AUTOMATIC VEHICLE DETECTION AND OCCLUSION HANDLING AT ROAD INTERSECTIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BERK ÜLKER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2015

Approval of the thesis:

AUTOMATIC VEHICLE DETECTION AND OCCLUSION HANDLING AT ROAD INTERSECTIONS

submitted by **BERK ÜLKER** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Gözde Bozdağı Akar
Supervisor, **Electrical And Electronics Eng. Dept., METU** _____

Examining Committee Members:

Prof. Dr. A. Aydın Alatan
Electrical And Electronics Engineering Department, METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical And Electronics Engineering Department, METU _____

Prof. Dr. Uğur Halıcı
Electrical And Electronics Engineering Department, METU _____

Assoc. Prof. Dr. Alptekin Temizel
MODSIM, METU _____

Assist. Prof. Dr. Osman Serdar Gedik
Computer Engineering Department, Yıldırım Beyazıt University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: BERK ÜLKER

Signature :

ABSTRACT

AUTOMATIC VEHICLE DETECTION AND OCCLUSION HANDLING AT ROAD INTERSECTIONS

Ülker, Berk

M.S., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. Gözde Bozdağı Akar

September 2015, 110 pages

Vision based intelligent transport system applications are extensively utilized and researched in recent years. Several applications with tracking, classification and counting functionalities are used for automatization of traffic management. Work in this thesis aims to provide an accurate vehicle detection method for improving performance of these tasks. Vehicle detection starts with detection of moving objects, using a background subtraction algorithm. Then, accuracy of the foreground mask is improved using a shadow detection algorithm. Occlusions are detected from both geometrical properties of blobs in binary mask, and associations between objects from consecutive observations. A segmentation method based on the assumptions on object geometry under occlusion is proposed and implemented, to detect vehicles under occlusion correctly.

Proposed solution is tested on several videos collected from different intersections. Results indicate a significant improvement in performance compared to the existing methods in literature.

Keywords: Vehicle Detection, Occlusion, Segmentation

ÖZ

KAVŞAKLARDA OTOMATİK ARAÇ TESPİTİ VE KAPANMA İŞLEME

Ülker, Berk
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Gözde Bozdağı Akar

Eylül 2015 , 110 sayfa

Görüntü işleme tabanlı akıllı ulaşım sistemleri son yıllarda yaygın şekilde uygulanmaya ve araştırılmaya başlamıştır. Hedef takibi, sınıflandırma, sayım gibi görevleri yerine getiren pek çok uygulama geliştirilmiştir. Bu tez çalışmasının amacı, söz konusu uygulamaların başarı oranını yükseltecek bir araç tespit metodu sağlamaktır. Araç tespiti, arka plan modeli oluşturularak sahnedeki hareketli objelerin tespiti ile başlamaktadır. Elde edilen ikili imgenin isabeti, gölge tespit metodu kullanılarak yükseltilmiştir. Kapanma durumu, ikili büyük objelerin geometrik özeeliklerinden, ve objelerin ardışık kesmelerde birbirleri ile olan ilişkilerinden elde edilen bilgilerle tespit edilmektedir. Kapanma durumundaki objelerin geometrik özelliklerine ait varsayımlara dayanan, ve bu objelerin doğrulukla tespitini amaçlayan bir kesimleme algoritması önerilmiş ve gerçeklenmiştir.

Önerilen çözüm farklı kavşak noktalarından toplanan çeşitli videolar üzerinde test edilmiştir. Sonuçlar literatürde bulunan metodlara göre kayda değer bir performans artışına işaret etmektedir.

Anahtar Kelimeler: Araç Tespiti,Kapanma,Kesimleme

To my family...

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Prof. Dr. Gözde Bozdağı Akar for her guidance and encouragement.

I am also grateful to Gizem Kocalar, for her assistance in ground truth generation for selected datasets, and editorial support.

I also want to thank Emre Tunalı, for his assistance in implementation of Self Adaptive Gaussian Mixture Model in C++.

Finally, I would like to thank to my family for their emotional support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AET	Average Execution Time
BMR	Bayesian Minimum Risk
CAPOA	Content Adaptive Progressive Occlusion Analysis
CDNET	ChangeDetection.NET
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPGA	Field Programmable Gate Array
FPR	False Positive Rate
GDM	Generalized Deformable Model
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HOG	Histogram Of Gradients
IDE	Integrated Development Environment
IDR	Interior Distance Ratio
ITS	Intelligent Transport Systems
KDE	Kernel Density Estimation
KL	Kullback-Liebler
LBP	Local Binary Pattern
LBSP	Local Binary Similarity Pattern
LOBSTER	Local Binary Similarity Segmenter
LR	Large Region
MET	Maximum of Execution Time
MHI	Motion History Images
MOD	Moving Object Detection
OpenCV	Open Source Computer Vision

PBAS	Pixel Based Adaptive Segmenter
PDF	Probability Distribution Function
PWC	Percentage of Wrong Classifications
RDHOG	Relative Discriminative Histogram Of Gradients
RE	Recall
ROC	Receiver Operating Characteristics
ROI	Region Of Interest
SAGMM	Self Adaptive Gaussian Mixture Model
SP	Specificity
SR	Small Region
TP	True Positive
TP	True Negative
VET	Variance of Execution Time
VMTM	Variant Mask Template Matching

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Demand for efficient and reliable transportation rises globally due to growth of industries and human population. This demand drives increase in both number of vehicles and distance traveled, which means increased traffic volume. Steady increase of the traffic volume reached to a point that handling of congestion and emergency situations could only be possible with a detailed traffic management approach. In recent years, efforts on traffic management field centralized under Intelligent Transport Systems (ITS). Today ITS covers methods for analysis of traffic related parameters for planning and management, control and optimization of traffic flow, detection and handling of traffic related events. Furthermore combined with surveillance applications, ITS decrease complexity and cost of imposing regulations and inspecting drivers.

Development of first generation ITS are triggered by increase in traffic accident related deaths and injuries, and ineffective traffic flow [20–23]. These systems were often based on invasive sensors, which are costly to maintain and install such as road surface inductive loops, and mechanical sensors. Moreover, most of the systems require an operator at observation, decision and control stages. Limited accuracy of sensors and human induced errors by operators limit success of these systems.

Development of sensor technology and intelligent applications enabled design of advanced and accurate ITS applications. Advances in sensor technology and cameras lead to development of complex vision applications to collect traffic parameters. Widespread use of cameras provided traffic analysts valuable parameters and met-

rics, which lead to developing efficient management applications for traffic control. Furthermore, installation and maintenance costs are dramatically reduced, as invasive sensor applications become obsolete. Although need for operators for ITS is not completely eliminated, increase in computational power and advances in data processing methods refined the data need to be handled by operators. Furthermore, vision based applications enabled covering larger areas of interest, improving efficiency of ITS applications.

Introduction and integration of vision based solution to ITS enabled handling of much more complex tasks which were not practical to handle with existing methods. These solutions range from collection of traffic analysis data such as vehicle counts by class, traffic volume, congestion statistics, to event based real time management tasks such as detection of violations, emergency detection, accident detection and toll collection. Furthermore, long term data collected from these systems is a key input for planning of new infrastructure and systems. Vision based solutions play a key role in these systems by handling object detection, classification and tracking. Completing these tasks with accuracy and high speed becomes a challenge, as working environment of these systems has a complex nature. Illumination variations, irregular vehicle movement patterns, presence of occlusions are some of the challenges researchers have been working on.

