Traffic Violation Detection Using Background Subtraction Technique

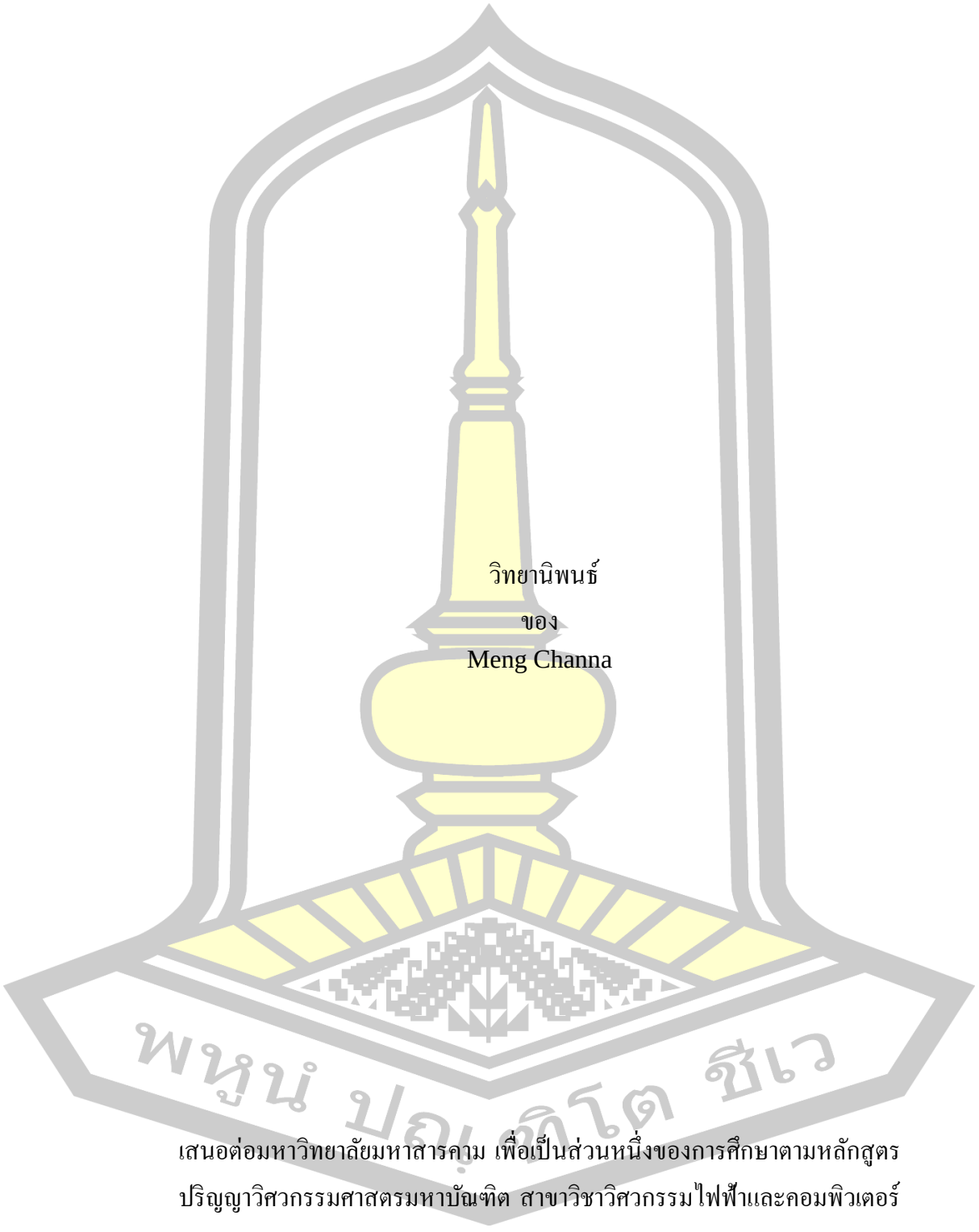
Meng Channa

A Thesis Submitted in Partial Fulfillment of Requirements for
degree of Master of Engineering in Electrical and Computer Engineering
April 2019

พหุน ปณฺ ฑิโต ชีเว

พหุน ปณฺ ฑิโต ชีเว

ระบบตรวจจับรถฝ่าไฟแดงโดยใช้เทคนิคการลบภาพพื้นหลัง



วิทยานิพนธ์

ของ

Meng Channa

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

เมษายน 2562

Traffic Violation Detection Using Background Subtraction Technique

Meng Channa

A Thesis Submitted in Partial Fulfillment of Requirements

for Master of Engineering (Electrical and Computer Engineering)

April 2019

The examining committee has unanimously approved this Thesis, submitted by Ms. Meng Channa , as a partial fulfillment of the requirements for the Master of Engineering Electrical and Computer Engineering at Mahasarakham University

Examining Committee

.....Chairman
(Asst. Prof. Anan Kruesubthaworn , Ph.D.)

.....Advisor
(Asst. Prof. Nattawoot Suwannata , Ph.D.)

.....Committee
(Assoc. Prof. Worawat Sa-Ngiamvibool , Ph.D.)

.....Committee
(Asst. Prof. Niwat Angkawisittpan , Ph.D.)

Mahasarakham University has granted approval to accept this Thesis as a partial fulfillment of the requirements for the Master of Engineering Electrical and Computer Engineering

.....
(Assoc. Prof. Anongrit Kangrang , Ph.D.)
Dean of The Faculty of Engineering

.....
(Asst. Prof. Krit Chaimoon , Ph.D.)
Dean of Graduate School

TITLE Traffic Violation Detection Using Background Subtraction Technique

AUTHOR Meng Channa

ADVISORS Assistant Professor Nattawoot Suwannata , Ph.D.

DEGREE Master of Engineering **MAJOR** Electrical and Computer Engineering

UNIVERSITY Mahasarakham University **YEAR** 2019

ABSTRACT

Thailand is one of the worst countries for traffic accidents - leading to about 24000 fatalities every year. About 1.5% of accidents are caused by Thai drivers ignoring the traffic lights but this causes about 400 deaths annually. This research designed an inexpensive system that was flexible enough to detect the Red Light Runners (RLRs) in poorly marked intersections and send a picture of the RLR to a police mobile phone. The system recognizes the red light and tracks the vehicles, with sizes ranging from bicycles to trucks, that ignore the red light. Gaussian Mixture Models were applied to generate a flexible background model that adapts to changing light conditions. After the background has been subtracted, models of moving vehicles are formed and tracked, initially by template matching. A Kalman filter was added to assist tracking and was shown to be successful when the vehicle merged into the background and disappeared in some frames. This system were tested more than 7300 frames, from videos taken at a variety of intersections, with the red light on, and RLRs were detected. Tracking, when vehicles occlude one another, was a key problem because of the low camera angle used (to keep the system flexible and inexpensive): in 91% of frames, merged vehicles were successfully followed. After capturing the RLRs, the system successfully sent their images to a mobile phone using the cell phone network.

Keyword : Red Light Runners(RLRs), Background Subtraction, Gaussian Mixture Models, Template matching, Kalman Filter, Cell Phone Network

ACKNOWLEDGEMENTS

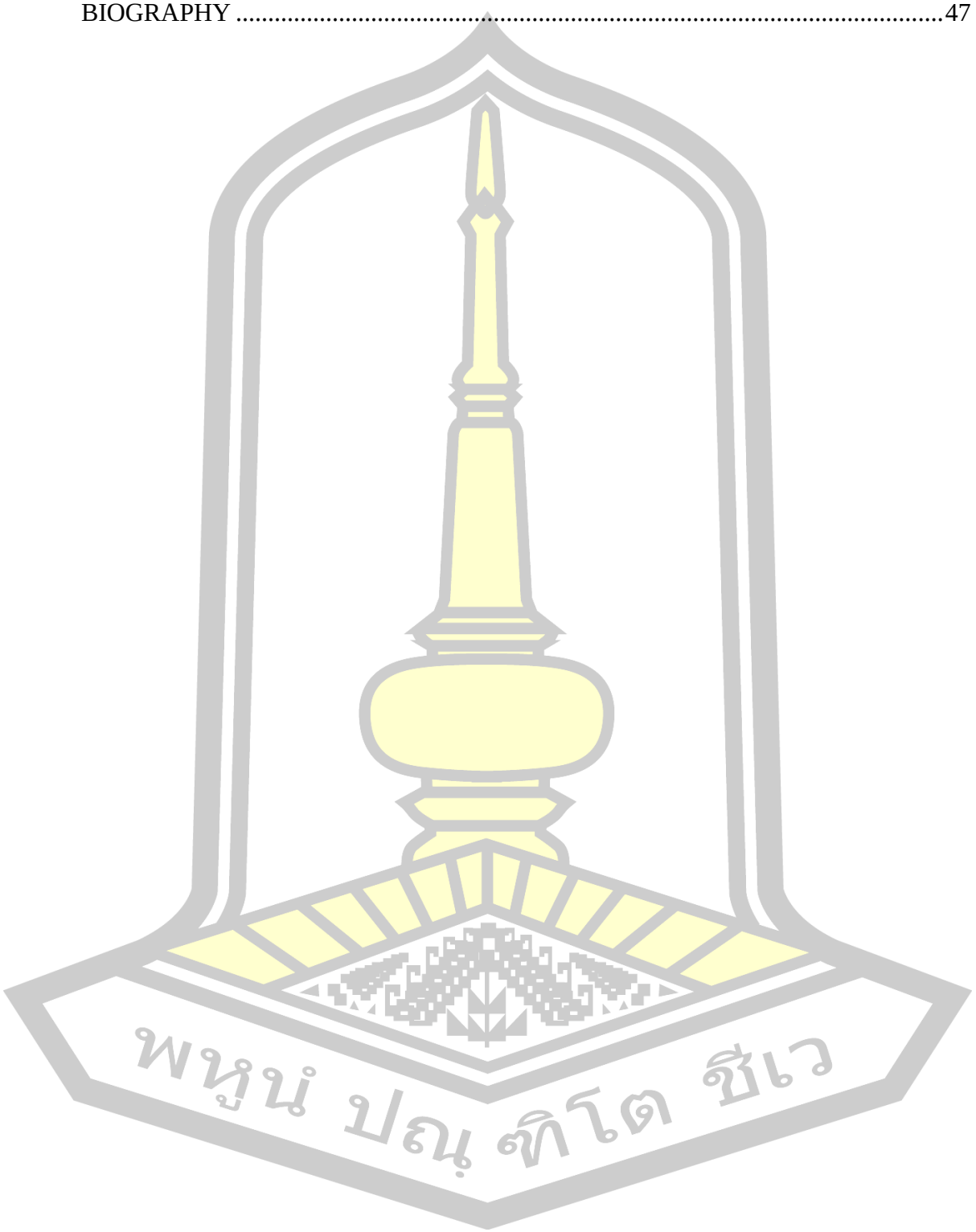
First of all, I would like to thanks to the Princess Maha Chakri Sirindhorn's Education Projects for Cambodia that provided me a good chance to study at the Mahasarakham University in Thailand. I am very appreciate with this values opportunity and promise that I will gain all my knowledge back to develop my country in really soon. I also would like to say thanks to the Mahasarakham University (MSU), Faculty of Engineering that give a chance to study and research related to my interested field.

The most to remember, I would like to special thanks to my adviser Asst. Prof. Dr. Nattawoot Suwannata and Assoc. Prof. John Morris who always supporting me not even in studying, researching but how to spend the life in Thailand. The last but not least, I would like to thanks to all lecturers in Faculty of Engineering that always provide a good advice in studying and researching. I would really missing to all my Thai classmates and staffs in the faculty that always supporting me.

Meng Channa

TABLE OF CONTENTS

Page



List of Tables

Page

List of Figures

Page

Chapter 1 Introduction

I. Overview

Thailand, 510890 square kilometers, 77 provinces, located in the South-East Asia Region, bordering to Laos, Cambodia, Myanmar, and Malaysia[1]. As of 2015, the total population of the country was estimated to be more than 67 million, with a growth rate of 0.34% and a Gross Domestic Product (GDP) per capita of $156000[1]. Observing traffic lights in Thailand sometimes suggests that 'Red' is interpreted as drive a bit faster. A count of one red light interval (only ~ 2 minutes) observed for this study saw 13 vehicles (mostly motorcycles) ignore the red light, which means that Thai drivers are reckless and ignore traffic rules, signs and lights.

Thailand was ranked by the World Health Organization(WHO) as seventh in the world for road fatalities in 2018 [2] at 32.7 per 100,000 with an annual estimate of over 22,400 deaths or 62 deaths every day. The reasons suggested are that cars have become in the financial range of the most Thai people. Therefore, most people learn how to drive from their relatives or friends, which means driving education, has not been set as a priority to Thai drivers. At the same time, there are only a few driving schools that will teach the basic of operating a car and some road signs which cannot be understood by Thai drivers. Therefore, Thais are killed by road casualties more than any other cause year by year with an average around 22,000 persons per year or more than 2 persons per hour[2] which means the country's economy losses about over 100,000 million baht per year or about 12 million baht per hour too[3]. Recent Thai statistics show little improvment[3]. Detailed analysis of these statistics show that 1.5% of fatalities are caused by red-light runners [3]: this factor alone means leads to the loss of about 400 Thai lives each year.

