MathWorks. *Understanding Color Spaces and Color Space Conversion - MATLAB & Simulink*. URL: https://www.mathworks.com/help/images/understanding-color-spaces-and-color-space-conversion.html (cit. on pp. 6, 7).

[10] OpenCV. *OpenCV: Color conversions*. URL: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_hsv (cit. on p. 6).

[11] J. Frost. *What is K Means Clustering? With an Example*. URL: https://statisticsbyjim.com/basics/k-means-clustering/ (cit. on pp. 7, 8).

[12] M. Banoula. *K-means Clustering Algorithm: Applications, Types, and Demos*. URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/k-means-clustering-algorithm (cit. on pp. 7, 8).

[13] V. Kumar. *K-Means*. URL: https://towardsmachinelearning.org/k-means/ (cit. on pp. 7, 8).

[14] R. Nirek. *K-Means Clustering Algorithm. Brief: K-means clustering is an... | by Ribhu Nirek | Level Up Coding*. URL: https://levelup.gitconnected.com/k-means-clustering-algorithm-4fc5f1d74a23 (cit. on p. 8).

[15] D. Bloisi. *OpenCV: How to Use Background Subtraction Methods*. URL: https://docs.opencv.org/4.5.2/d1/dc5/tutorial_background_subtraction.html (cit. on p. 8).

[16] S. K. Singh. "A COMPARATIVE STUDY OF DIFFERENT MOTION DETECTION ALGORITHMS IN COMPUTER VISION APPLICATIONS". In: *International Research Journal of Modernization in Engineering Technology and Science* (), pp. 2582–5208. URL: www.irjmets.com (cit. on p. 9).

[17] K. Singh et al. "A COMPARITIVE STUDY OF MOG AND KNN FOR FOREGROUND DETECTION". In: (2018), p. 855. URL: www.jetir.org (cit. on p. 9).

[18] T. Trnovszký, P. Sýkora, and R. Hudec. "Comparison of Background Subtraction Methods on Near Infra-Red Spectrum Video Sequences". In: *Procedia Engineering* 192 (2017-01), pp. 887–892. ISSN: 1877-7058. DOI: 10.1016/J.PROENG.2017.06.153 (cit. on p. 9).

[19] S. Qasim et al. *EasyChair Preprint Performance Evaluation of Background Subtraction Techniques for Video Frames Performance Evaluation of Background Subtraction Techniques for Video Frames*. Tech. rep. 2021 (cit. on p. 9).

[20] Z. Zivkovic. "Improved adaptive Gaussian mixture model for background subtraction". In: *Proceedings - International Conference on Pattern Recognition* 2 (2004), pp. 28–31. ISSN: 10514651. DOI: 10.1109/ICPR.2004.1333992 (cit. on p. 9).

[21] Z. Zivkovic and F. Van Der Heijden. "Efficient adaptive density estimation per image pixel for the task of background subtraction". In: *Pattern Recognition Letters* 27.7 (2006-05), pp. 773–780. ISSN: 0167-8655. DOI: 10.1016/J.PATREC.2005.11.005 (cit. on p. 9).

[22] K. H. Cheong et al. "Practical Automated Video Analytics for Crowd Monitoring and Counting". In: *IEEE Access* 7 (2019), pp. 183252–183261. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2958255 (cit. on p. 9).

[23] Z. Mu and Z. Li. "A Novel Shi-Tomasi Corner Detection Algorithm Based on Progressive Probabilistic Hough Transform". In: *Proceedings 2018 Chinese Automation Congress, CAC 2018* (2019-01), pp. 2918–2922. DOI: 10.1109/CAC.2018.8623648 (cit. on p. 10).

[24] N. Gandhi. *Harris Corner Detection and Shi-Tomasi Corner Detection*. URL: https://medium.com/pixel-wise/detect-those-corners-aba0f034078b (cit. on p. 10).

[25] OpenCV. *OpenCV: Shi-Tomasi Corner Detector & Good Features to Track*. URL: https://docs.opencv.org/4.x/d4/d8c/tutorial_py_shi_tomasi.html (cit. on p. 10).