1.2 SCOPE OF THESIS

Purpose of this thesis work is to provide an object detection solution, with occlusion detection and handling capabilities. Most tasks in vehicle surveillance applications are based on tracking, classification and counting of the vehicles. All these tasks require robust and accurate object detection to perform their function properly. Also real time operation is crucial, as most of these tasks are implemented for online systems. Therefore, a solution to accurately detect vehicles, while maintaining a considerable speed is targeted.

Intersections are the points where traffic flow becomes highly irregular, and vehicles accumulate. Therefore occlusion frequently occurs, and performance of higher layer

tasks like tracking and classification degrade. The solution proposed in this thesis aims to reduce this degradation by improving object detection accuracy, with a geometry based occlusion handling algorithm.

When scene geometry, vehicle motion and camera placements are considered, it is concluded that a moving object detection is needed. Therefore, many methods from literature with different approaches to the problem, are examined. Theoretical performance expectations are compared with both qualitative and quantitative results to select optimal solution.

Occlusion detection and handling tasks are the main focus of this thesis work, and both inter frame and intra frame features are utilized for this purpose. A wide range of ITS solutions are examined from literature, and existing methods are analyzed from different aspects. An improved occlusion handling framework for accurate object detection is proposed, which is based on shape geometry properties of the objects appearing in foreground binary mask.

Vehicle group or vehicle level association is utilized for inter frame occlusion detection. It must be stated that target tracking is not in the scope of this work, but the solution proposed in this thesis can work in conjunction with a target tracking algorithm, resulting a higher performance for both functions.

To evaluate performance of the proposed system, several tests in subsystem and framework levels are conducted. Also, a base method in literature is selected and implemented for comparative evaluation of both performance and computational cost.

1.3 OUTLINE OF THESIS

Thesis consists of five chapters. In Chapter 2, existing methods for occlusion detection, occlusion handling and object detection are examined in detail. A theoretical analysis supported by provided experimental results is done to evaluate each method proposed.

In Chapter 3, a multi stage solution for object detection is proposed. For each stage, detailed explanations of used and proposed methods are given.

In Chapter 4, implementation details, system and subsystem level tests and evaluation metrics are given. Results are compared to existing methods.

In the last chapter, results obtained in Chapter 4 are discussed along with theoretical expectations. Improvements and weak points of the proposed solution are given with possible future extensions.

CHAPTER 2

BACKGROUND AND LITERATURE

2.1 MOVING OBJECT DETECTION

Moving object detection is one of the most important and popular topics in vision. It is mostly utilized in first stages of more complex vision applications, since most of higher level tasks such as tracking and classification require precise detection of moving objects. Most surveillance applications, including surveillance based ITS applications, are dependent on moving object detection.

Moving object detection can be defined as operation of defining pixels belonging to moving objects relative to the scene or background. Different approaches are adopted for moving and stationary camera scenarios. In scope of this thesis, only stationary camera based moving object detection methods are examined and utilized.

There are many completed works and many challenges present in moving object detection field. Most common challenges present in literature can be listed as follows:

- Lack of robustness to global illumination variations
- Lack of robustness to change in camera parameters and non-uniformities in sensors
- Noise caused by imaging device
- Difficulty in detection of moving objects or object parts similar to background
- Non stable background objects (vegetation, light emitting objects)

- Moving objects with movement patterns like slowing or stopping for certain amounts of time

- Local illumination variations on background caused by light sources associated with moving objects, such as headlights of vehicles

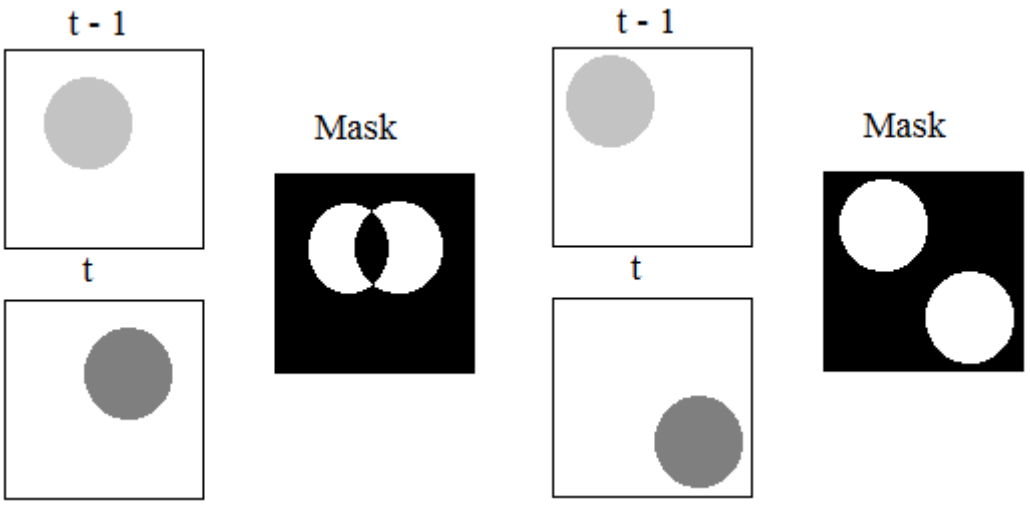
Many different approaches are discussed in literature, each one answering the above challenges with different success. In this thesis work, these challenges are going to be examined and prioritized according to our application domain. Then suitable algorithms will be evaluated quantitatively and qualitatively.

There are also works discussing groupings of moving object detection methods [24–27]. Based on these work, there are three main types of moving object detection algorithms, grouped by the method of detecting foreground pixels.

2.1.1 TEMPORAL DIFFERENCING

Temporal differencing based methods rely on the assumption that a moving object causes noticeable intensity difference in intensity values of pixels associated with movement. Basic approach here is using previous frame as a background model for the next frame [28]. Although it is computationally efficient with small memory usage, this approach lacks precision in object localization, and prone to produce erroneous results as illustrated in Figure 2.1. Results are dependent on temporal distance between consecutive frames. If temporal distance is small, and object overlaps itself in two consecutive frames, resulting in cavities in object mask. Conversely, if temporal distance is large, object may have moved away from previous position, resulting a ghost of object in object mask. It must be noticed that different objects with different speeds relative to the scene may produce both of these erroneous responses, and adjusting temporal distance is not a solution.

To improve object localization and reduce erroneous response, several methods are proposed. Collins et al [29] proposed a three frame differencing algorithm, which improves object localization by incorporating object mask from previous frame in decision for current frame. To further improve object localization and eliminate ghosts,



(a) Slow movement case (b) Fast movement case

Figure 2.1: Fail cases for simple temporal differencing

Motion History Images (MHI) is proposed [30, 31]. MHI is not restricted to any temporal distance as in three frame differencing. Instead, motions from distant frames are weighted accordingly to enable decaying. MHI can also utilize future frames, in expense of constant a delay in generation of output mask. Although MHI has better performance, processing load and memory usage significantly increases compared to basic approaches mentioned before.