US statistics, with 260,000 crashes and 750 fatalities annually[4], suggest that applying the same ratio, red light runners likely cause roughly 300,000 crashes annually in Thailand. Furthermore, crashes involving red-light runners have a greater impact than other crashes and cause occupant injuries in 45% of crashes, compared to 30% in other crashes[5]. The total cost of road accidents to a community is very high, even in countries with much lower per capita accident rates than Thailand. For example, in Victoria, Australia, with only 5.4 fatalities per 100,000 people, the estimated cost is $ A4 billion per annum[6] in a total population of just 5.8 million (2014 census), thus the cost in Thailand with 17 fatalities per 100,000 [7], the economic cost must be significantly greater. Some causes of the problem include and became the basis of recommendations for legislative improvements [8]:

1. Unclear speed limit laws not a related type of roads (i.e. municipal and city): speeds of more than 50km/h which are unsafe for pedestrians and cyclists are routinely observed. In 2013, speeding was identified as the main factor (12.6%),

2. Drink driving remains a severe problem. WHO recommended that drink driver laws be based on blood alcohol concentration (BAC) at 0.05 g/dl. In this case, Thailand law, setting the alcohol limit at less than 0.05g/dl matches WHO guidelines.

3. Motorcycle helmets. Thai law is strong, but the implementation is weak: drivers pay little attention for physical protection by wearing helmets. Injuries

to the head and neck are the main cause of death, so helmets can reduce the risk of death by almost 40%.

4. Seatbelts, also required by Thai law, can reduce the risk for drivers and front-seat passengers by about 45-50%. However, implementation is weak.

5. Child restraints, known to be important to save children, are not covered by Thai law: young children are allowed to sit in the front and there is no child restraint law based on age, weight, and height.

WHO and Bloomberg summarized the factors that cause Thai road traffic crashes, see Figure 1.

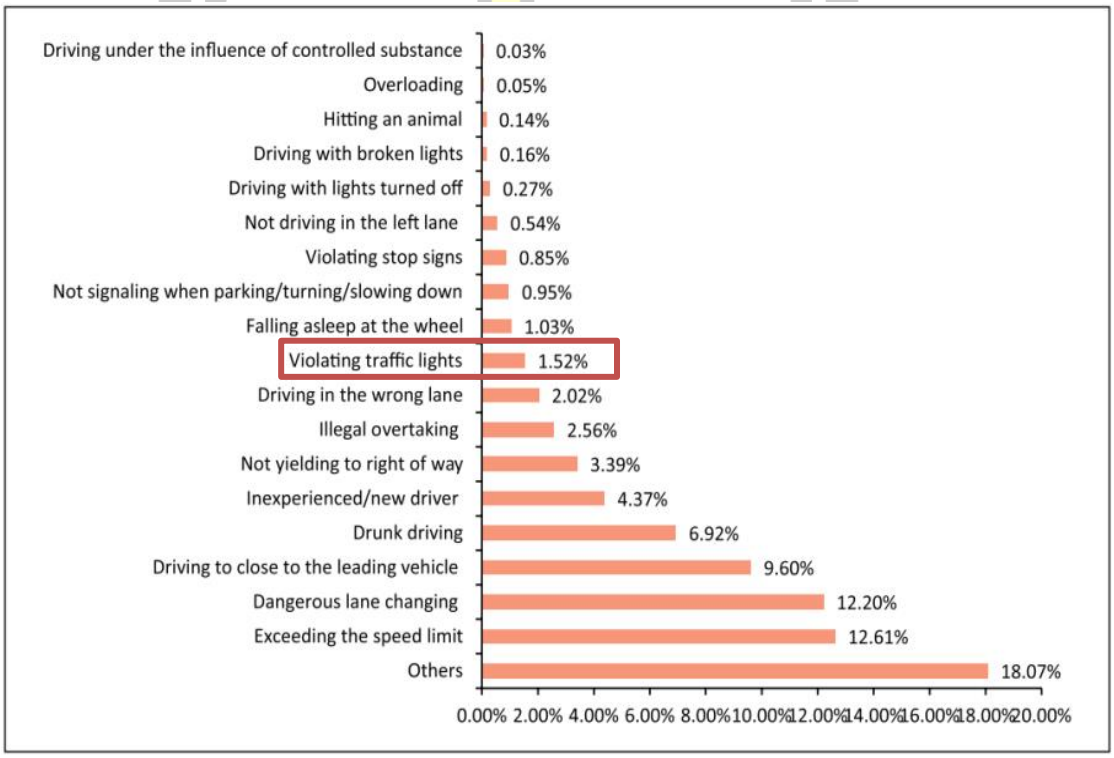


Figure 1 Cause of road traffic accident: source Royal Thai Police, 2016[1]

Although 'Red Light Runners' (RLR - vehicles which do not stop at a red traffic light) are relatively low on the frequency table, they still cause more than 20,000 crashes annually in Thailand which has a significant impact on the economic and social cost. Nowadays, many systems have been developed to detect traffic violations by capturing drivers' actions with cameras [8]. However, most of these are based on fixed installations, monitoring well-marked intersection with links between traffic light controllers and the computer capturing images. However, in Thailand, common with other developing countries, intersections are often poorly marked. Further, police have limited resources to support large complex fixed installations. Thus, to address the social and economic effects of road accidents, there is potential for a system which meets these requirements: it should be

1. Cheap: *i.e.* use cheap, readily available cameras and other components
2. Portable: easy to setup
3. Flexible: not dependent on road markings (or their absence)
4. Work in a variety of scenarios
   2.9 the camera should be able to be positioned within a 20^{o} to 90^{o} to the traffic flow,
   2.9 handle multiple lanes of traffic and
   2.9 not be affected by road-side clutter, *e.g.* random fixed objects, banners, flags *etc*
5. Require minimal changes to current police practices

Thai Police currently use road-blocks as the primary tool for detecting vehicle violations, so the system reported here positions a camera at the monitored traffic light and transmits an image to a road-block that may be 500m further down the road.

Kristan et al. described more than 60 tracking technique in the VOT challenge[9]. Yung and Lai discussed early work with fixed systems attached to traffic light control systems[10]. Several commercial systems are also now available, suited to well-controlled intersections in western countries, where vehicles tend to stay in well-marked lanes, not overtake within lanes, etc. and cameras can be aligned with lanes and monitor vehicles traveling in a single direction, e.g. Jenoptick's Roadstart system, which monitors up to four lanes and relies on road loops or radar[11] and Smart Vision Technology's Smart-RLCS using a camera per lane[12]. They also generally assume a fixed camera positioned meters above the road surface leading to a smaller number of occlusions between vehicles, whereas my aim is to create a small portable system with a camera set 1-2 m from the ground so that it will view many vehicles obscured by others. Previous attempts by Klubsuwan et al., in Thai conditions, detected only ~80% traffic light violations with images alone[13]. Automated traffic light detection is reported by many since the red light is necessarily bright enough to be seen and therefore contrasts well with the background: it works well if the camera is aligned with lanes but becomes a harder problem if the camera may be aligned in an arbitrary direction and several lights appear in its field of view. Previous work in our laboratory used simplistic background subtraction and optical flow to detect and track moving vehicles[14]. Although this generally worked satisfactorily, it was necessary to run several trials to determine acceptable optical flow parameters, object bounding boxes tended to be over-large and multiple merged objects were not well handled. Thus here, I aimed to allow for a wider variety of backgrounds, obtain better object outlines and track multiple overlapping objects. I also allowed a wide range of camera alignments to the traffic flow, implying that several red lights may appear in the camera field of view.

II. Purposes of Research

The key aims of this project are:

1. To reduce traffic accidents
2. To provide a tool to educate drivers to respect traffic rules
3. To reduce the loss of Thai life and
4. To make Thailand's economy grow and
5. To reduce the not trivial social costs of road accidents.

III. Project Importance

To design a system which provides significant benefit from a simple, low budget system, easily affordable in Thailand.

IV. Scope

1. The study used movies of actual traffic violations including over 7300 'Red Light' frames, e.g. frames in which one or more Red Light Runners were observed.
2. Various types of road junctions: 4-way intersections and 'T' intersections
3. Various weather conditions: daylight (with and without wind), evening (low light), after rain (wet and shiny road surface)
4. Various types of vehicle: car, van, truck, and motorcycle

Chapter 2 Literature Review

Object tracking algorithms usually divide the problem into two parts: (a) identification and removal of the static background and (b) tracking objects in the foreground.

I. Background Subtraction

Background subtraction (BGS) aims to extract objects of interest, i.e. the foreground, in an image, by subtracting a background model: we can observe two types of background model, static and dynamic [15]. Historically, BGS algorithms have evolved from early simple algorithms which simply removed pixels which match those in a static background image to adaptive ones that allowed the background to vary during to environmental changes, e.g. changing illumination and others that can remove 'dynamic' features of the background, e.g. moving tree leaves. Problems that need complex background models include

1. Scene illumination changes, e.g. sunlight variation from dawn to dusk
2. Haze in the scene from rain, fog and thermal gradient changes (shimmer)
3. Fixed components of the scene that move with the wind, e.g. tree leaves and flags
4. Shadows (usually tracked with objects but strictly irrelevant and ideally removed)
5. Background camouflage (causing objects to be mistaken for others)
6. Camera jitter
7. Bootstrapping (initially static objects assigned to the background at startup)

Generally, BGS has three steps: (a) background initialization, using a few initial frames to create a background model, (b) foreground extraction, or removing the initial background from the scene, and (c) background maintenance, updating the background model from the current frame. Although apparently a simple process, 'perfect' BGS is not easy and there is extensive literature discussing techniques to solve the problems listed above.

Authors	**Model**	**Description**	**Ref**
Choudhury *et al.*	2016 Survey	Evaluated 11 state-of-the-art algorithms for BGS	2016
Reddy *et al.*	BCCPDI	Block-based classifier cascade with probabilistic decision integration	2013
Wren *et al*	Pfinder	Real-time tracking of the human body	1997
El Baf *et al*	Fuzzy Integral	Fuzzy integral for moving object detection	2006
Maddalena and Petrosino	SOBS	Self-organizing background subtraction	2008
Kaewtrakul	IABMM	An adaptive background mixture model with	2001

pong and Bowden		shadow detection	
Yao and Odobez	Multi-Layer	Multi-layer background subtraction based on color and texture	2007
Hofmann *et al*	PBAS	Pixel-based adaptive segment	2012
Rodriguez and Wohlberg	FPCP	Fast principal component pursuit via alternating minimization	2013
Zhou and Tao	GoDec	Randomized low-rank and sparse matrix decomposition	2011
Barnich and Van Droogenbroeck	Vibe	Visual Background Extractor (VibeGray and VibeRGB)	2011
Subudhi *et al.*	2016 Survey	Local change detection with twelve existing BGS techniques	2016
Stauffer and Grimson	Adaptive Background Mixture Models for Real-Time Tracking	Thresholding the error between an estimate of the image without moving objects and the current image.	1999
Zivkovic	GMM with kNearestNeighbours	Implemented as `BackgroundSubtractionKNN` in OpenCV	2004

Table 1 Background subtraction techniques

Choudhury *et al*. surveyed work up to 2016 and evaluated several key algorithms for videos available in five commonly used video datasets: Wallflower, I2R, Carnegie Mellon, Change Detection and the Background Models Challenge [15]. Important approaches and algorithms are summarized in Table 1 and discussed in the following sections. The papers listed by Choudhury *et al*. might suggest that most of the problems related background subtraction have been solved - except very fast illumination changes. However, BGS work is clearly ongoing and at least one paper was found in February of 2018[16].

One ‘background’ task that is unique to this project is the detection and monitoring of the red traffic light controlling an intersection. It is discussed separately at the end of this section.

1. Basic Models

Early basic models set the first frame as a background model and subtracted it from the current frame to extract the foreground. Then the frame difference idea uses the previous frame instead of the initial frame as a background; because the foreground might be falsely detected as background if it appeared in the initial frame: this scheme adapts to very slow illumination. Further improvements find

the arithmetic mean of pixel values over a temporal sequence and the W^4 system [17], that was used for outdoor human detection and tracking, which attempts to separate people from their background in an outdoor environment using three values - minimum gray value, maximum gray value and maximum intensity difference between two adjacent frames. These are generally unimodal, *ie* a single Gaussian model, and do not work well with uncertain backgrounds [17].

2. Gaussian Mixture Models

Univariate Gaussian Models – with a Gaussian for each RGB channels – can model temporal variations in background intensities but are inadequate for the oscillating background. Stauffer and Grimson, therefore, used Mixed Gaussian Models (GMM) with each pixel represented by a fixed number of Gaussians to follow illumination changes with a learning rate parameter. However, too slow a learning rate failed to handle rapid illumination changes, whereas too fast a learning rate incorporates slow moving objects into the background [18].

Zivkovic and Heijden added a new simple training technique to implement an efficient adaptive GMM model. The data belongs to $\boldsymbol{m}^{th}$ component depends on the mixing weights. For this new automatically selected component idea, GMM produced an effective background model, especially in traffic scenes. It quickly adapts to oscillating background and processing time is reduced [12]. Their approach is readily available in the OpenCV library, **BackgroundSubtractionKNN** method [19]. Because it functioned acceptably for vehicle tracking, it was used as the basis of this work.

2.1. Non-Parametric Models

Kernel Density Estimation (KDE) techniques, extended from default Gaussian distribution, adapt to unknown distributions, but they require sufficient training samples to correctly estimate the density function. The functions are expensive to compute, so several have discussed methods to reduce the kernel bandwidth (number of kernels used) without sacrificing performance: mean-shift models were used by Piccardi and Jan [20] and also Elgammal *et al*[21]. Mittal and Paragios used optical flow to initialize the background model with features modeled with a KDE technique[22]. They showed successfully modeling of the moving trees in a street scenario: these problems were common in videos studied but were avoided by the low camera angle which allowed the moving leaves to be cropped out of the top of each frame.

2.2. Buffer Based Subtraction

Lo and Velastin kept pixel history in a finite First-In-First-Out (FIFO) buffer and compared the current buffer median with the current pixel to determine whether it is a foreground or a new background pixel which is then added to the buffer: the FIFO rule handles overflows[23]. An alternative used the medoid rather than median to make this decision. Linear prediction from the buffer with a Wiener filter has also been used[23]. Wang and Suter describe a buffer consensus approach which was successful on a number of indoor and outdoor scenes with varying illumination levels[24].