[26] M. Anandhalli et al. "Corner based statistical modelling in vehicle detection under various condition for traffic surveillance". In: *Multimedia Tools and Applications* 81.20 (2022-08), pp. 28849–28874. ISSN: 15737721. DOI: 10.1007/S11042-022-12422-0/METRICS. URL: https://link.springer.com/article/10.1007/s11042-022-12422-0 (cit. on p. 11).

[27] N. Patel and K. N. Brahmbhatt. "A Video-Based System for Vehicle Tracking Based on Optical Flow and Shi-Tomasi Corner Detection Algorithm". In: (2023), pp. 715–723. DOI: 10.1007/978-981-99-1479-1{_}53. URL: https://link.springer.com/chapter/10.1007/978-981-99-1479-1_53 (cit. on p. 11).

[28] B. Chiang and J. Bohg. "Optical and Scene Flow". In: (2022). URL: https://web.stanford.edu/class/cs231a/course_notes/09-optical-flow.pdf (cit. on p. 11).

[29] OpenCV. *OpenCV: Optical Flow*. URL: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html (cit. on pp. 11–13).

[30] D. J. Fleet and Y. Weiss. "Optical Flow Estimation". In: (). URL: https://www.cs.toronto.edu/~fleet/research/Papers/flowChapter05.pdf (cit. on p. 11).

[31] C.-e. Lin. *Introduction to Motion Estimation with Optical Flow*. URL: https://nanonets.com/blog/optical-flow/ (cit. on p. 12).

[32] A. Aminfar et al. "Application of optical flow algorithms to laser speckle imaging". In: *Microvascular Research* 122 (2019-03), pp. 52–59. ISSN: 00262862. DOI: 10.1016/j.mvr.2018.11.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0026286218302164 (cit. on p. 12).

[33] T. Pallejà et al. "Implementation of a robust absolute virtual head mouse combining face detection, template matching and optical flow algorithms". In: *Telecommunication Systems* 52.3 (2013-03), pp. 1479–1489. ISSN: 10184864. DOI: 10.1007/S11235-011-9625-Y/METRICS. URL: https://link.springer.com/article/10.1007/s11235-011-9625-y (cit. on p. 12).

[34] OpenCV. *OpenCV: Template Matching*. URL: https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html (cit. on p. 13).

[35] MathWorks. *Object for estimating optical flow using Lucas-Kanade method - MATLAB*. URL: https://www.mathworks.com/help/vision/ref/opticalflowlk.html (cit. on p. 14).

[36] Florida Department of Transportation. *Traffic Infraction Detectors (RLRC)*. 2021. URL: https://www.fdot.gov/traffic/TrafficServices/RLRC.shtm (cit. on p. 14).

[37] IIHS. *Most drivers favor red light cameras, a new survey of 14 big U.S. cities finds*. 2011. URL: https://www.iihs.org/news/detail/most-drivers-favor-red-light-cameras-a-new-survey-of-14-big-u-s-cities-finds (cit. on pp. 15, 25).

[38] NCHRP. "Impact of Red Light Camera Enforcement on Crash Experience A Synthesis of Highway Practice NATIONAL COOPERATIVE HIGHWAY RESEARCH PROGRAM NCHRP SYNTHESIS 310". In: (2003). URL: www.TRB.org (cit. on pp. 17, 18).

[39] LA Times. *L.A. City Council shuts down red-light cameras - Los Angeles Times*. 2011. URL: https://www.latimes.com/local/la-xpm-2011-jul-28-la-me-red-light-cameras-20110728-story.html (cit. on p. 18).

[40] U.S. PIRG Education Fund. *Caution: Red Light Cameras Ahead*. 2011. URL: https://pirg.org/resources/caution-red-light-cameras-ahead/ (cit. on p. 18).

[41] S. Saha et al. "Development of an automated Red Light Violation Detection System (RLVDS) for Indian vehicles". In: (2010-03). DOI: 10.48550/arxiv.1003.6052. URL: https://arxiv.org/abs/1003.6052v2 (cit. on pp. 18–20, 23, 33, 35).

[42] P. Polhan and J. Morris. *Imaging Red Light Runners*. Tech. rep. 2016 (cit. on pp. 20, 21, 23, 25, 26, 30, 35).

[43] L. Q. Thao et al. "Automatic Detection of Red Light Running Using Vehicular Cameras". In: *undefined* 27.1 (2017-12), pp. 75–80. ISSN: 21167125. DOI: 10.18280/ISI.270109 (cit. on pp. 22, 23).