It is worth to emphasize that, temporal differencing based methods do not produce any robust background model. Lack of a maintained background model is major disadvantage, considering scenarios in which objects show irregular motion characteristics like moving with different speeds or stopping for a certain amount of time.

2.1.2 BACKGROUND MODELING

This type of methods are based on construction of a background model, and often adaptation of this model based on changes occurring in observed scene. Then a foreground mask is constructed, by observing differences between current frame and constructed background model. Although many different approaches exist, nearly all proposed work under this category consists this two main steps. Difference between methods are variations of these steps, namely constructing and updating background

model, and measurement of difference from this model.

Early methods proposed often utilized pixel level independent models. These models often include pixel intensity levels. Simplest form of this approach is pixel intensity averaging, and using a Kalman filter [32] to update background model [33–35]. Although averaging costs less computational power, and requires lower memory usage, results often suffer from artefacts like ghosts, or cavities within objects as illustrated in Figure 2.1. To overcome these problems several methods are proposed such as utilizing an adaptive threshold and learning rates. Later, more complex models are introduced, which can be grouped under two main categories as Parametric Models and Non-Parametric Kernel density estimations.

Parametric models have become a popular research area, and many methods and improvements are proposed in literature. Earlier form of this approach is a simple single Gaussian kernel for modelling background. In this approach, a Normal Distribution $N(\mu, \sigma)$ is constructed for each pixel. This distribution is initialized as μ being temporal average of pixel, and σ is variance of intensity value in this period.

Then for each new frame, foreground mask M is obtained as in Equation 2.1, while K is a distance parameter typically set to 2.5 in most applications.

$$M(i,j) = \begin{cases} 1, & |I(i,j) - \mu(i,j)| > K\sigma. \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Parameters of the model are updated as in Equation 2.2 and Equation 2.3, while α is learning rate and i and j are pixel coordinates.

$$\mu_{new}(i,j) = \begin{cases} \alpha I(i,j) + (1-\alpha)\,\mu(i,j), & M(i,j) = 1. \\ \mu(i,j), & \text{otherwise.} \end{cases} \tag{2.2}$$

$$\sigma^2_{new}(i,j) = \begin{cases} \alpha\left(I(i,j) - \mu(i,j)\right) + (1-\alpha)\,\sigma^2(i,j), & M(i,j) = 1. \\ \sigma^2(i,j), & \text{otherwise.} \end{cases} \tag{2.3}$$

As an extension of this approach, Staufer and Grimson [36] proposed using of a weighted mixture of Gaussian distributions to model each pixel. Model is expressed as in Equation 2.4, where $\hat{\overrightarrow{\mu}}_m$ and $\hat{\sigma}_m$ are estimates of mean and variance for each

mode respectively, $\hat{\pi}_m$ are mixture weights, I is identity matrix, and chi_T is sample set for training.

$$\hat{p}\left(\overrightarrow{x}|\chi_T, BG+FG\right) = \sum_{m=1}^{M} \hat{\pi}_m N\left(\overrightarrow{x}; \hat{\overrightarrow{\mu}}_m, \hat{\sigma}_m^2 I\right) \tag{2.4}$$

This approach further refined by utilizing information that foreground pixels appear significantly less compared to background pixels as in Equation 2.5, where B is the number, which defines how many of the largest modes are selected. An adaptation rate is used to determine B, in order to prevent foreground data to influence background model [37].

$$p\left(\overrightarrow{x}|\chi_T, BG\right) \sim \sum_{m=1}^{B} \hat{\pi}_m N\left(\overrightarrow{x}; \hat{\overrightarrow{\mu}}_m, \sigma_m^2 I\right) \tag{2.5}$$

Gaussian mixture models become quite popular, and used in many vision applications [38–44]. Also, several improvements are introduced to both governing equations of model update [45–48], and distance measure. One of the most noticeable of these works is introduced by Zivkovic and Geijden [49]. They present a new set of recursive equations to constantly update Gaussian Mixture Model (GMM) parameters, and to determine number of required mods for each pixel dynamically. They also present a non-parametric density estimation method. Utilizing this method, they achieved superior segmentation performance compared to their predecessors [36, 50], with reduced computational cost. Later, Self-Adaptive GMM (SAGMM) [51] is introduced, addressing to the problems caused by constant learning rate, such as incorporation of slow moving objects into background, or large amount of false positives due to slow adaptation. SAGMM introduces a separate learning rate for each mode of the mixture model. With this extra parameter, they achieved improvement in convergence and accuracy of the background model, without losing temporal adaptability.

Non-Parametric models based on Kernel Density Estimation (KDE) are introduced and developed along with parametric methods. In Non-Parametric model based methods, probability density functions (PDF) associated with background and foreground differ for different scenes, and they do not have a parametric generalized form. PDF is estimated directly from set od samples, without using assumptions on distributions. Work of Elgammal et al [50] being a milestone, several improvements and new methods are proposed based on this approach [1, 37, 52–56]. In most methods based on

this approach, modelling of repetitive movements with long and short periods became problematic, due to high memory requirements caused by stored observations [4]. Due to same reason, these methods are highly data driven, and considered for hardware platforms specialized in parallelization, such as FPGAs or GPUs.

One of the methods introduced to reduce memory use in non-parametric methods is ViBe [1, 54, 57]. This method has become a base method for most of the successful methods competing with benchmarks, on [1]. ViBe opposes the traditional approach that, oldest samples are removed first, from the set of observations when construction background model. Instead, a stochastic sampling approach with random observation replacement is adopted. This approach is further improved in [55, 56]. Each background pixel is modelled as an array of samples from different frames as in Equation 2.6.

$$M\left(x\right) = \{v_1, v_2, ..., v_N\} \tag{2.6}$$

Decision on pixel level is different from classical approaches. Unlike parametric model based methods, a pixel can be identified as background, even if it is only close to a 'few' of the samples in observation set. This means, new value does not need to be close to the majority of the sample set. Exact mechanism is illustrated in Figure 2.2 [57]. A sphere with radius R is formed on 2D space, centered on the current pixel value $v\left(x\right)$. A decision threshold, $\#_{min}$, is defined. If number of samples intersecting with the sphere is larger or equal to $\#_{min}$, then pixel is classified as background.

After decision, a randomly selected sample from the set M is replaced with current value $v\left(x\right)$. ViBe also increases spatial consistency, by propagating a pixel value to observation set of a randomly selected neighbor, if a pixel is detected as background. This is a major advantage versus pixel-based methods, which rely only on parameter adaptation to achieve spatial consistency. Some work [58] is done to achieve similar consistency with pixel based GMM, utilizing a region based algorithm, although with limited success and high computational cost.