2.3. Fuzzy Models

Fuzzy models have been used to decide whether a pixel belongs to the background. Color space choice was an important factor when dealing with camouflage, water surface, waving trees and fast illumination change. Fuzzy models returned some very high scores for specificity in Choudhury's evaluations of El Baf et al's algorithm[25].

2.4. Learning Models

Neural nets have been used to train the underlying density distribution of the pixel sequence and predict the next frame. Culibrk et al's Background modeling Neural Net (BNN) used 124 neurons to alter background model using color and texture detection and improved the handling of gradual illumination variation, sudden illumination variation, shadow and bootstrapping [26]. Yao and Odobez' Multilayer was the only system to produce acceptable results for the time of day variation in Choudhury's evalution[27]. However, neural nets typically require significant amounts of training data[16], so they are difficult to apply to 'fresh' previously unviewed scenes.

2.5. Low-Rank Sparse Decomposition

Zhou and Tao's GoDec decomposed the image into a low-rank background matrix, (the background model), a sparse foreground matrix and a noise component. GoDec performed well on the camouflage data sets where the foreground objects are (deliberately) similar to background ones [28].

2.6. Shadow Removal Model

Shadows generally appear as foreground. Two schemes for identifying them are based on color space features (matching them with the background color) or texture features (matching them with textures in the background). RGB color space based methods have been observed to be noisy at low intensities, and HSI color spaces have been used instead with a median filter updating the existing model. Texture-based techniques match features of a potential shadow to textures already present in the scene. Two-stage algorithms combining both techniques have been used [29]. Hardas *et al.* discussed removal of (slowly moving) shadows in the background. They distinguished two types of shadows: 'self-shadows', part of the object shadowing itself (ie part of the foreground), and 'cast shadows - or moving cast shadows, shadows cast onto the background. A shadow contains low brightness regions, so RGB channels were normalized and then multiplied with a fixed value matrix. If the result was below a threshold that pixel was assigned to a shadow region, otherwise it was marked foreground. Post-processing morphological operations filled in foreground holes [30].

(a) Original Frame (b) BGS method (c) Shadow removal

Figure 2 Shadow removal result [30]

2.7. Post Processing Refinement

A foreground, extracted using a background model, might be incorrect due to a poor initial background model. Post-processing checks for false alarms: steps used include median filtering and morphological operations to improve scattered noise pixels and connected components analysis to reconnect disjointed pixel. These operations can fill up camouflage gaps.

2.8. Bag of features models

Subudhi *et al.* described a bag of features containing three old features - (a) Brightness (BS), (b) Inverse contrast ration (ICR), and (c) Average Sharpness (AS) - and three new) features – (d) Absolute Relative Height (ARH), (e) Local Weighted Variance (LWV) and (f) Integrated Modal Variability(IMV).

1. Brightness (BS): Pixel brightness - the sum of R, G and B channels.
2. Inverse Contrast Ratio (ICR): Variation of pixel intensity from its background: computed as the variance as a fraction of the mean intensity
3. Average Sharpness (AS): Derived from the strength of the boundaries between pixels.
4. Absolute Relative Height (ARH): Variation of the highest peak in a region from the sharpness – computed as the kurtosis.
5. Local weight variance (LWV): LWV attempts to preserve differences between low variation in homogeneous regions and the higher variations of regions with more texture.
6. Integrated Modal Variability (IMV): IMV describes histogram variation. In real images, pixels at any location will vary from one frame to another but the histogram over time will vary less.

Their results were compared with twelve state-of-the-art algorithms. They claim their technique was robust: measured of noise, surface reflection, effects of

uniform shading and lighting changes and verified that it worked well for a variety of scenes, except for camouflaged objects [31].

2.9. Colour Spaces

Sajid and Cheung noted that the human visual system uses rods for low light imaging and cones for color distinction and used color spaces to mimic this: their system includes RGB, YCbCr, HSV, and HIS spaces. YCbCr is suitable for foreground segmentation. For BGS modeling, multiple BG models were used to obtain a better result or non-parametric algorithms with single classifications[32]. Another step is binary classification and model update. Binary Classification includes four sub-processes:

1. Color channels activation/deactivation: Cb and Cr channels status depend on the mean of the image intensity
2. Pixel level probability estimation: the pixel is likely is foreground while the calculation between RGB and YCbCr higher than the BG model color space
3. Megapixel formation: extraction accuracy increased because of the assignment of the same probability to each pixel of the model
4. Average probability estimation and labeling: compute the average probability of the megapixel

Moreover, the model update replaces the old frame, through the passage of the time. After a while, the foreground is identified as a background e.g. people on the chair for a long time, and therefore dynamically determining the suitable changing BG algorithm is much needed. The two linear parameters, model update, and change rate are expressed for the scene e.g if the foreground is not moving, the change rate is close to zero and there no model update if FG is not present in the scene[32].

The universal background subtraction expresses obtained a good performance with any types of conditions especially for the camera jitter, shadow suppression, illumination changes[32].

2.10. Performance Reviews

Jeeva and Sivabalakrishnan assessed BGS techniques including PBAS, SOBS, and ViBe. They found that ViBe was about 25% faster than SOBS and PBAS [33], see

and

.

Xu et al's [34] more extensive survey covered GMM [18], KDE, CodeBook, AGMM, SACON, SOBS, Vibe, and PBAS. Overall, they rated AGMM, SOBS, Vibe and PBAS as the 'most promising' with Vibe exhibiting the highest frame rates for large images closely followed by AGMM. AGMM and Vibe also showed the smallest memory requirements. This supports the decision in this work to use Zivkovic's AGMM. Frame rates for CodeBook were also high, but memory requirements were typically 10× those of other methods.

(a) (b)

Figure 3 (a) Original Image, (b) Pixel based adaptive segmented method [33]

(a) (b)

Figure 4 (a) Original image, (b) Self-Organizing Background Subtraction result [33]

3. Red Light Monitoring

Automatic red light detection is usually straightforward since the red light is, by design, bright enough to be detected easily in most situations. Several techniques have been used, for example, Sooksatra and Kondo[35] used color from the CIELab model and a fast radial symmetry transform. Traffic light detection from a moving (autonomous driving) vehicle was addressed by Guo *et al* [36]: starting in the HSV space, they scanned the scene to detect candidate locations and then used a histogram of gradient features and an SVM approach to detect the light. Diaz *et al* also published a survey – targeting red light detection from moving vehicles, a much more difficult task than my current application see Figure 5, but showing potential for expansion to detect violations from moving vehicles see Figure 6[37].

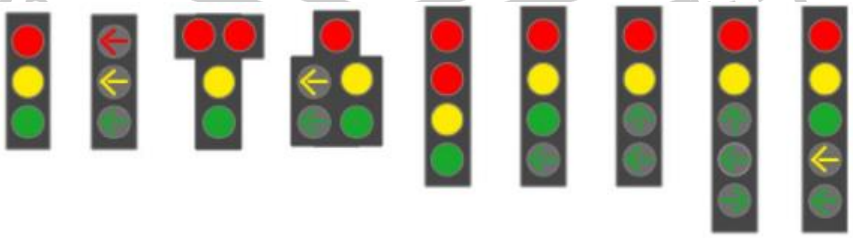


Figure 5 Example traffic light detection [37]

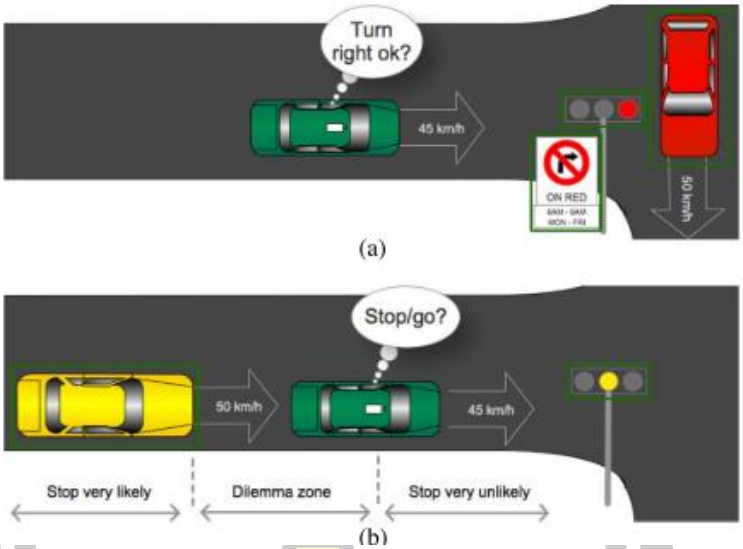


Figure 6 Fused DAS system in intersection scenarios
(a) Turn right on red assistance (b) Dilemma zone assistance [37]

II. Object Tracking

Object tracking is another critical component for object classification in computer vision. It provides the object position based on the previous location in the frame. Again, many techniques have been implemented and improved to track efficiently. An early survey by Yilmaz *et al.* in 2006 [38] has been updated by recent surveys: Smeulders *et al.* in 2013 assessed trackers experimentally [39].

Wu et al [40] evaluated the state of the art of object tracking algorithms based on the factors listed above. The algorithms should track the object again after being blocked and appearing again in the same location. They noted that at the beginning of the tracker is not the same while some algorithms could not be tracked well in early times. Thus, they would like to implement a new evaluation algorithm in order to determine what kind of tracking algorithm was robust to different conditions. On a large scale, they evaluated tracking methods using benchmark experiments, their results showed the tracking ability of each algorithm that can be used to improve a new algorithm for object tracking.

Kristan and a large team of collaborators reported on the VOT2015 challenge for object tracking algorithms [40]: VOT2015 is the third version of the VOT challenge, that compares single-camera, single-target, model-free, causal trackers, applied to short-term tracking; it describes 62 tracking algorithms, twice as many as VOT2014 and contains improvements from VOT2013 and VOT2014 in types of dataset. In VOT2015, the MDNet tracker (classified as a deep convolution neural network) was rated as the best tracker based on its adaptability and robustness (Nam (2016)). However, other trackers showed better performance on individual datasets, which makes the overall assessment quite complex. VOT2016 added a new sub-challenge - VOT-TIR for tracking objects in thermal images[9]. Further enhancements to datasets and evaluations were made in 2017[9].

1. Optical Flow

Optical flow detects object movement and measures accurate object motion based on motion vector techniques. It has been used in many applications. Doyle *et al.* presented a real-time pan/tilt camera object tracking with an optical flow background. Determined 'strong corners' allowed optical flow to classify the background, moving objects or noise lying on the estimated feature location[41]. Background threshold allowed tracking the background motion while moving based on estimated background location; another moving object threshold marks the good feature center. The pan/tilt system selected the previous location if it cannot find the location near the center of the image while the object was not moving. Their technique combined point classification, Kalman filtering, and PZT control to generate simple single tracking, object movement based features changed without background initialized/operator selection or pre-defined capture area showing processing time and a number of features[41].

Kale *et al.* used motion vector estimation of optical flow, object position estimation, to provide object movement information to a result robust to image blur and cluttered background. Optical flow calculated the movement between two images taken at different times by which were useful in filtering in the detection step. After segmentation, a thresholding function allowed them to remove the unwanted objects and they performed a morphological close operation; removed holes in the binary image after thresholding then object blob analysis. For the third step, object tracking using motion vector estimation provides two-dimensional vector of the frame which used to predict between two acceptable frames, but this step time process calculation is not acceptable like dense search accuracy. In this technique conclusion, optical flow and motion vector estimation is very efficiently they concluded that and accuracy is acceptable[42].

2. Kalman Filter

Kalman Filter is an object tracking technique used to predict the future position of an object that has been detected. It estimates a vector calculated from the current position based on a state vector which represents the object behavior.

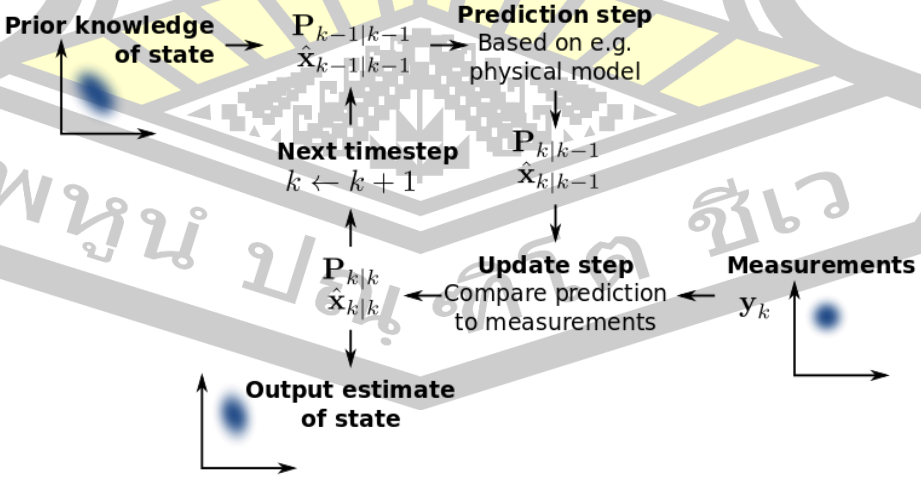


Figure 7 Kalman Filter Basic Steps [43]

1. Multiple Object Tracking method

In “Multi-Object Tracking Via High Accuracy Optical Flow and Finite Set Statistics”, Koch, *et al.* presented a novel method of tracking an unknown number of objects with a single camera system in real time[44]. Their algorithm is based on high-accuracy optical flow and finite set statistics. The main target is treated as a random vector and the number of possible objects as a random number, which has to be estimated correctly. They could also deal with false alarms, clutter and object spawning. In the scene, they proposed a probability model for these events, in order to obtain stable results in the case of missing detections. Additionally, we show how track labeling, based on color and state information, can improve the results. Since the method partly relies on color information, it can handle partial occlusion and is invariant to rotation and scaling. They verified their system on various scences[44].