[44] O. Rostamianfar, F. Janabi-Sharifi, and I. Hassanzadeh. "Visual tracking system for dense traffic intersections". In: *Canadian Conference on Electrical and Computer Engineering* (2006), pp. 2000–2004. ISSN: 08407789. DOI: 10.1109/CCECE.2006.277838 (cit. on p. 23).

[45] R. H. Brasil and A. M. C. Machado. "Automatic Detection of Red Light Running Using Vehicular Cameras". In: *IEEE Latin America Transactions* 15.1 (2017-12), pp. 81–86. ISSN: 15480992. DOI: 10.1109/TLA.2017.7827891. URL: https://www.researchgate.net/publication/312668556_Automatic_Detection_of_Red_Light_Running_Using_Vehicular_Cameras (cit. on p. 23).

[46] N. Aliane et al. "A System for Traffic Violation Detection". In: *Sensors 2014, Vol. 14, Pages 22113-22127* 14.11 (2014-12), pp. 22113–22127. ISSN: 1424-8220. DOI: 10.3390/S141122113. URL: https://www.mdpi.com/1424-8220/14/11/22113/htmhttps://www.mdpi.com/1424-8220/14/11/22113 (cit. on p. 24).

[47] B. Ugbede Umar et al. "Traffic Violation Detection System Using Image Processing". In: *Computer Engineering and Applications* 10.2 (2021). ISSN: 2252-5459 (cit. on p. 24).

[48] H. S. Prasantha. "Traffic Red Light Violation Detection using Image Processing". In: *International Journal of Advances in Engineering and Management (IJAEM* 2.5 (2008), p. 440. DOI: 10.35629/5252-0205440445. URL: www.ijaem.net (cit. on pp. 24, 33).

[49] A. Aeron-Thomas and S. Hess. "Red-light cameras for the prevention of road traffic crashes". In: *Cochrane Database of Systematic Reviews* 2 (2005-04). ISSN: 1469493X. DOI: 10.1002/14651858.CD003862.PUB2/INFORMATION/EN. URL: https://www.cochranelibrary.com/cdsr/doi/10.1002/14651858.CD003862.pub2/fullhttps://www.cochranelibrary.com/cdsr/doi/10.1002/14651858.CD003862.pub2/abstract (cit. on p. 24).

[50] L. Tan et al. "Speech Emotion Recognition Enhanced Traffic Efficiency Solution for Autonomous Vehicles in a 5G-Enabled Space-Air-Ground Integrated Intelligent Transportation System". In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2022-03), pp. 2830–2842. ISSN: 15580016. DOI: 10.1109/TITS.2021.3119921 (cit. on p. 24).

[51] R. Chauhan, K. K. Ghanshala, and R. C. Joshi. "Convolutional Neural Network (CNN) for Image Detection and Recognition". In: *ICSCCC 2018 - 1st International Conference on Secure Cyber Computing and Communications* (2018-07), pp. 278–282. DOI: 10.1109/ICSCCC.2018.8703316 (cit. on p. 24).

[52] Innodata. *Ethical Issues in Computer Vision and Strategies for Success*. 2022. URL: https://innodata.com/ethical-issues-in-computer-vision-and-strategies-for-success/ (cit. on pp. 25, 35).

[53] P. Wonghabut et al. "Traffic light color identification for automatic traffic light violation detection system". In: *ICEAST 2018 - 4th International Conference on Engineering, Applied Sciences and Technology: Exploring Innovative Solutions for Smart Society* (2018-08). DOI: 10.1109/ICEAST.2018.8434400 (cit. on pp. 26, 27, 30, 35).

[54] M. Omachi and S. Omachi. "Traffic light detection with color and edge information". In: *Proceedings - 2009 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009*. 2009, pp. 284–287. ISBN: 9781424445196. DOI: 10.1109/ICCSIT.2009.5234518 (cit. on pp. 27–30).

[55] A. Kamdar. *Image Segmentation using K Means Clustering - GeeksforGeeks*. URL: https://www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/ (cit. on p. 31).

[56] L. Iliassich. *Clustering Algorithms: K-Means, EMC and Affinity Propagation | Toptal®*. URL: https://www.toptal.com/machine-learning/clustering-algorithms (cit. on p. 31).