In [56], an improved version of ViBe, Pixel Based Adaptive Segmenter (PBAS) is introduced by Hoffman et al. Based on ViBe implementation in [57], this method replaces static thresholds imposed in ViBe, with dynamic state variables. Firstly, it replaces 2D distance threshold R, with $R\left(x_i\right)$. $R\left(x_i\right)$ is independent for each pixel,

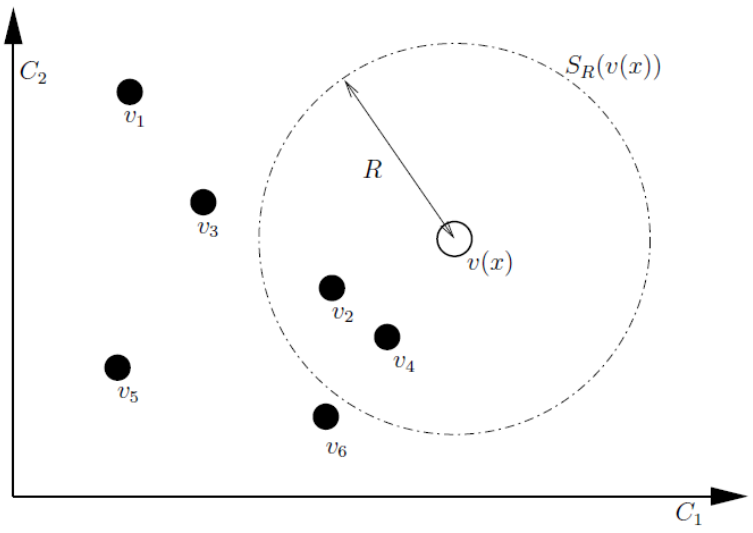


Figure 2.2: Pixel level decision in ViBe. $v\,(x)$ is center pixel, and v_n are samples from training set [57].

and updated after each frame, where x_i is index of each pixel. By adding an update mechanism for $R\,(x_i)$, this variable is adapted to increase for highly dynamic areas, and decrease for static areas of the scene. Therefore, false positive rate in dynamic areas are reduced, and foreground detection accuracy is increased in static background regions.

PBAS also improves ViBe by introducing a variable learning rate, $T\,(x_i)$ for each pixel. Instead of replacing a randomly selected neighbor sample every time a background pixel decision is made as in ViBe, this replacement is done with probability of $1/T\,(x_i)$. This change enables control of the learning rate. Also $T\,(x_i)$ is increased for large blobs, and decreased for small blobs. This way, large blobs corresponding to the moving objects in the scene are only slightly 'eaten up', while small erroneous blobs are completely eliminated.

With discussed methods and improvements, pixel-based method family offers a lightweight and effective solution for moving object detection. But an important information supplied in video sequences, spatial relations between pixels are not utilized. Region, texture and block based methods are introduced to exploit this information by using feature descriptors, block descriptors, [3,58,59] and color distributions [60]. Use of local binary descriptors are studied, an a solution based on Local Binary Pat-

tern (LBP) is proposed in [3]. A LBP descriptor is defined as in Equation 2.7 and calculated as in Figure 2.3.

$$LBP_{P,R}\left(x_c, y_c\right) = \sum_{p=0}^{P-1} s\left(g_p - g_p\right) 2^c, \quad s\left(x\right) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.7}$$

where R is the nxn neighborhood of center pixel, P is the selected subset od R, (x_c, y_c) is pixel coordinate of center pixel, g_c is gray value of center pixel, g_p is the gray value of selected pixel from the set P, s is the decision function.

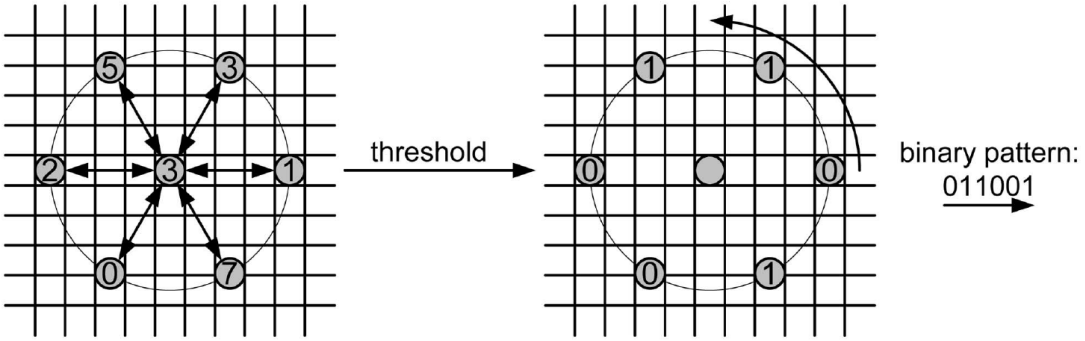


Figure 2.3: LBP sampling pattern, retrieved from [3]

As LBP is a texture primitive statistic, it strongly describes the spatial relation between local pixels, without being affected from mean intensity levels. This means, this descriptor is invariant to the global or local illumination changes, and shadows.

To improve LBP feature, and enable detection of both texture and intensity variation, Local Binary Similarity Pattern (LBSP) is proposed [61]. LBSP is defined as in Equation 2.8 and Equation 2.9, and calculated as in Figure 2.4.

$$LBSP_R\left(x_c, y_c\right) = \sum_{p=0}^{P-1} d\left(i_p - i_c\right) 2^p \tag{2.8}$$

$$d\left(x\right) = \begin{cases} 1 & |x| \leq T_d \\ 0 & |x| > T_d \end{cases} \tag{2.9}$$

where R is the nxn neighborhood of center pixel, P is the selected subset od R, (x_c, y_c) is pixel coordinate of center pixel, i_c is intensity of center pixel, i_p is the

intensity of selected pixel from the set P, d is the decision function, and T_d is the decision threshold.

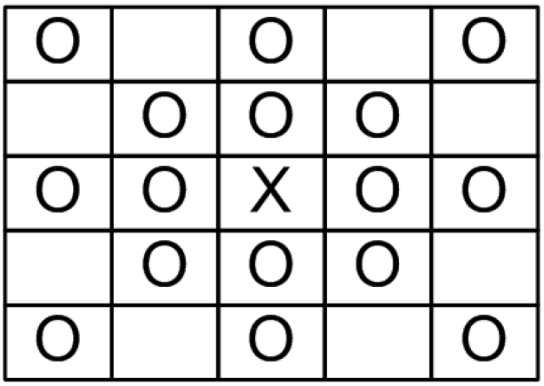
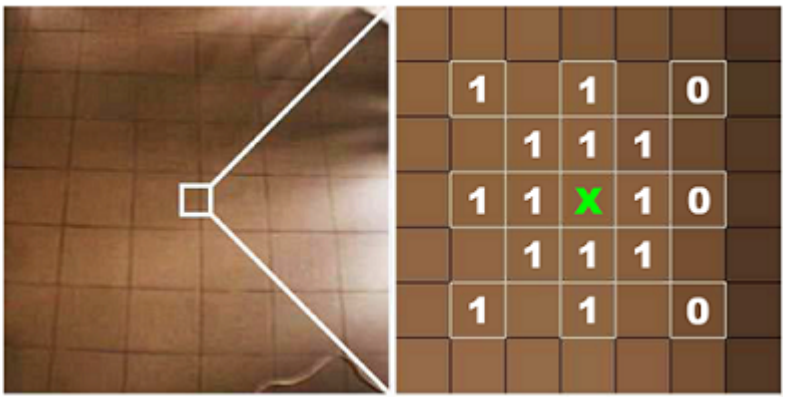
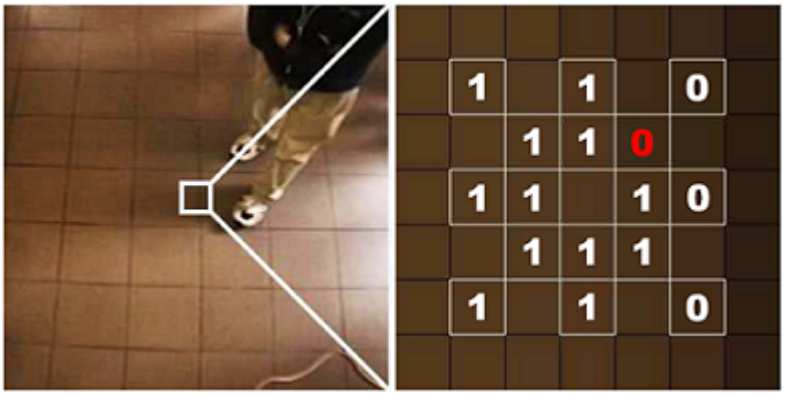