Li *et al.* discussed the object tracking algorithm using Kalman filters. The system is fully automatic and requires no manual input of any kind for initialization of tracking. Through establishing a Kalman filter motion model with the features centroid and area of moving objects in a single fixed camera monitoring scene, using information obtained by detection to judge whether a merge or split occurred, the calculation of the cost function can be used to solve the problems of correspondence after a split occurred[45]. The measurement position of the object result with Kalman Filter is needed to pass the four steps of KF which are:

1. Process equation
2. Measurement equation
3. Time update equations
4. Measurement update equations

and pass through three modules system state model, feature matching and model update.

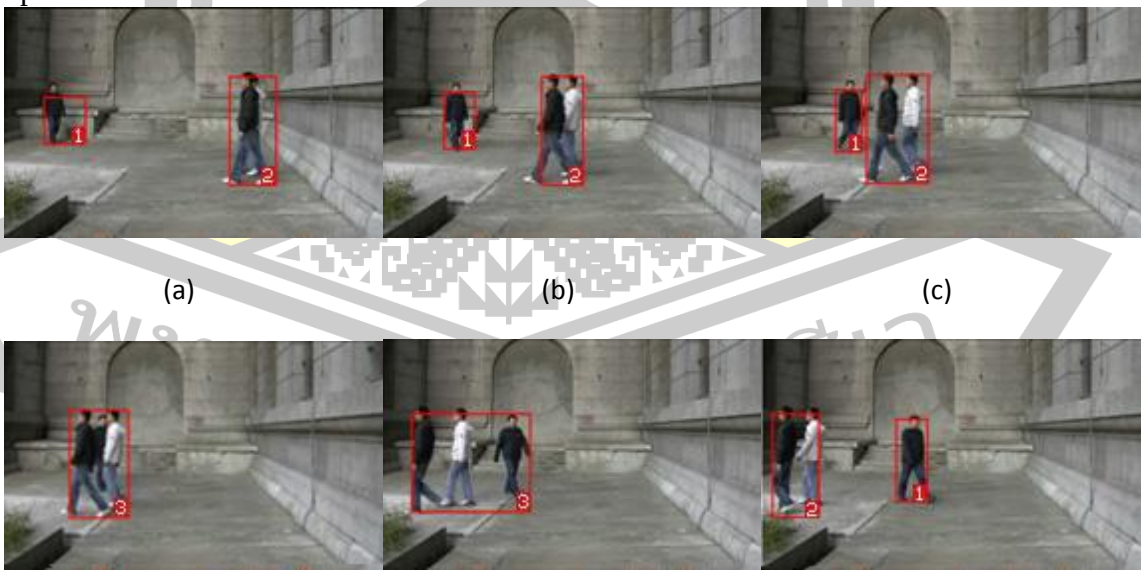


Figure 8 Tracking three people in an outdoor scene A[45]

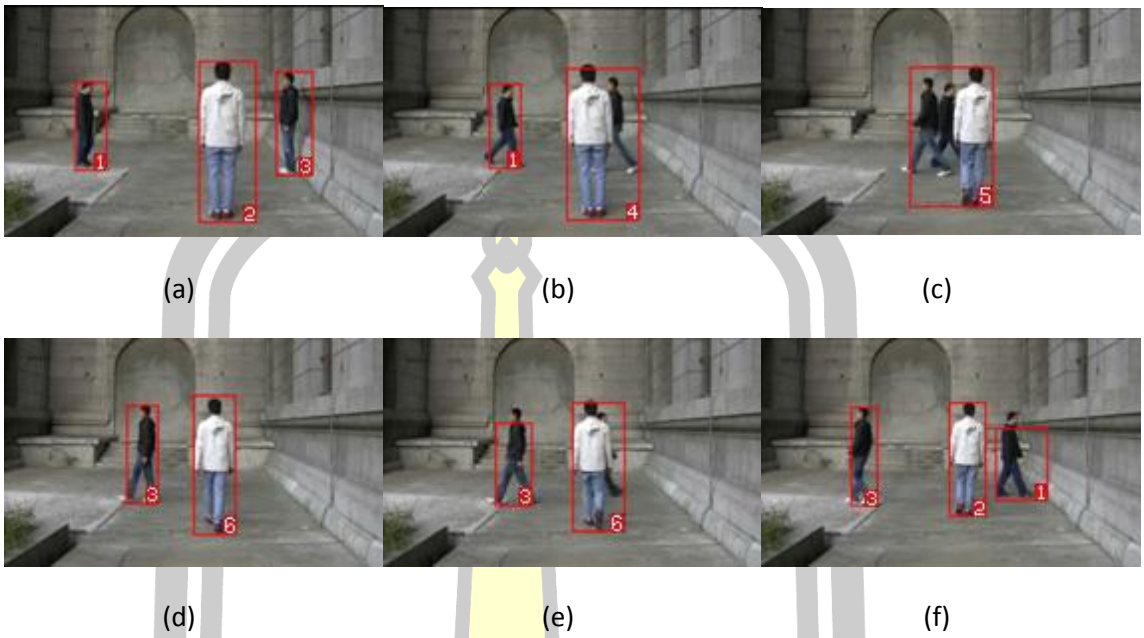


Figure 9 Tracking three people in an outdoor scene B[45]

Figure 8 and 9 show the tracking result from a video of moving people in an indoor and outdoor scene; they correctly tracked while human motion when merged or separated. Vehicle tracking is another issue, speed, to be verified in this research when this KF technique can track the fast moving object in real time shown in Figure 10.

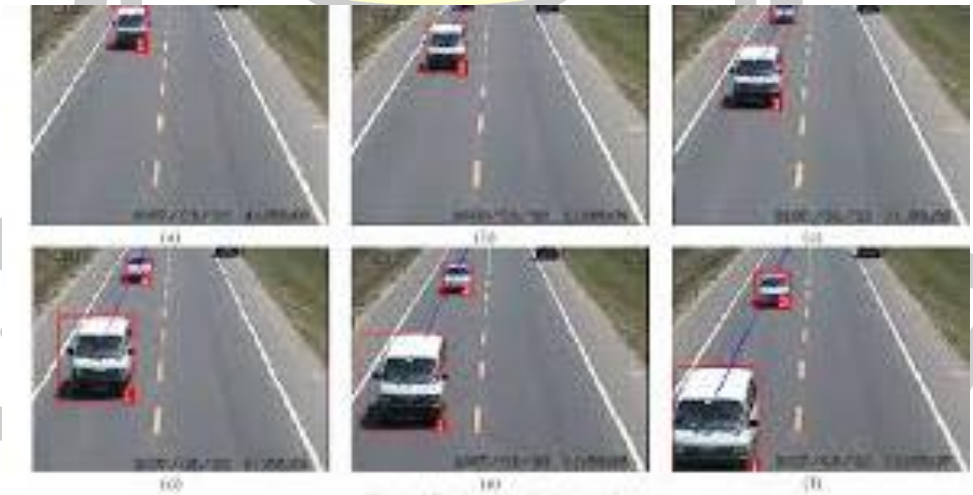


Figure 10 Tracking results for vehicles

2. Kalman filter with partially observed inputs

Su *et al.* discussed the existing Kalman filter estimation of the input object. The Kalman filter is a linear estimation technique for providing the state

estimation of the observed input object. To make a motivation, they marked the Kalman filter properties. After calculating the method of a linear filter, they test the predicted result based on the observed input and then they asymptotic stability of the filter for time stable system. This technique is adaptable in special cases based on the variety of the object information[46].

3. Accurate continue discrete unscented Kalman Filtering

Li *et al.* discussed multi-object tracking where targets and observations need to be matched from frame to frame in a video sequence[45]. Following Figure 7, KF keeps track of the estimated state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements. Denotes the estimate of the system's state at time step k before the k-th measurement has been taking into account; is the corresponding uncertainty. KF contain 2 main processes:

1. Predict Step: KF predicts the position based on the state transition the future state variables.
2. Measurement Step: KF present input measurements and the previously calculated state will produce the measurement state.

Kulikov and Kulikova used an accurate continuous discrete unscented Kalman filter (ACD-UKF), accurate the different equation result which explains the communication of the sigma vector of the unscented. Filtering is calculated by mean and Gaussian formula and scale with local and global error controls. They claimed that this ACD-UKF technique provided the better results for the state estimation and robustly stored errors compared to the fixed step size of unscented Kalman filter[47].

III. Network

1. TCP/IP

The Transmission Control Protocol/ Internet Protocol (TCP/IP) was developed in the late 1960s by the US Department of Defense [48] and has become the standard protocol on the Internet and very widely used. It creates a virtual link between a sender and a receiver and can use a wide variety of underlying physical media ranging from copper wires to links using cell phone circuits. The sending data appears to be transmitted in a character-by-character fashion, rather than as discreet packets [49]. In addition, each TCP/IP host is identified by a logical IP address, which is a network layer address and has no dependence on the data-link layer address (such as MAC address of a network adapter); a unique IP address is required for each host and network component that communication using TCP/IP [49].

TCP/IP connections should be reliable as checksums are attached to each packet and retry protocols in the lower network layers recover lost or corrupted packets. However, in practice, using cell phone connections, we observed many lost packets and implemented a simple retry scheme to ensure that each image was transferred correctly (see Section V).

2. Cellphone Network

The GSM (Global System for Mobile Communication) network is a global standard for mobile communication [50]. Janecek *et al.* discussed the infrastructure of a mobile cellular network as composed of a radio access network (RAN) and a core network (CN) [51]. In addition, Janecek *et al.*also noted that each mobile device can be in an active and an idle state allowing voice calls or data transfers anytime [51].

IV. Application

In Google's official blog, they described the innovation of Android's mobile phone operating system (OS). With free open-source OS, developers can implement a new software for a lower price smartphone for several mobile platforms. Google Play Store now contains over one million apps[52].

1. Java and Native Language in Android Application

Android is a mobile operating system implemented by Google, The applications in Android were written using the Java Software Development Kit (SDK), C++ and recently announced Kotlin programming for Artificial Intelligence (AI) e.g. face recognition, finger scanning.

Android Studio, Android's official IDE, was announced at Google I/O 2017. It is a large update focused on accelerating app development on Android[53]. It is purpose built for Android to accelerate your development and help build high-quality apps for every Android device. Android Studio contains one or more modules with source code files and resource files[53]. Types of modules included:

1. Android app modules
2. Library module
3. Google App Engine modules

Android will display the project file shown in Figure 11. By default, it contains the following folders:

1. manifests: contains the AndroidManifest.xml
2. java: Contains the Java code files, include JUnit test code
3. res: Contains all non-code, such as XML layout, images, UI strings

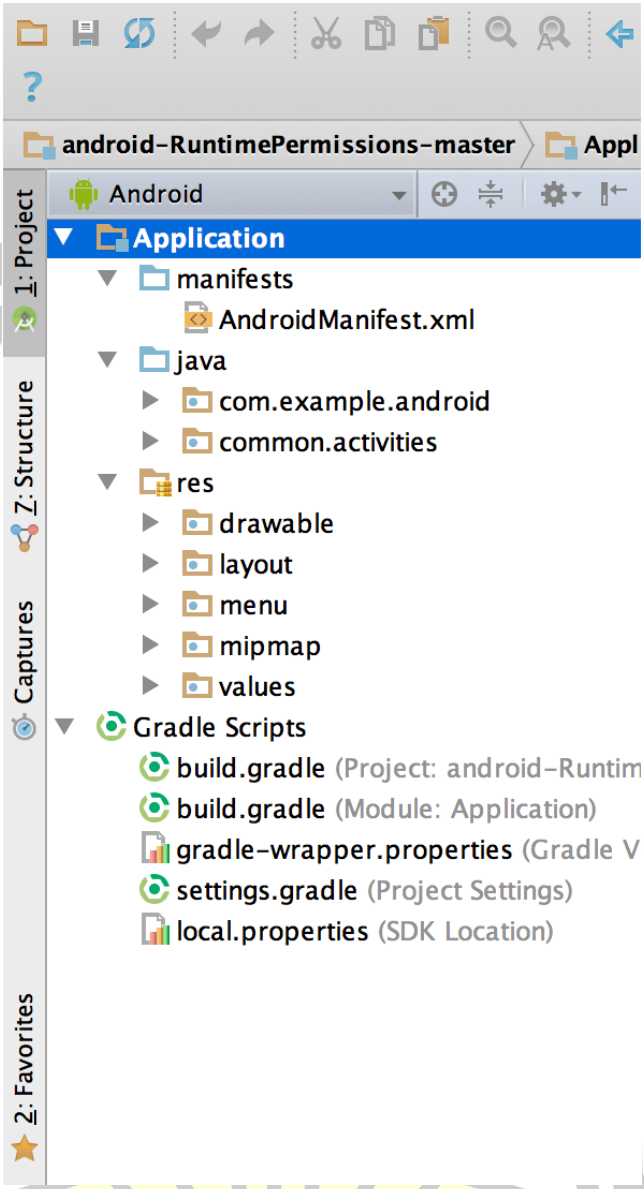


Figure 11 Default project files (in Android) [54]

Chapter 3 Methodology

I. Data acquisition

My system consists of a single camera attached to a PC and positioned to monitor a controlled intersection. It monitors the traffic lights themselves and does not require assistance from traffic control hardware or road loops. When it detects a red light runner, it captures images of the offending vehicle and transmits a set of images to a police post, which can be several hundred meters down the road. The system components are inexpensive: see a cost estimate in

.