[57] A. Muralidharan et al. "Management of intersections with multi-modal high-resolution data q,qq". In: *TRANSPORTATION RESEARCH PART C* 68 (2016), pp. 101–112. DOI: 10.1016/j.trc.2016.02.017. URL: http://dx.doi.org/10.1016/j.trc.2016.02.017 (cit. on p. 31).

[58] R. Kalsotra and S. Arora. "Background subtraction for moving object detection: explorations of recent developments and challenges". In: *The Visual Computer 2021 38:12* 38.12 (2021-08), pp. 4151–4178. ISSN: 1432-2315. DOI: 10.1007/S00371-021-02286-0. URL: https://link.springer.com/article/10.1007/s00371-021-02286-0 (cit. on p. 32).

[59] J. C. Neves, K Wysoczanska, and H Proença. "Evaluation of Background Subtraction Algorithms for Human Visual Surveillance". In: () (cit. on p. 32).

[60] Y. Zhang and Y. Sung. "Traffic Accident Detection Using Background Subtraction and CNN Encoder–Transformer Decoder in Video Frames". In: *Mathematics 2023, Vol. 11, Page 2884* 11.13 (2023-06), p. 2884. ISSN: 2227-7390. DOI: 10.3390/MATH11132884. URL: https://www.mdpi.com/2227-7390/11/13/2884/htmhttps://www.mdpi.com/2227-7390/11/13/2884 (cit. on p. 32).

[61] S. Baker et al. "A Database and Evaluation Methodology for Optical Flow". In: *Int J Comput Vis* 92 (2011), pp. 1–31. DOI: 10.1007/s11263-010-0390-2. URL: http://vision.middlebury. (cit. on p. 34).

[62] N. Parashar and S. Kumar. "Traffic Light Controller System using Optical Flow Estimation". In: *International Journal of Computer Applications Technology and Research* 4.10 (2015), pp. 755–761. ISSN: 2319-8656. URL: www.ijcat.com755 (cit. on pp. 35, 36).

[63] G. Tanner. *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)*. URL: https://ml-explained.com/blog/dbscan-explained (cit. on p. 65).

[64] P. Training. *DBSCAN vs. K-Means: A Guide in Python - Pierian Training*. URL: https://pieriantraining.com/dbscan-vs-kmeans-a-guide-in-python/ (cit. on p. 65).

[65] GeeksforGeeks. *Agglomerative Methods in Machine Learning - GeeksforGeeks*. URL: https://www.geeksforgeeks.org/agglomerative-methods-in-machine-learning/ (cit. on p. 65).

[66] Scikit Learn. *sklearn.cluster.AgglomerativeClustering — scikit-learn 1.3.1 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html (cit. on p. 65).

[67] Scikit Learn. *sklearn.cluster.MeanShift — scikit-learn 1.3.1 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html (cit. on p. 65).

[68] GeeksforGeeks. *ML | Mean-Shift Clustering - GeeksforGeeks*. URL: https://www.geeksforgeeks.org/ml-mean-shift-clustering/ (cit. on p. 65).

[69] A. KHARWAL. *Mean Shift Clustering in Machine Learning | Aman Kharwal*. URL: https://thecleverprogrammer.com/2021/09/29/mean-shift-clustering-in-machine-learning/ (cit. on p. 65).

[70] Scikit Learn. *2.1. Gaussian mixture models — scikit-learn 1.3.1 documentation*. URL: https://scikit-learn.org/stable/modules/mixture.html (cit. on p. 65).

[71] GeeksforGeeks. *Gaussian Mixture Model - GeeksforGeeks*. URL: https://www.geeksforgeeks.org/gaussian-mixture-model/ (cit. on p. 65).

[72] J. Yang et al. "Affinity propagation feature clustering with application to vehicle detection and tracking in road traffic surveillance". In: *Proceedings - IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2010* (2010), pp. 414–419. DOI: 10.1109/AVSS.2010.40 (cit. on p. 65).

[73] Scikit Learn. *sklearn.cluster.AffinityPropagation — scikit-learn 1.3.1 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html (cit. on p. 65).

[74] F. Sutel De Moura and C. T. Nodari. "Application of the Affinity Propagation Clustering Technique to obtain traffic accident clusters at macro, meso, and micro levels". In: () (cit. on p. 65).