Figure 2.4: LBSP sampling pattern, retrieved from [4]

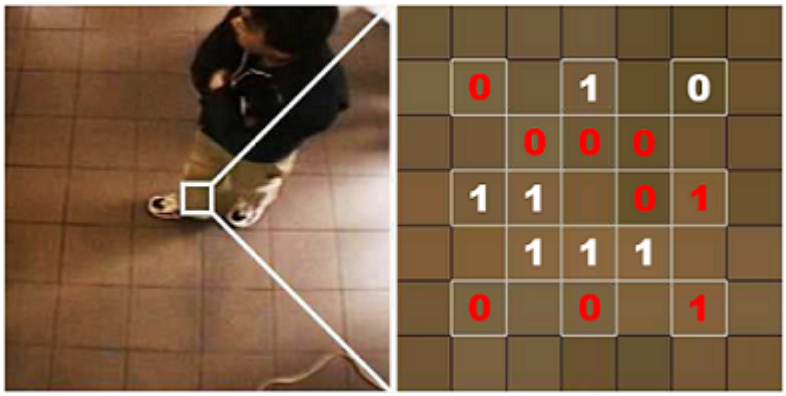
Two LBSP descriptors are proposed, namely intra-LBSP and inter-LBSP for updating background model and detecting foreground respectively. Intra-LBSP is calculated within background model. Inter-LBSP is calculated by selecting center pixel in current frame instead of background model. In Figure 2.5, these two descriptors, and a typical evaluation is shown.



(a) Pixel is tagged as background



(b) Pixel is tagged as background



(c) Pixel is tagged as foreground

Figure 2.5: LBSP pixel level decisions, retrieved from [4]

New methods utilizing LBSP descriptor are proposed by St-Charles et al [4, 62], and achieved considerable success. First, an upgrade to lightweight ViBe is proposed, with using a vector of LBSP descriptors, instead of directly using pixel intensity values [62]. New method is named Local Binary Similarity Segmenter (LOBSTER).

Introduction of LBSP descriptor into a pixel-level algorithm caused an increase of computational cost of sampling and distance measurement [62]. To overcome this, a per-channel approach for reducing number of sample comparisons conducted on LBSP descriptor vectors. This per-channel approach dictates that, descriptors and color values for each channel are compared sequentially. First, a significant difference is checked between pixel intensity values. If intensity values are close, then LBSP strings are compared using Hamming distance thresholding. As this second step is more expensive, first check reduces cost by avoiding unnecessary LBSP descriptor comparison. For each channel of pixel, these checks are performed sequentially until pixel is tagged. This way, average number of comparison and distance calculation operations is reduced.

Another important step to reduce computational complexity is changing the distance measure used in measurement of sample similarity. Instead of using L2 distance as in ViBe, L1 distance is used, which is a cheaper operation. Authors claim that, L1 distance also results better overall performance compared to L2 distance, based on their experiments. Benchmarks done on CDNET dataset indicate that, LOBSTER has achieved significant improvement in overall performance for all videos, compared to the base method ViBe [62]. Results of this benchmark are shown in Table 2.1.

Improvements achieved by introduction of LBSP proved that LBSP descriptors increase spatial consistency of background subtraction. Motivated by these results, St-Charles proposed that PBAS method [56], which is an improved version of the ViBe algorithm, can also be improved using LBSP descriptors. A new method, SuBSENSE [4], is proposed based on PBAS. With utilization of LBSP, SuBSENSE offers better spatial consistency compared to PBAS. Also, adaptability improvements inherited from PBAS enables fast and stable responses to variations in the background.

SuBSENSE also improves adaptability by introducing a new feedback scheme. Authors claim that feedback process proposed in PBAS have two major drawbacks:

Table 2.1: ViBe benchmark results, retrieved from [1]. Metrics are explained in Chapter 4

Category	Recall	Specificity	FPR	FNR	PWC	Precision	F-Measure
baseline	0.8955 (+9%)	0.9983 (∼0%)	0.0017 (-17%)	0.0045 (-40%)	0.5774 (-35%)	0.9558 (+3%)	0.9242 (+6%)
cameraJitt	0.6742 (-5%)	0.9921 (+2%)	0.0079 (-74%)	0.0141 (+22%)	2.0930 (-48%)	0.8368 (+58%)	0.7423 (+24%)
dynamicBg	0.7670 (+6%)	0.9820 (-1%)	0.0180 (+73%)	0.0023 (-15%)	1.9984 (+56%)	0.5923 (+11%)	0.5679 (∼0%)
intermittObj	0.5589 (+9%)	0.9752 (+2%)	0.0248 (-48%)	0.0428 (∼0%)	5.8427 (-25%)	0.7102 (+9%)	0.5770 (+14%)
shadow	0.8784 (+12%)	0.9930 (∼0%)	0.0070 (-14%)	0.0052 (-44%)	1.1549 (-30%)	0.8765 (+5%)	0.8728 (+9%)
thermal	0.8135 (+49%)	0.9934 (∼0%)	0.0066 (+72%)	0.0093 (-71%)	1.4210 (-55%)	0.8572 (-8%)	0.8248 (+24%)
overall	0.7646 (+12%)	0.9920 (+1%)	0.0110 (-35%)	0.0130 (-26%)	2.1812 (-30%)	0.8048 (+9%)	0.7515 (+13%)

1. Local comparison results cannot be used when a pixel is detected as a background.

2. Pixel value is only incorporated into the model by a probability of $1/T_i$.

Results of these drawbacks are long convergence time for variables, and false foreground detections, which are observed in noisy and dynamic areas. To solve these issues, a feedback mechanism observing background dynamics are proposed.