Installed components		Est costs ($US)
Intersection	Camera	300 – 500
	Tripod	500
	Laptop	1000
	Network interface	300
	Battery + inverter	600
Monitoring station	Smartphone	500 – 1000
	Optional Laptop	1000
Total		< 5000

Table 2 Costs

The software is rule-based: it builds models of all moving vehicles in the processed region and uses multiple rules to match models from one frame to the next and thus track vehicles. Vehicles in opposite lanes (which may not be red light runners) are tracked because their templates interfere with foreground and the models of both are updated accordingly to allow tracking through such merges.

II. Operator Setup

Since there is only a single camera attached to a PC and it monitors the traffic light themselves without assistance from other hardware e,g. road loops, set up at the intersection is simple. When it detects a Red Light Runner, images of the offending vehicle are transmitted to a mobile phone to a police post, which can be several hundred meters away – as shown in Figure 12. Figure 12 shows the range of positions of the camera on a complex controlled intersection.

1. Link to Monitor Post

First, the operator checks the link between the intersection camera and the monitoring post computer. It is assumed that an available cell-phone network including an internet-capable cell-phone network can be used at each end. Then the camera is placed so that it has a clear view of one of the controlling traffic lights: an example, see Figure 13. Only the portion of the image containing the red light runner needs to be transmitted to a police post, which can be a several hundred meters down to the road, in tests reported here, useable ~300KB images were obtained from a 2Mpixel camera.

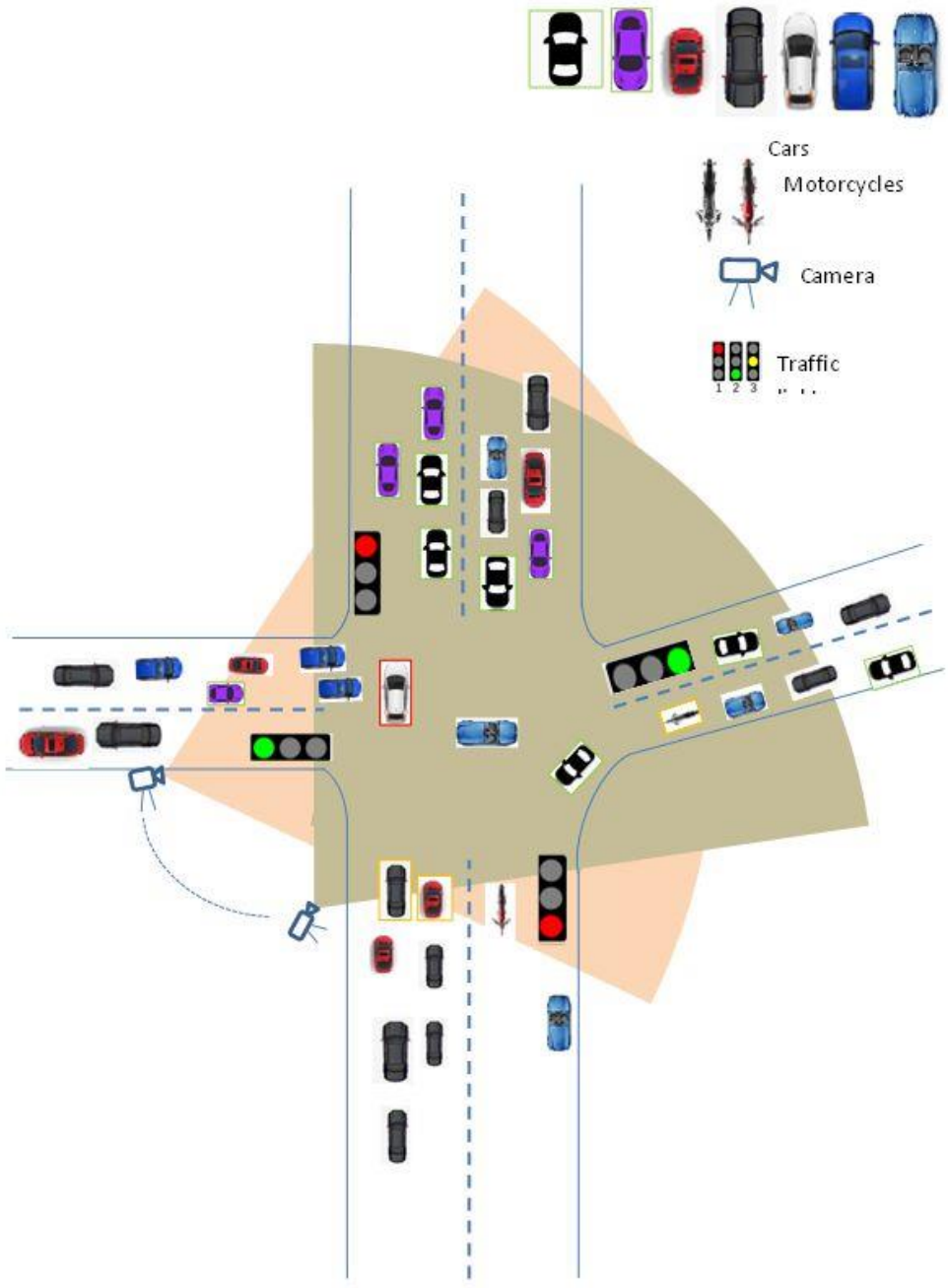


Figure 12 Overview of a system at a controlled intersection showing the range of positions of the camera. The solid regions show the camera field of view at the extremes of the camera position.

2. Camera Setup

Given the variety of the intersection configurations to be monitored, I allow the operator to position the camera flexibly: he or she only needs to choose a position where the camera's field of view covers

a) one of the controlling red lights,
b) the 'stop line' (position at which vehicles must stop in front of a red light) and
c) Sufficient area of the intersection itself so that moving vehicles can be detected and tracked.

Then the operator is asked to manually delineate the forbidden region, i.e. the area into which a vehicle should not move when the light is red. This is fast and not particularly critical. Operators can choose to assist delineation of the forbidden region with markers, e.g. cones, tapes, temporary road marks etc., which are placed before the control program is started. The operator can check and adjust, if necessary, the marked forbidden region. The forbidden region is then expanded automatically to include an area, the ***processed region***, which includes vehicles that may enter or leave the forbidden region soon. This increases the probability that an accurate template of any vehicle is captured before it merges with other vehicles – sample regions are seen in Figure 13. In the horizontal direction, the expansion factor is sufficient to include a complete bounding box of a large vehicle – enabling a vehicle to be tracked just before it enters the forbidden region. In the vertical direction, it is based on the height of a large vehicle at any point in the forbidden region – remembering that the forbidden region defines an area on the road surface.

Limiting the processed region reduces but does not eliminate interference from overhanging trees, flags, flexible advertising, etc. This is discussed later in Section IV.

Then the program will identify or select red lights (when they appear) and accept them with a single operator key-stroke. Then the program moves into automatic mode – identifying and capturing red light runners until it is stopped.

III. Red light detection

Before the system is executed, the red light detection algorithm will loop until the operator marks a suitable red light: The program steps are:

1. Identify 'red' areas by comparing high intensities in the red channel of the image with intensities in the other two channels. Typical traffic scenes contain large white areas, which show high red intensities, so these are eliminated because the blue and green intensities are similar to the red ones. Unfortunately, scenes often contain several red patches (red cars seem to be very popular), so I did not attempt to fully automatically identify lights: previous efforts, e.g. Sooksatra and Kondo [35], found less than 90% correct detections in fairly ideal conditions, as might be expected, due to the likelihood of round and red objects appearing in a typical scene.
2. Select red light candidates by shape and size.
3. Present candidates to the operator, one-by-one

4. Terminate when the operator accepts a candidate
5. Go to the next frame

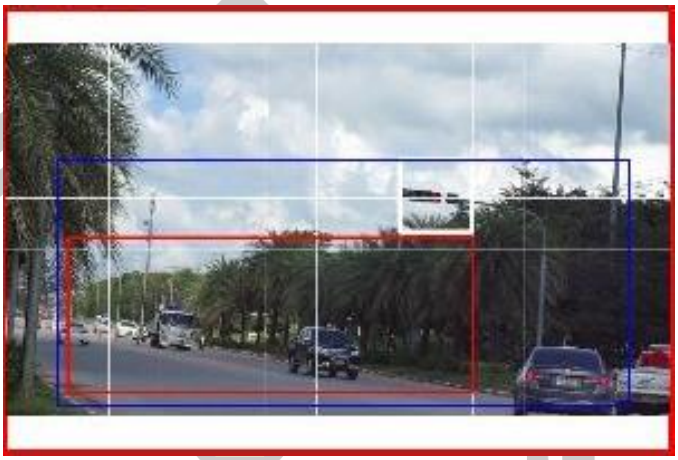


Figure 13 Red Light Detection: Red light detected in the white box. The operator clicks once to verify that this is the controlling light. Note that the red box marks the ***forbidden region***: no vehicle in the left lane should be moving in this area when the light is red. The blue box is the automatically added ***processed region***: vehicles are tracked in this area to capture templates of vehicles which may soon enter the ***forbidden region***.

IV. Vehicle Detection and Tracking

The ***processed region*** reduces computation time significantly because the area in which vehicles can be observed typically has a width: height ratio significantly higher than the typical 4:3 camera ratio. It prevents the system from tracking many large, moving, but irrelevant objects such as large tree branches and flags by the roadside. In typical intersections, the background is reasonably stable with respect to visible structures, so it is usually simple to segment the scene into background and foreground.

1. Background Subtraction

In the problem studied here, shadows and lighting generally very slowly during the day with small numbers of low-level dynamic variations from tree, flags etc. The background model in this research fundamentally used a *Gaussian Mixture Model* (GMM) following Stauffer and Grimson's original work [43], enhanced by Zivkovic [26] and der Heijden's algorithm (openCV *BackgroundSubtractionKNN* [19]) was used to build background. This tackles some problems with shadows, illumination changes and varying background [55] – see Figure 14. Waving trees and flags remain as occasional problems [56] but these unwanted moving objects can be trivially ignored by an operator who will not try to issue a traffic violation to a tree or a flag.

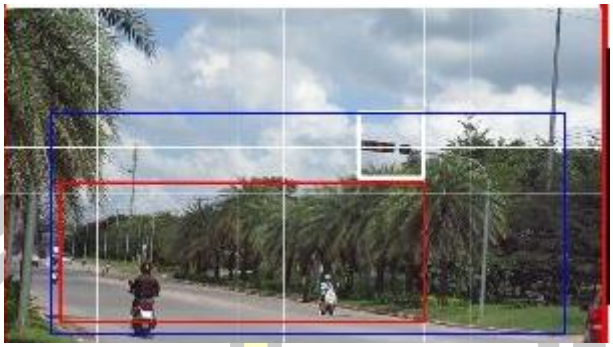


Figure 14 Original frame before Background Subtraction

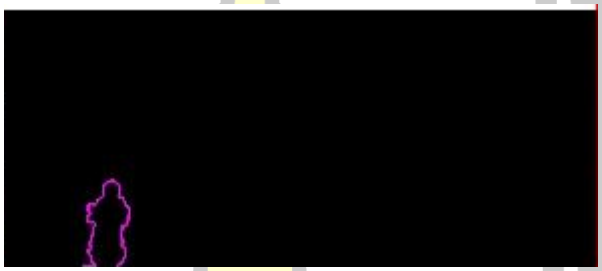


Figure 15 Contour around background subtraction mask outlining moving vehicle

Figure 16 Bounding box formed around moving the vehicle

Figure 17 Cropped image of the Red Light Runner (RLR)

The OpenCV ***backgroundSubtractionKNN*** class segments the foreground and background models and is potentially able to construct a usable background model even if moving vehicles appear in the scene, but, in practice, when the scene included a moving vehicle, large 'ghosts' of the moving vehicle remained in the background and caused matching problems, so it became necessary to select sequences of more than seven vehicle-free scenes to build the background model. When the light was green, whenever possible, the software selected scenes with no moving objects and used them to update the background using options available in the OpenCV routine.

We also tested ***backgroundSutbractionKNN*** by deliberately injecting significant movements of the camera on its tripod – because large fast-moving vehicles generate strong wind gusts able to disturb the camera and thus the background. However, movements of large blocks of leaves were not incorporated correctly into the background model and were processed separately.

2. Tracking with Kalman Filter

Figures 4 to 7 show the steps in the detection of a single moving vehicle. Firstly, the previously modeled background was subtracted from the current frame leading to a foreground mask. The binary foreground masks (there may be several moving objects) were then contoured to identify moving objects large enough to be vehicles. A model, ***Vj***, for each potential vehicle was constructed containing its

1. position
2. bounding box (derived from a foreground mask contour)
3. direction and velocity (updated after a vehicle has been matched in second and subsequent frames)
4. template (cut from the original image using bounding box)
5. contour moments up to 1^{st} order.

The tracking algorithm steps are:

First set the controlling parameter settings:

1. Set $d_{\boldsymbol{thresh}}$ based on an estimate of the maximum distance (at normal traffic speeds) that a vehicle can move, in pixels, in one frame time: a value of 50 pixels was used for most videos.

2. Set a_r, an acceptable contour area change from frame to frame: based on the difference between a car and a motorcycle. This ensures that a change caused by a motorcycle merging with a larger vehicle is detected.