First, a continuously updated moving average, $D_{min}(x)$ is defined as in Equation 2.10 for each pixel x, where $d_t(x)$ is minimal normalized color-LBSP distance between all samples in sample set of background model for both LBSP descriptors and intensities, α is the learning rate. This moving average is used as measure of model fidelity.

$$D_{min}(x) = D_{min}(1-\alpha) + d_t(x)\,\alpha \tag{2.10}$$

However this type of continuous update regardless of pixel classification is not adopted in [56] as updating adaptive distance threshold $R(x)$, and learning rate $T(i)$ based on this variable may cause false classification of slow moving objects into background

model. Slow moving foreground objects or large foreground objects may cause an increase in $D_{min}(x)$. A new feedback variable, $v(x)$ is introduced by authors, to eliminate this effect. $v(x)$ is an indicator of blinking in a pixel, and defined as in Equation 2.11, where $x(t)$ is a map constructed by XOR operation on consecutive frames to detect blinking.

$$v(x) = \begin{cases} v(x) + 1 & if X_t(x) = 1 \\ v(x) - 0.1 & otherwise \end{cases} \tag{2.11}$$

Using these two indicators, $R(x)$ and $T(x)$ are controlled as in Equations 2.12, 2.13, 2.14, 2.15, where R^0_{color} and R^0_{lbsp} are static thresholds, and $S(t)$ is resulting binary map before application of feedback.

$$R(x) = \begin{cases} R(x) + v(x) & if R(x) < (1 + 2D_{min}(x))^2 \\ R(x) - \frac{1}{v(x)} & otherwise \end{cases} \tag{2.12}$$

$$R_{color}(x) = R(x) R^0_{color} \tag{2.13}$$

$$R_{lbsp}(x) = 2^{R(x)} + R^0_{lbsp} \tag{2.14}$$

$$T(x) = \begin{cases} T(x) + \frac{1}{v(x) D_{min}(x)} & if S_t(x) = 1 \\ T(x) - \frac{v(x)}{D_m in(x)} & if S_t(x) = 0 \end{cases} \tag{2.15}$$

With these improvements and adaptations, SuBSENSE becomes an efficient foreground/background segmentation method with successful responses to the multiple challenges presented by complex scenes in surveillance applications.

Figure 2.6 illustrates development of discussed algorithms, with key improvements introduced to base algorithms.

2.2 SHADOW DETECTION

Moving shadow detection has been an active topic, as in most surveillance applications imaging unit requires scene to be illuminated by an external source. This source may be sun or a lighting unit, which causes formation of shadows observable in generated images.

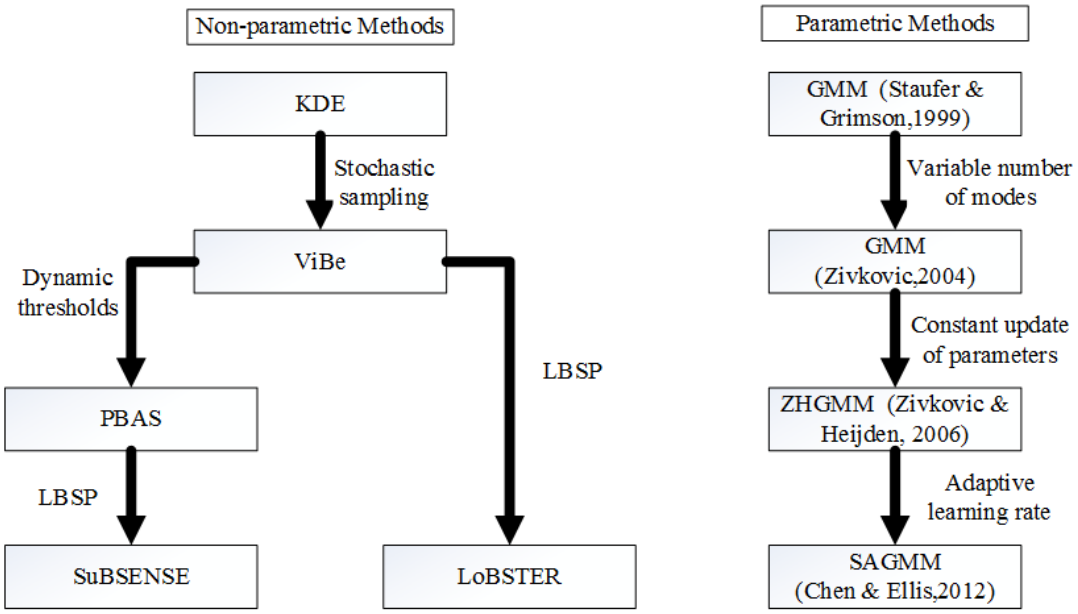


Figure 2.6: Development stages and improvements of some popular state of art background subtraction methods

Moving shadows have a deteriorating effect on performance of vision based applications, as differentiating foreground and background becomes more complex. Therefore, tasks as object detection, tracking and classification require careful consideration of effects of shadows on their performance. In literature, there are several works proposing methods to detect, identify and eliminate shadow.

Shadow is formed by occlusion of the light source by an object. Two regions of shadow are formed in this case, which are self-shadow and cast shadow. Self-shadow is part of the object, which is not illuminated by the light source due to obstruction caused by itself. Cast shadow is the shadow occupying the area behind obstructing object. This area is a two dimensional projection of the object. These two types of shadows are shown in Figure 2.7, with a rendered image.

Cast shadow has two regions, umbra and penumbra [6]. Umbra is the part of the shadow, where light source is totally blocked by the occluding object. Penumbra is the area, where light source is partially blocked. Penumbra area is illuminated directly by some portion of the light source. Notice that a point light source produces only umbra where an area light source can produce both umbra and penumbra. Umbra and

Figure 2.7: Rendering of a pawn under a light source. Self shadows are visible on the body of the object. Image retrieved from [5]

penumbra regions are illustrated in Figure 2.8.

In the context of moving object detection, cast shadows become problematic as they often decrease accuracy of object detection. For most surveillance applications with object tracking feature, moving object detection is a crucial step, bounding an upper limit for overall performance. Without proper segmentation and isolation of each object, errors will be introduced, and propagated to the further steps of surveillance applications such as tracking and classification.

Surveillance tasks on ITS applications are often performed continuously, which requires independence from, or adaptation to, external parameters such as daytime and weather conditions. As weather conditions and daytime change, structure and position of the cast shadows of objects in the observed scene change. For urban traffic scenes in daytime there are two main sources of illumination, sun and ambient light from sky. These two sources have different effects on the observed scene.

Ambient light can be modeled as continuous light source spanning entire sky. This source generally does not produce large cast shadows. Cast shadows from this source become significant only in the areas where large portion of source is blocked such as area under a vehicle. For most surveillance applications, this area is mostly blocked by object itself, and performance degradation caused by these areas are not significant.

Effect of sun can be modeled as a point source, as penumbra area generated by ob-

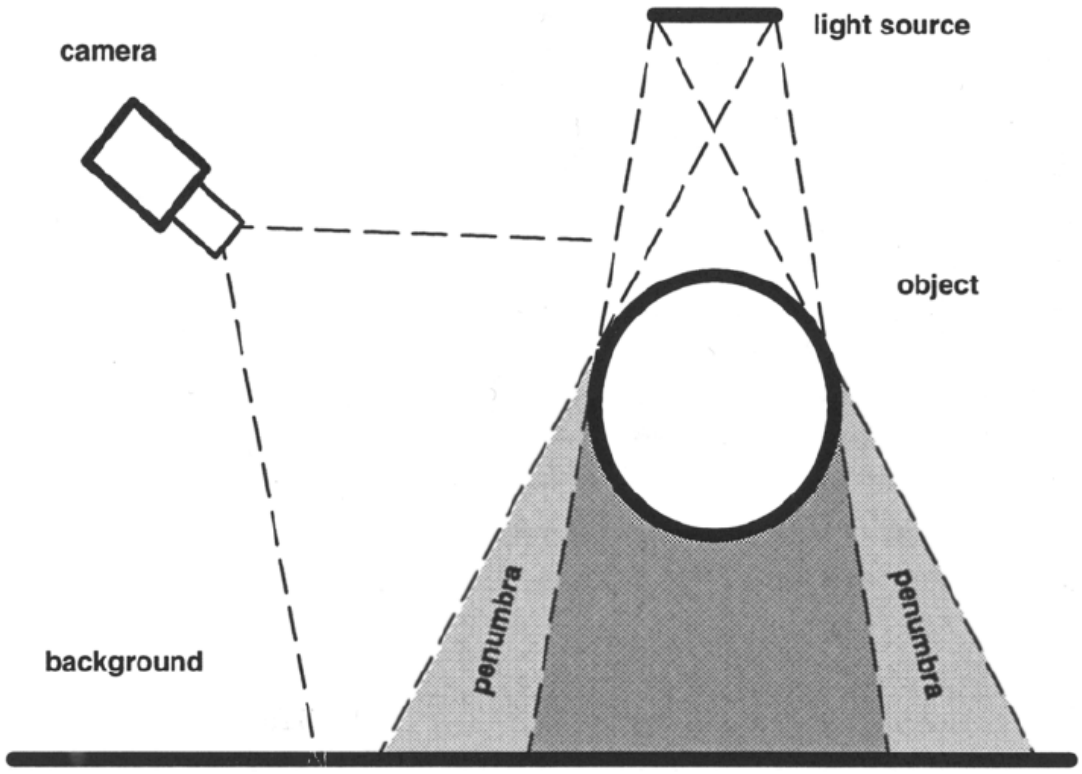


Figure 2.8: Shadow regions, retrieved from [6]

struction of a portion of sun is negligible and often lost due to digitization of the captured scene. Therefore generated cast shadows are often significantly different from non-shaded areas, and shadow area has stronger edges at boundaries. Shape and structure of still and cast shadows change with angle of sunlight during daytime, but when adaptation rate of a foreground detection algorithm is considered, this change is negligible. Therefore still shadows cast by sun are often counted as background features. Figure 2.9 illustrates static shadows from Highway sequence.

Still cast shadows which can be assumed to be stationary during observation interval, are incorporated into the background model. After a learning period, pixels belonging to the still cast shadow areas are regarded as background pixels. Similar case applies to the frame differencing based foreground segmentation algorithms.

Moving cast shadows generated by obstruction of sunlight are problematic cases for surveillance applications, as color values of these shadow pixels are often significantly different from those of background model and shadow regions have similar

Figure 2.9: Static shadows in a background model

motion characteristics with object. For moving object detection methods, moving cast shadow area has different color values which are often enough for misclassifying these pixels as foreground pixels. Also their movement pattern is nearly same as the object casting them. These properties of moving cast shadow pixels cause moving objects to be extracted with their cast shadows after foreground detection stage. As illustrated in Figure 2.10 after the application of GMM based background foreground segmentation, object blobs include object bodies with their cast shadows.

A connected component analysis performed on this erroneous foreground mask can yield errors on object properties such as shape, orientation, size and position. Furthermore, separate objects which are connected by a cast shadow can be misclassified as one object in single blob. An example of such misclassification can be seen on Figure 2.10, where three vehicles without any occlusion have connected blobs in binary mask. These effects can introduce errors to the counting, classification, segmentation and tracking algorithms which are employed widely in vision based ITS applications.

There are various methods to handle moving cast shadows in literature, and several studies are made to evaluate these methods [2, 7, 8]. Prati et al [7] examined several proposed methods up to year 2001, and suggested a two level taxonomy. At first level, they grouped methods by considering whether methods are deterministic or statistical. Then at second level statistical and deterministic methods are further divided into two groups separately, as illustrated in Figure 2.11.

(a) Original image

(b) Binary Foreground Mask

Figure 2.10: Foreground detection under effect of shadow

Later, Al Najdawi et al [2] examined methods up to year 2010, and suggested a new 4 level taxonomy. At first stage, algorithms are classified by checking whether they are designed for shadow detection on specific objects like vehicles or human. At second stage, classification is done by checking whether algorithms are designed for operating on a specific environment like indoor, road or aerial. Third stage classification is based on the domain in which shadow detection features belong. At the fourth stage, algorithms are classified by the color space in which shadow features are extracted. Groupings and evaluated algorithms can be seen in Table 2.2.

Sanin et al [8] proposed another taxonomy, which is mainly based on the features algorithms use for shadow detection. They grouped algorithms into four main groups,

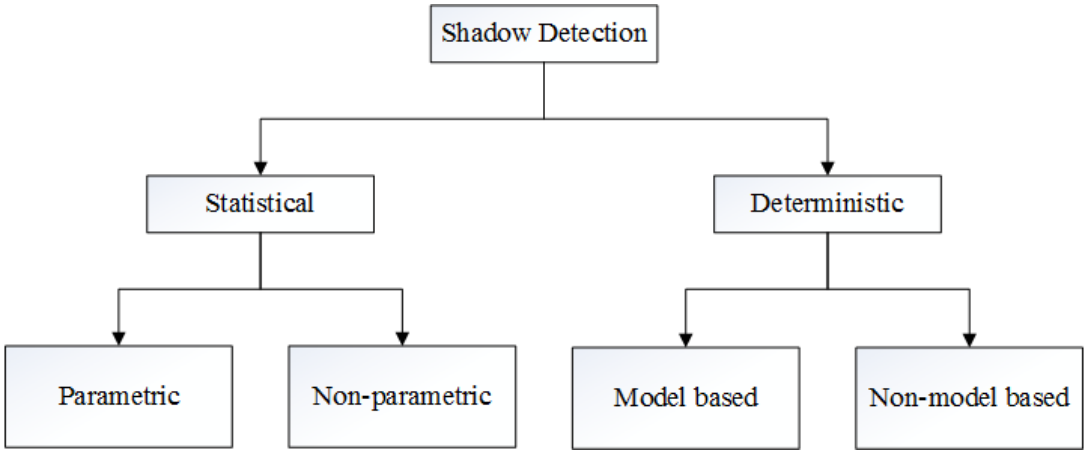


Figure 2.11: Taxonomy based on work of Prati et al [7]

as in Figure 2.12.

Table 2.2: Taxonomy based on work of Al Najdawi et al [2]

Layer 1 Object Dependence	Layer 2 Environment Dependence	Layer 3 Domain	Layer 4 Color Domain
Dependent	Dependent	Spatial	Color
			Monochrome
		Frequency	-
	Independent	Spatial	Monochrome
Independent	Dependent	Spatial	Color
			Monochrome
		Frequency	-
	Independent	Spatial	Color
			Monochrome
		Frequency	-