   For each frame

   1. Compute contours on the binary mask,
   2. Select contours (a) large enough to be vehicles *and* (b) within the *processed region*,
   3. Compute a bounding box,
   4. Form a template from the original RGB frame within the bounding box,

3. For the first frame - create a model for each potential vehicle
4. For subsequent frames - find the *best match* for each vehicle in the previous frame with potential vehicles in the current frame *or* create a new model for newly entering vehicles

The *best match* function incorporates a number of rules to deal with complex scenarios when vehicles *merge*, *i.e.* obscure another and appear as part of a single contour, or *split*, *i.e.* separate and appear as two or more contours. Thus we have two lists, $\{V^{prev} \mid j = 0,n-1\}$ and $\{V^{cur} \mid k = 0,m-1\}$, where n and m are not always identical, due to

1. vehicles leaving or entering the *processed region*
2. vehicles obscuring one another and *merging* or
3. vehicles separating, when one contour *splits* into several

The algorithm first checks each vehicle in the previous list, $\{V^{prev} \mid j = 0,n-1\}$ to find a potential match in the current list, using these rules:

for $\boldsymbol{j \in 0 \rightarrow n-1}$
  for $\boldsymbol{k \in 0 \rightarrow m-1}$
    match V^{prev} with V^{curr}
    Compute $\boldsymbol{d_{md}}$ = best template match $V_j^{prev}.template$ with $V_j^{curr}.template$
    using matchTemplate [57] and
    set $V_j^{curr}.pos = V_j^{prev}.pos + d_{md}$;
    update V_K^{curr} from V_j^{prev}
    if $ist(V_k^{curr}.pos, V_j^{prv}.pos) < d_{thresh}$
      match: set $V_k^{curr}.matched = j$
      update V_k^{curr}
      break
    else $V_j^{prev}.area < V_j^{curr}.area$
      assume split j and k
    if $a_r V_j^{prev}.area < V_j^{curr}.area$,
      assume split j and k and
      create new vehicle structure
Remove leaves masquerading as vehicles from list
  if $(V_k^{curr}.pos.y > \max(V_j^{prv}.pos.y) + h_{thresh})$ and dominantly green,
  remove $V_k^{curr}\ from\ \{V^{curr}\}$

When a vehicle obscures one behind it (a ***merge***),

1. identify the foreground vehicle from its template match,
2. remove the area inside its contour from the template of the vehicle behind and
3. match this (smaller) template back to templates from the previous frame.

This allows us to track background vehicles until they are almost fully obscured.

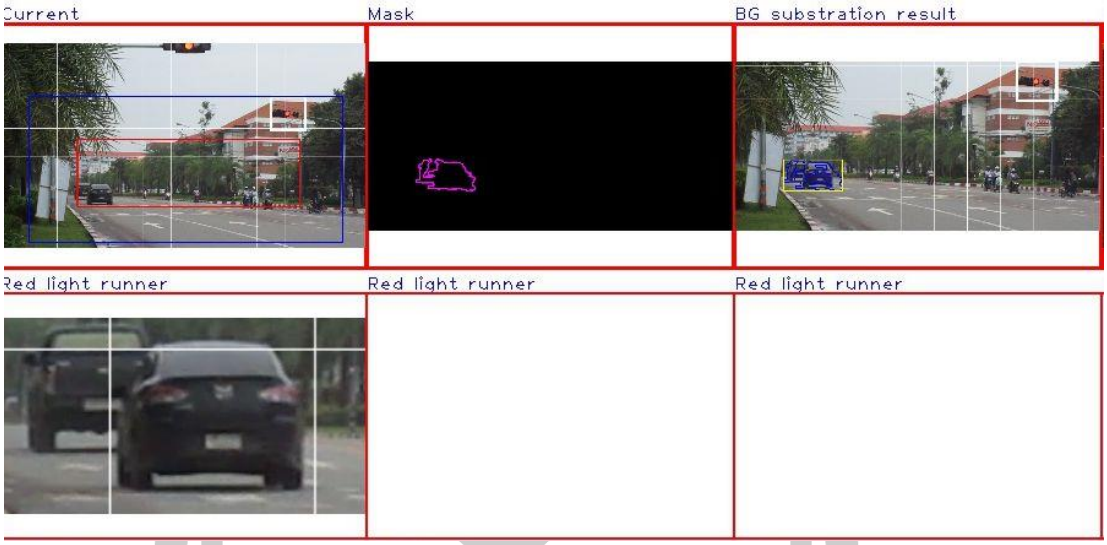


Figure 18 Detecting red light runners at intersection (16.23585N, 103.2638E)
Top row: marked up intersection image - forbidden region: red,
Processed region: blue; contoured foreground mask after background subtraction;
Red light runner outline;
Bottom row: image of two detected red light runners to be transmitted to the monitor station.

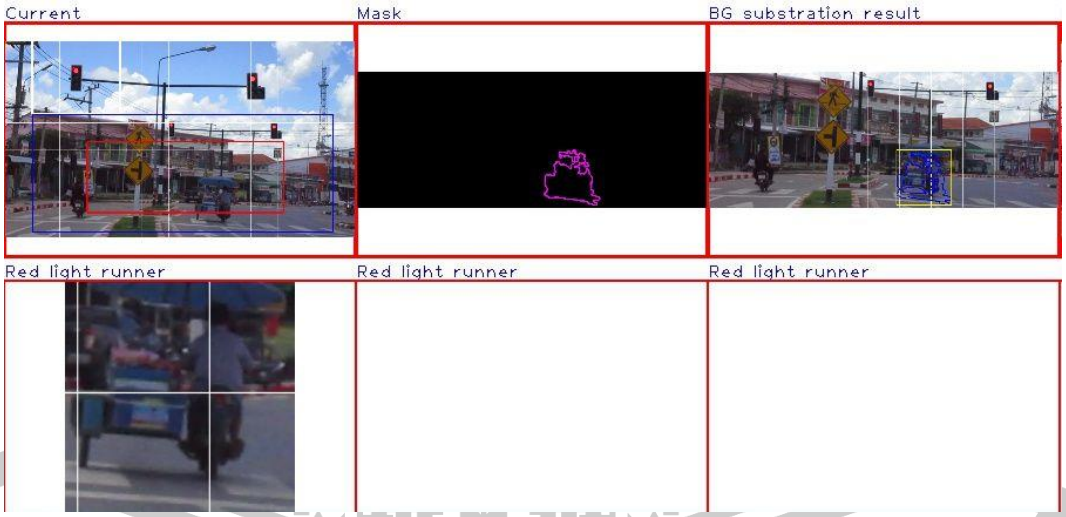


Figure 19 Detecting a red light runner
Top row: marked up intersection image - forbidden region: red, processed region: blue; contoured foreground mask after background subtraction; red light runner outline;
Bottom row: the image of detected red light runners to be transmitted to the monitor station.

Figure 18 Detecting red light runners at intersection (16.23585N, 103.2638E)Figure 18 and Figure 19 show vehicles moving forward while the light is red. The camera used was a consumer camera with 1280 × 720 pixel resolution (Canon IXUS 155); angle of view to the traffic flow direction ∼0◦: the camera was on

a tripod placed on the island between opposite lanes. The bottom row of each image shows the images of the red light runners sent to the monitor station. In Figure 18, one runner followed closely behind the leader and could not be separated. However, a composite image (Figure 18 lower left) was transmitted allowing both runners to be stopped at the monitor station.

In Thailand, red light running is extremely common and often practiced by teams – particularly of motorcycles although larger vehicles often participate. Thus the ability to handle multiple runners automatically in one event was essential. The system built here handles light change events fully automatically and requires no manual input for initialization of tracking. Once a vehicle is detected moving in the forbidden area, it will be tracked while it is still visible in subsequent frames.

The steps for adding the Kalman filter to each vehicle model follow. After the moving vehicle has been recognized by background subtraction, a list of KF states is created:

1. Add the position of the bounding box to the KF list in the first frame of each new vehicle
2. First frame
    1. Add a new position to KF list
    2. Add KF list to a vectors
    3. Set measurement list position
3. Second frame
    3.1. If object found, add a new position to KF list
    3.2. If the position of the current vehicle does not show up
        3.2.1. Get the position from measurement list instead
        3.2.2. Check if the new position contain in the measurement list
            1. If it contains, adds a new position to KF list
            2. If it does not contain, update the position in KF list

An example of the use of the Kalman filter to track a ‘lost’ vehicle is in the next chapter.

V. Applied to Mobile Phone

Now, the system is being tested using IP links over the cell-phone network, which may prove satisfactory in the long run: a Red Light Runner traveling at 60km/hour needs the 30s to reach a monitoring station 500m down the road and a communications latency of much less than that is readily achieved. A frame rate as low as 10fps requires a bandwidth of ~0.25MB/s to capture 30 frames of a red light runner at 60 km/hour: so a clear picture can still be obtained if any frames are dropped. There is also considerable potential to use higher resolution images but seemingly no need to increase frame rates.

1) System flexibility

Note that automatic number plate recognition would require higher resolution and also restrict the position of the camera to be almost directly behind the moving vehicle. In contrast, in this system, there are only weak constraints: it can be rapidly set up and only needs a clear view of the intersection and a controlling traffic light, so can be set up in any convenient and safe location: the camera height can also vary from just above the traffic level – about 1.2m from the ground was used in the tests reported here, but this is not critical. The first aim of this system would be driver training using police in the monitoring station rather than enforcement, which will probably only become an issue after drivers start mostly stopping. Currently, Figure 22 appearing at the monitoring station receives images, clearly identifying the Red Light Runner, enabling the police to decide appropriate action.

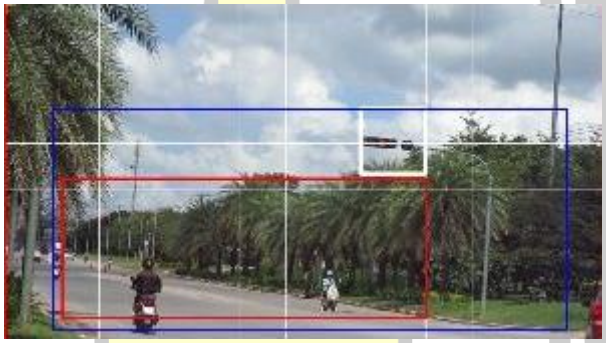


Figure 20 Image captured by the PC attached to the camera at the intersection

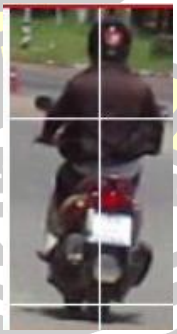


Figure 21 Red Light Runner image extracted in the PC attached to the camera at the intersection

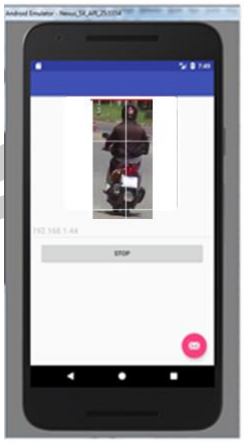


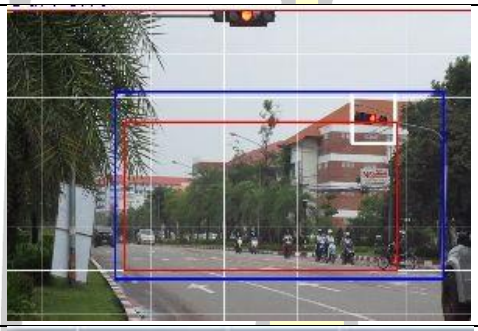
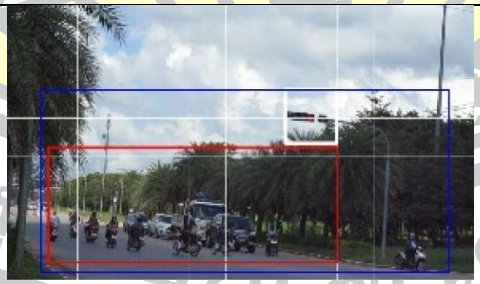
Figure 22 Red Light Runner appearing in mobile phone at the monitor station

Chapter 4 Result

I. Vehicle Detection

In this chapter, results from all the acquired videos are presented and discussed with particular emphasis on situations in which occlusions provide challenges for the basic template matching algorithm which tracks single vehicles.

a) Intersection Locations

Label Location	View	Description	View angle
A 16:24177ºN, 103:25321ºE		Mahasarakham university's main road, 2 lanes in each direction plus side road turning to the university on the right side, well-marked road	10º
B 16:24177ºN, 103:25321ºE		The minor road leading out of Mahasarakham city, 2 lanes in each direction + side road, almost no road markings. Note the forbidden area stops short to the left: this avoids tracking vehicles which entered on the right before the light turned red.	50º
C 16:24648ºN, 103:24590ºE		The main road in front of Mahasarakham University campus, 2 lanes in each direction + side road entering campus, well-marked road	10º
D 16:23585ºN, 103:26380ºE		The road through shopping center near campus, 2 lanes in each direction + 2 side lanes, angled smaller road entering from	0º

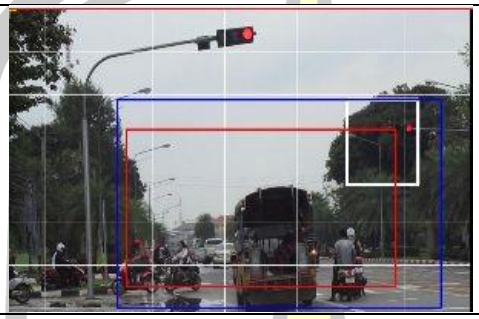
		left, new well-marked, multiple traffic lights	
E 16:24177ºN, 103:25321ºE		Mahasarakham university's main road, 2 lanes in each direction plus side road turning to the university on the left side, well-marked road	0^{o}

Table 3 Details of intersections used in tests

Video Location	Frame Count	Vehicle Tracked	RLR		Events		Tracking Errors	Error (%)
			GT	Imaged	Merge	Split		
A1	237	13	2	2	5	9	17	7.2
A2	278	7	4	3	2	3	6	2
A3	276	11	2	2	2	5	34	12
B	492	13	-	-	12	2	1	0.2
C1	366	22	2	0		11	38	10
C2	399	17	0	0	20	11	38	1.5
C3	398	24	2	2	12	13	25	6.3
C4	398	21	1	1	12	9	46	12
C5	398	23	1	1	14	17	67	17
D1	699	21	0	0	6	7	32	4.6
D2	809	22	4	4	22	20	53	9
D3	809	28	4	4	9	7	97	12
E1	590	37	0	0	40	33	77	13
E2	590	38	0	0	18	18	58	9.8
E3	591	41	0	0	27	20	62	10.5

Table 4 Summary of tracking errors from 7330 frames in 15 videos

b) Tracking error counts

Table 4 shows the overall counts for vehicles tracked, Red Light Runners detected, events, i.e. situations in which vehicles occluded one another and errors. Only frames for which the light was red were processed and counted, so counts

in the ‘Frame Counts’ column do not include frames when the light was amber or green. Errors were counted for each object in each frame, i.e. a vehicle which was not correctly tracked for n frames was counted as n errors and represent total tracking errors and not the much lower counts for incorrectly tracked vehicles discussed in Chapter 3. Results presented here focus on the correct tracking of vehicles - including some vehicle observed and tracked traveling in the opposite direction, which would not be classified as red light runners and reported to the monitoring station. No vehicles failed to be detected: errors were caused when overlaps between vehicles led to groups of vehicles (sometimes as many as four) being temporarily tracked as one. The vehicles were tracked completely in more than 83% of the frames in which they appeared. However, a much higher fraction of red light runners was identified and images captured - approaching 100% for single event red light runners (with only one failure due to inexperience in positioning the camera to properly view right turning vehicles).

The tracking error counts in the final column show the good results from this system. Remember that these are per frame tracking errors and routinely a single tracking error does not cause a RLR to be missed – it will be detected correctly in other frames: an RLR will appear in the forbidden region for typically 50 or more frames (several seconds at 15fps), thus the RLR detection rate approaches 100% even with these tracking error rates. They are strictly worst case numbers and were reported as part of an extension of this project to detect other violations like illegal lane changing.

Tracking individual vehicles, i.e. ones which did not overlap with others, caused few problems: background subtraction of vehicles close to the camera is accurate because in this environment the background was relatively stable and a reliable model was obtained. However, the low camera angle caused many vehicles to appear overlapped or occluded by others and required special processing and, as might be expected, led to some situations where tracking is not accurate. These situations may be classified as (a) merges or (b) splits and are discussed in the next sections.

1. Merge Condition

Merges occur when one vehicle starts to occlude another going in the same or opposite direction. This is first recognized when the size of the vehicle appears to be 3 times larger than it was in the previous frame.

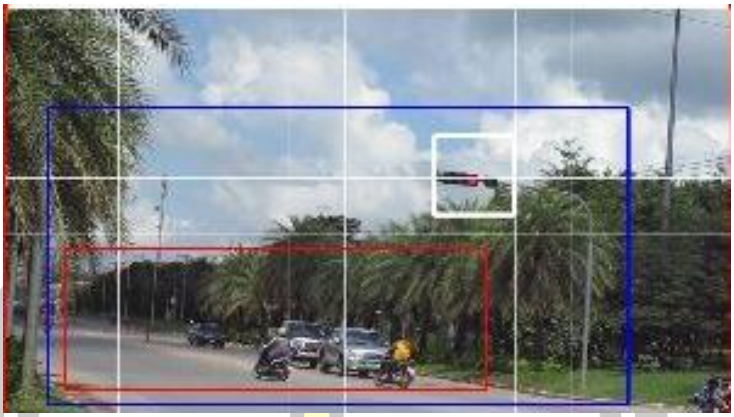


Figure 23Original Frame (frame 4526)
Marked up intersection image
Forbidden area – red color,
Processed area – blue color

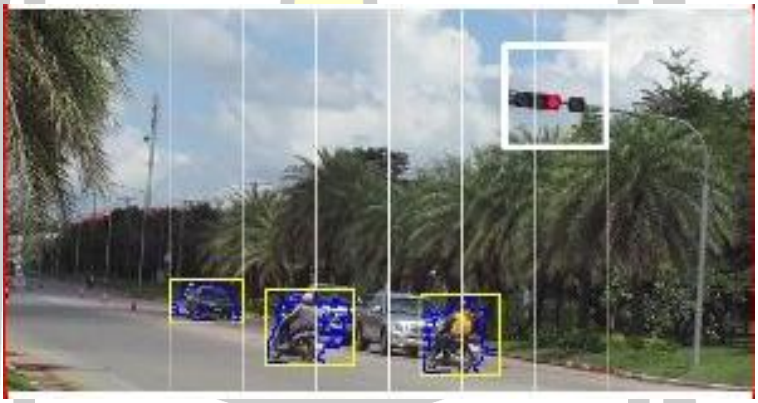


Figure 24Moving vehicles detected after background subtraction

Figure 25 Vehicles detected and images to be sent to the monitor station

Figure 26 some frames later than the previous frame in Figure 24- Original Frame (Frame number4532)

Figure 27 left motorcycle merged with the black car behind it; car behind the yellow motorcycle has stopped and become part of the background

Figure 28 both motorcycles detected and images sent to monitor station

2. Split Condition

Splits usually follow merges: the vehicles that were overlapped separate and are detected as distinctly separated vehicles in subsequent frames. Figure 22 shows two cars coming from different directions that split after being merged in the previous frame.

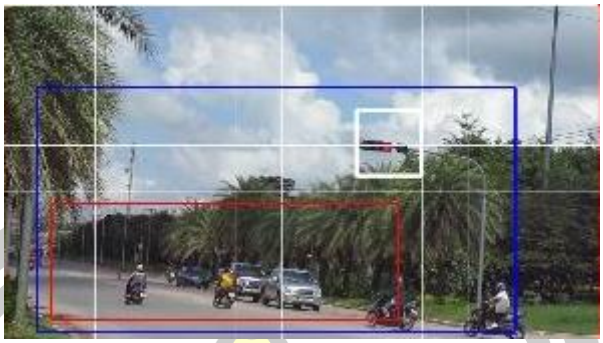


Figure 29 Original Frame: Note that in this scene the two cars in the opposite lane are stationary (presumably blocked waiting for the two motorcycles coming in from the right) and thus were removed as part of the background (frame 4546)

Figure 30 Splitting after Merging: Several frames after Figure 29: the motorcycle on the left has split (its bounding is now much smaller). The vehicle behind has stopped so is removed by background subtraction. The other two motorcycles are correctly located in their bounding boxes: again the car behind the yellow motorcycle is stopped and removed by background subtraction.

Figure 31 Vehicles detected – Split condition

II. Vehicle Tracking (Kalman Filter)

Tracking may ‘lose’ a vehicle when it becomes mostly occluded by one in front of it. Although the system tracks a vehicle when it is only partially occluded by another and there is a sufficient visible template to match it from a previous frame, eventually this partial template becomes too small for reliable matching. The model for each vehicle retains a history of known positions in previous frames and uses this history to predict the next position using the Kalman filter. In tests, a vehicle was ‘lost’ in 5328 frames but the Kalman filter predicted the position successfully in 5328 frames for a 91% success rate.

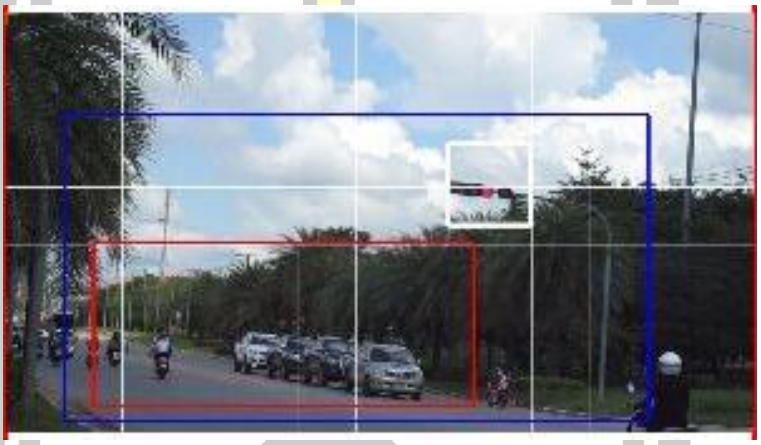


Figure 32 Original Frame (frame number 5328)

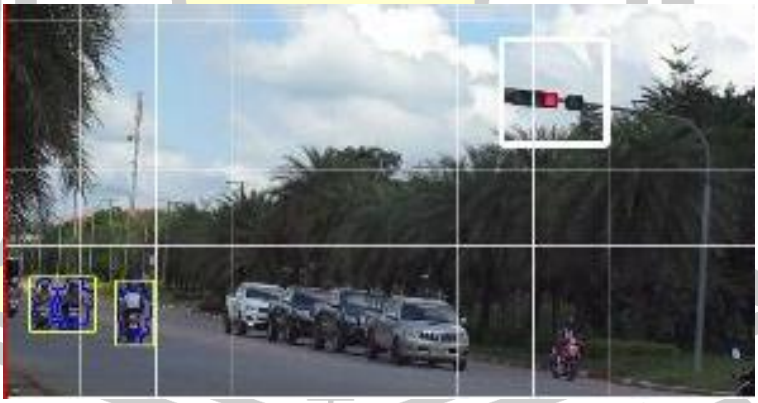


Figure33 Frame 5332 – bounding boxes correctly placed after BGS and contouring alone; two motorcycles on the left have been tracked together because they were too close from many frames previously

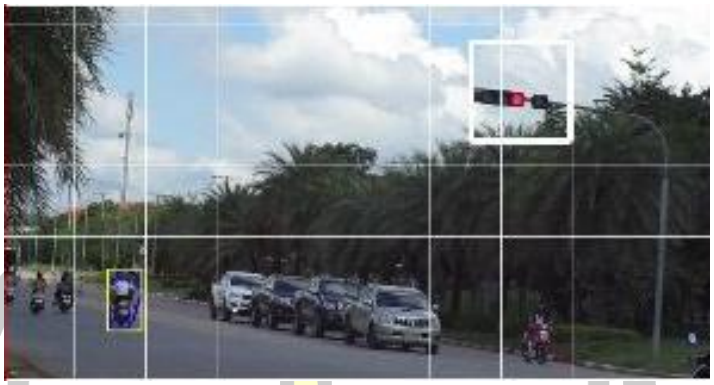


Figure34 Frame 5332 After BGS, the vehicle was lost

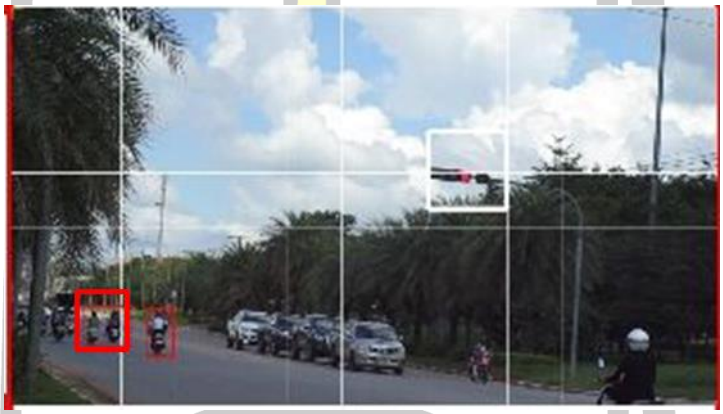


Figure35 Frame 5332 Kalman Filter used to estimate bounding box (bold red) incorrect

Figure36 Final detected image of two Red Light Runners from frame 5332 (tracked together from the start because they were moving at the same speed)

One interesting phenomenon showed the effectiveness of the Kalman filter. At a distance, the vehicle becomes small (the bounding box shrinks due to the perspective effect and it may become part of the adaptive background and simply disappear. However, the system continues to track it until it is predicted to leave the

processing area and it has been observed to ‘reappear’, i.e. captured in a subsequent frame as a separate object. This is a consequence of the background model, which consists of a number of Gaussians and a probability that any pixel, which has a color close to that of the background, may fall into the background class. However, this probability may drop (due to image noise) – causing a vehicle to re-appear in a subsequent frame.

III. Data Transfer

Figure 23 shows a (50x50pixels, the minimum size of the model) image of a Red Light Runner successfully transferred from the PC at the intersection to a mobile phone. The latency was estimated only 0.36s, i.e. well below the time needed (~30 s) for a Red Light Runner to travel 500m with speed 60km/h. The mobile phone images are clear enough to allow correct identification of the RLR when stopped at the monitor station. The travel time is long enough to allow many RLR images to be transmitted before the RLR reaches the monitor station, so potentially a full scene image showing the RLR, in the forbidden region while the light is red, can also be sent. The travel time is also sufficient to allow for variable and unpredicted latency expected by using the public shared network, so it is likely that this will not cause many messages to be lost. Multiple transmission of the RLR image will ensure that it is rarely corrupted or completely lost.

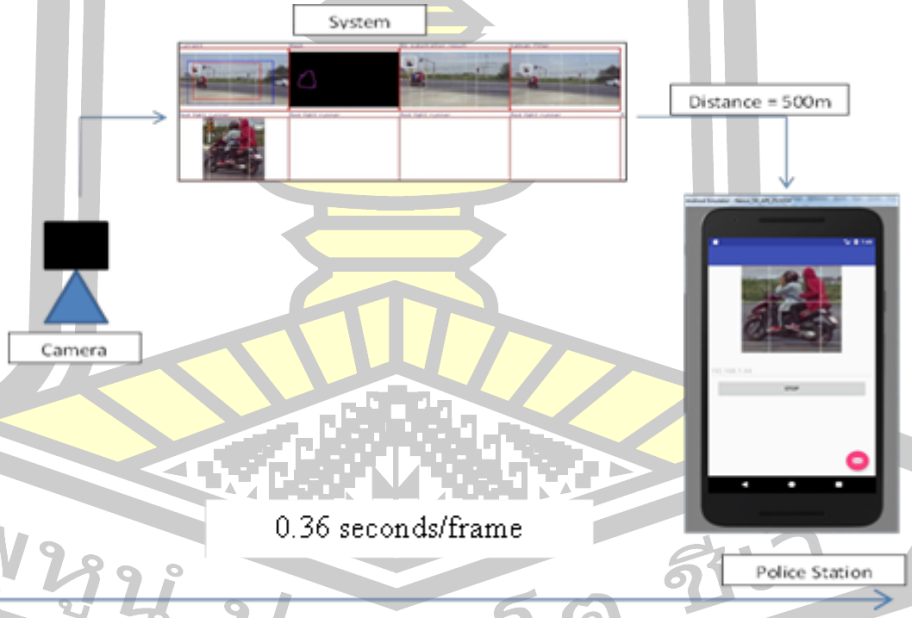


Figure 37 Data transfer – sending the RLR image information from monitor to mobile

Chapter 5 Conclusions

In conclusion, my system completely tracked Red Light Runners (RLRs) correctly in 83% of frames and then successfully transferred images of Red Light Runners to a mobile phone using the mobile network 96% of the time. The key steps in the system for each frame were (a) adaptive background subtraction, (b) contouring the moving objects and forming a model for each one including a position, bounding box, image template, velocity and data for a Kalman filter. Since the system is simple and very flexible, the camera could be positioned only a small height above the ground, but this lead to many occlusions being observed. When occlusions were detected my software used a rule-based approach to resolve many merges (where one vehicle appeared in front of another) and splits (when front vehicle again stopped occluding the one behind it). Even though the vehicle was sometimes lost after background subtraction, it was able to understand that it is now part of the background, e.g. it stopped. A Kalman filter was used in some 'fail' situations to predict the current position of the object to make the system more effective. One interesting problem that was encountered was a vehicle in the distance could be included in the adaptive background: the Kalman filter was able to track the vehicle when it returned from the image noise.

Capturing the Red Light Runner image from the monitor and sending it a police mobile phone is very effective since it is easy to use and inexpensive and easy to adopt in a developing country like Thailand.

For future work, I believe my system is ready to be tested by police. To start the system, they only need to have a computer, camera, mobile phone using Android OS version >19 and an application. So it just needs a thorough evaluation by police to see whether it can be used to reduce the rate of this particular traffic violation. For example, we could gather some 'before and after' statistics for rates of Red Light Running before the system is used and then after it has been used for some time. Fatality statistics could also be used to demonstrate the system effectiveness.

REFERENCES

REFERENCES

[1] WHO and Bloomberg, “Road Safety Institutional and Legal Assessment Thailand,” 2016.

[2] World Health Organization (WHO), *Global status report on road safety 2018*. .

[3] Y. TANABORIBOON and T. SATIENNAM, “Traffic Accidents in Thailand,” *IATSS Res.*, vol. 29, no. 1, pp. 88–100, 2005.

[4] R. Retting, A. Williams, and M. Greene, “Red-Light Running and Sensible Countermeasures: Summary of Research Findings,” *Transp. Res. Rec. J. Transp. Res. Board*, vol. 1640, pp. 23–26, Jan. 1998.

[5] R. A. Retting, A. F. Williams, D. F. Preusser, and H. B. Weinstein, “Classifying urban crashes for countermeasure development,” *Accid. Anal. Prev.*, vol. 27, no. 3, pp. 283–294, Jun. 1995.

[6] “Lives Lost - Annual - TAC - Transport Accident Commission.” [Online]. Available: https://www.tac.vic.gov.au/road-safety/statistics/lives-lost-annual. [Accessed: 31-Aug-2017].

[7] T. Sinlapabutra, “Current Situation of Road Safety in Thailand Current Situation and Problems.”

[8] World Health Organization (WHO), “Global Status Report on Road Safety 2015,” p. 340, 2015.

[9] M. Kristan *et al.*, “The visual object tracking {VOT2016} challenge results,” *Eccvw*, pp. 1–27, 2016.

[10] N. H. . Yung and A. H. . Lai, “An effective video analysis method for detecting red light runners,” *IEEE Trans. Veh. Tech.*, vol. 50, no. 4, pp. 1074–1084, 2001.

[11] “Flexible systems and services for Traffic Safety,” 2016. [Online]. Available: https://www.jenoptik.com/products/traffic-safety-systems/combined-speed-redlight-enforcement-monitoring. [Accessed: 15-Sep-2017].

[12] Z. Zivkovic and F. Van Der Heijden, “Efficient adaptive density estimation per image pixel for the task of background subtraction,” *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, May 2006.

[13] K. Klubsuwan, W. Koodtalang, and S. Mungsing, “Traffic violation detection using multiple trajectories evaluation of vehicles,” *Proc. - Int. Conf. Intell. Syst. Model. Simulation, ISMS*, vol. 5, no. 12, pp. 220–224, 2013.

[14] P. Polhan and J. Morris, “Imaging Red Light Runners,” no. March, 2016.

[15] S. K. Choudhury, P. K. Sa, S. Bakshi, and B. Majhi, “An Evaluation of Background Subtraction for Object Detection Vis-a-Vis Mitigating Challenging Scenarios,” *IEEE Access*, vol. 4, pp. 6133–6150, 2016.

[16] M. Babaee, D. T. Dinh, and G. Rigoll, “A deep convolutional neural network for video sequence background subtraction,” *Pattern Recognit.*, vol. 76, pp. 635–649, Apr. 2018.

[17] I. Haritaoglu, D. Harwood, and L. S. Davis, “W4: Real-time surveillance of people and their activities,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 809–830, 2000.

[18] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” *Proc. 1999 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Cat No PR00149*, vol. 2, no. c, pp. 246–252, 1999.

[19] “OpenCV: cv::BackgroundSubtractorKNN Class Reference.” [Online]. Available: http://docs.opencv.org/trunk/db/d88/classcv_1_1BackgroundSubtractorKNN.html. [Accessed: 02-Sep-2017].

[20] M. Piccardi and T. Jan, “Mean-shift background image modelling,” in *2004 International Conference on Image Processing, 2004. ICIP ’04.*, vol. 5, pp. 3399–3402.

[21] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, “Background and foreground modeling using nonparametric kernel density estimation for visual surveillance,” *Proc. IEEE*, vol. 90, no. 7, pp. 1151–1162, 2002.

[22] S. Mittal, “Object Tracking Using Adaptive Frame Di ff erencing And Dynamic Template Matching Method Object Tracking Using Adaptive Frame Di ff erencing and Dynamic Template Matching Method.”

[23] B. P. L. Lo and S. A. Velastin, “Automatic congestion detection system for underground platforms,” in *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing. ISIMP 2001 (IEEE Cat. No.01EX489)*, pp. 158–161.

[24] H. Wang and D. Suter, “A Consensus Based Method for Tracking: Modelling Background Scenario and Foreground Appearance.”

[25] F. El Baf, T. Bouwmans, and B. Vachon, “Fuzzy Integral for Moving Object Detec-tion. FUZZ-IEEE,” pp. 1729–1736, 2008.

[26] D. Culibrk, O. Marques, D. Socek, H. Kalva, and B. Furht, “Neural Network Approach to Background Modeling for Video Object Segmentation,” *IEEE Trans. Neural Networks*, vol. 18, no. 6, pp. 1614–1627, Nov. 2007.

[27] J. Yao and J.-M. Odobez, “Multi-Layer Background Subtraction Based on Color and Texture,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[28] T. Zhou and D. Tao, “GoDec: Randomized Low-rank & Sparse Matrix Decomposition in Noisy Case,” 2011.

[29] O. Barnich and M. Van Droogenbroeck, “ViBe: A universal background subtraction algorithm for video sequences,” *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709–1724, 2011.

[30] A. Hardas and M. Vibha, “Moving Object Detection using Background Subtraction Shadow Removal and Post Processing,” *Int. J. Comput. Appl.*, pp. 975–8887, 2015.

[31] B. N. Subudhi, S. Ghosh, S. C. K. Shiu, and A. Ghosh, “Statistical feature bag based background subtraction for local change detection,” *Inf. Sci. (Ny).*, vol. 366, pp. 31–47, Oct. 2016.

[32] H. Sajid and S. C. S. Cheung, “Universal multimode background subtraction,” *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3249–3260, 2017.

[33] S. Jeeva and M. Sivabalakrishnan, “Survey on background modeling and foreground detection for real time video surveillance,” *Procedia Comput. Sci.*, vol. 50, pp. 566–571, 2015.

[34] D. Mao, Y. Y. Cao, J. H. Xu, and K. Li, “Object tracking integrating template matching and mean shift algorithm,” *2011 Int. Conf. Multimed. Technol. ICMT 2011*, pp. 3583–3586, 2011.

[35] S. Sooksatra and T. Kondo, “Red traffic light detection using fast radial symmetry transform,” in *2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2014*, 2014.

[36] S. Sooksatra and T. Kondo, “Red traffic light detection using fast radial symmetry transform,” in *2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2014, pp. 1–6.

[37] M. B. Jensen, M. P. Philipsen, A. Mogelmose, T. B. Moeslund, and M. M. Trivedi, “Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1800–1815, Jul. 2016.

[38] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *ACM Comput. Surv*, vol. 38, p. 45, 2006.

[39] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, “Visual tracking: An experimental survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, 2014.

[40] Y. Wu, J. Lim, and M. H. Yang, “Object tracking benchmark,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, 2015.

[41] D. D. Doyle, A. L. Jennings, and J. T. Black, “Optical flow background estimation for real-time pan/tilt camera object tracking,” *Meas. J. Int. Meas. Confed.*, vol. 48, no. 1, pp. 195–207, 2014.

[42] K. Kale, S. Pawar, and P. Dhulekar, “Moving object tracking using optical flow and motion vector estimation,” *Reliab. Infocom Technol. Optim. (ICRITO)(Trends Futur. Dir. 2015 4th Int. Conf.*, pp. 1–6, 2015.

[43] “OpenCV: cv::KalmanFilter Class Reference.” [Online]. Available: http://docs.opencv.org/3.2.0/dd/d6a/classcv_1_1KalmanFilter.html. [Accessed: 03-Sep-2017].

[44] W. Koch, M. Schikora, and D. Cremers, “Multi-Object Tracking Via High Accuracy Optical Flow and Finite Set Statistics,” *Inf. Fusion*, pp. 1409–1412, 2011.

[45] X. Li, K. Wang, W. Wang, and Y. Li, “A multiple object tracking method using Kalman filter,” *Inf. Autom. ...*, vol. 1, no. 1, pp. 1862–1866, 2010.

[46] J. Su, B. Li, and W.-H. Chen, “On existence, optimality and asymptotic stability of the Kalman filter with partially observed inputs,” *Automatica*, vol. 53, pp. 149–154, Mar. 2015.

[47] G. Y. Kulikov and M. V. Kulikova, “Accurate continuous-discrete unscented Kalman filtering for estimation of nonlinear continuous-time stochastic models in radar tracking,” *Signal Processing*, vol. 139, pp. 25–35, 2017.

[48] D. Meyer and G. Zobrist, “TCP/IP versus OSI,” *IEEE Potentials*, vol. 9, no. 1, pp. 16–19, Feb. 1990.

[49] "TCP/IP Protocol Architecture Model (System Administration Guide, Volume 3)." [Online]. Available: https://docs.oracle.com/cd/E19455-01/806-0916/ipov-10/index.html. [Accessed: 18-Jan-2018].

[50] *GSM Global system for Mobile Communications*. 4G Americas.

[51] A. Janecek, D. Valerio, K. A. Hummel, F. Ricciato, and H. Hlavacs, "The Cellular Network as a Sensor: From Mobile Phone Data to Real-Time Road Traffic Monitoring," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2551–2572, 2015.

[52] Hiroshi Lockheimer, "Official Google Blog: Android has helped create more choice and innovation on mobile than ever before," *Google Official Blog*. [Online]. Available: https://googleblog.blogspot.com/2015/04/android-has-helped-create-more-choice.html. [Accessed: 31-Oct-2017].

[53] "Android Developers Blog: Android Studio 3.0." [Online]. Available: https://android-developers.googleblog.com/2017/10/android-studio-30.html. [Accessed: 18-Jan-2018].

[54] "Meet Android Studio | Android Studio." [Online]. Available: https://developer.android.com/studio/intro/index.html#project-structure. [Accessed: 18-Jan-2018].

[55] A. Varghese, "Background Subtraction with Outlier Replacement," no. December, pp. 45–49, 2015.

[56] A. Nurhadiyatna, W. Jatmiko, B. Hardjono, A. Wibisono, I. Sina, and P. Mursanto, "Background Subtraction Using Gaussian Mixture Model Enhanced by Hole Filling Algorithm (GMMHF)," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 4006–4011.

[57] "Template Matching — OpenCV 2.4.13.3 documentation." [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html. [Accessed: 02-Sep-2017].

BIOGRAPHY

NAME	Miss Channa Meng
DATE OF BIRTH	October 14, 1991
PLACE OF BIRTH	Cambodia
ADDRESS	Khon Kaen, Thailand
POSITION	Software Engineering
PLACE OF WORK	Inet-logistics GmbH, Lak Muang Office Gallery 3/21 Sri Chan Road, Nai Muang Subdistrict Muang District, Khon Kaen 40000
EDUCATION	2008 Junior High School, Chbar Ampov School 2010 Senior High School, Chbar Ampov School 2014 Bachelor of Science degree in Computer Sicence (B.Sc.), Mahasarakham University 2019 Master of Science degree in Engineering Electrical and Computer Engineering (M.Eng.), Mahasarakham University
Research grants & awards	The Mahasarakham International Journal of Engineering Technology (MIJET)
Research output	Tracking Vehicles in the Presence of Occlusions