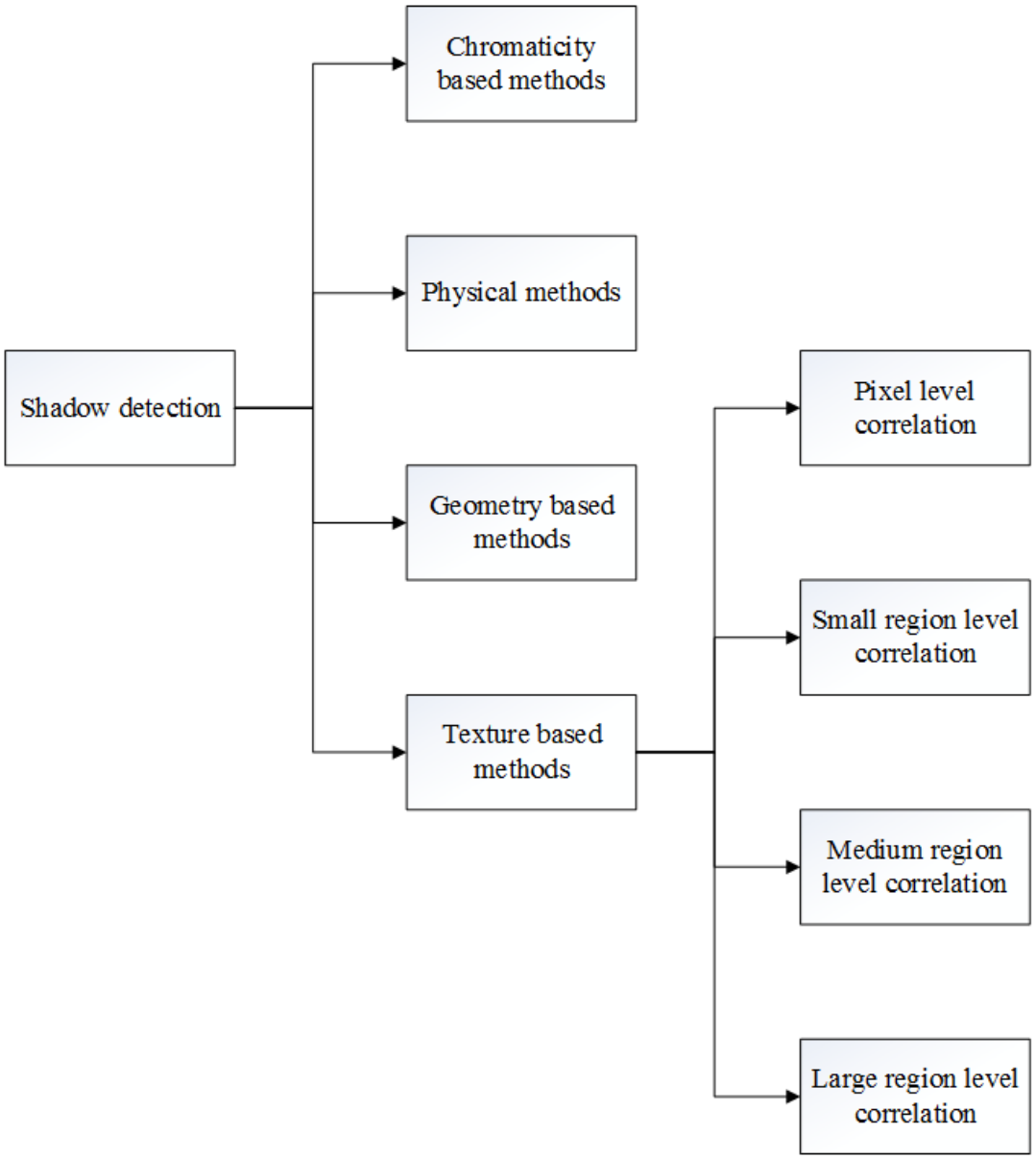


Figure 2.12: Taxonomy based on work of Sanin et al [8]

Taxonomy provided by Sanin et al [8] is used for classification of shadow detection algorithms evaluated in this thesis work. Performance and cost analysis based on feature types are easier, and algorithms based on similar features tend to produce similar quantitative results.

2.2.1 CHROMACITY BASED METHODS

Chromaticity is defined as a measure of color quality independent of luminance. Methods exploiting chromaticity information are based on the assumption that cast shadow on a surface causes a limited change in chromaticity, while changing intensity significantly. This assumption is called color constancy [63] or linear attenuation [64]. Based on this assumption, multiple regions are defined on 3D color space, according to their luminance difference and chromaticity similarity compared to a base color. For shadow/highlight detection, two different thresholds are defined [9], as in Figure 2.13.

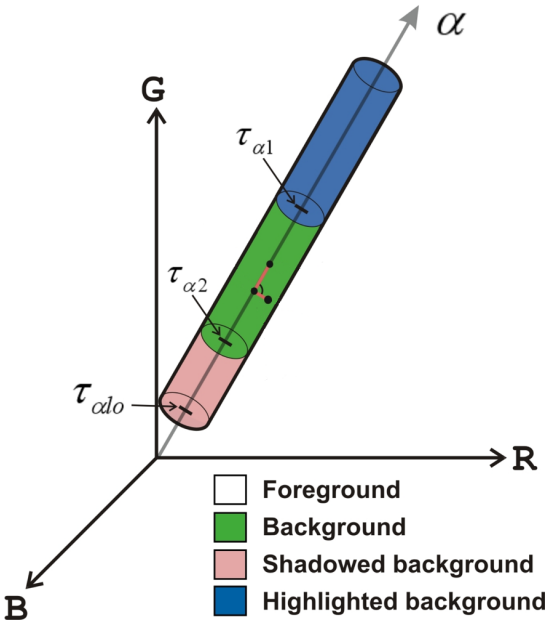


Figure 2.13: Chromaticity thresholds, retrieved from [9]

Another threshold, $\tau_{\alpha lo}$ is defined to avoid noise, when intensity becomes very low. All chromaticity based methods use this approach to identify shadow/highlight areas.

2.2.2 PHYSICAL PROPERTIES BASED METHODS

Chromaticity method is based on Linear Attenuation model, which assumes that pure white light is produced by illumination source. This is often not the case, as sun light greatly varies depending on time and location. Also, sky illumination is present in most cases, and can become more significant than sun when sunlight decreases. This can be observed as a shift of color in the shadow area, towards blue component [6]. To overcome this problem, a dichromatic model [64] is proposed. Later, non-linear attenuation models [64, 65] and introducing adaptability into these models [66–69] are proposed. Although these methods have better overall performance compared to chromaticity based methods as stated in [66, 67], they do not utilize any spatial information. Performance degrades when dealing with objects with similar chromaticity with background model [64].

2.2.3 GEOMETRY BASED METHODS

Geometry based methods utilize scene and object geometry to predict and detect shadows. Location of illumination sources, location and shapes of objects and orientation of ground plane are key properties utilized in these methods. By utilizing this information, several methods are proposed to detect shadows [70–74]. Geometry based methods do not require a background reference for detection of shadows. But they are limited to the scenarios where object and light source properties are known accurately. Therefore, they are not optimal solution to the shadow detection challenge in dynamic outdoor scenes, which are case for this thesis work.

2.2.4 TEXTURE BASED METHODS

Textures are strong features, and they retain their properties under illumination changes. Cast shadow on a surface changes intensity values at area significantly, while pixels under the shadow tend to preserve their spatial relationship with surrounding pixels. Therefore, correlation of texture descriptors becomes a strong feature for shadow detection.

Texture based methods generally consist two main steps:

1. Detection of shadow candidate pixels and regions
2. Classification of candidate areas using texture correlation

Shadow candidates are typically generated by using a lightweight shadow detector. For most methods, spatial features like chromaticity are utilized. This step reduces number of costly texture feature calculations and comparisons.

In second step, several methods are used for measuring texture similarity, like normalized cross-correlation [75], gradient or edge correlation [76,77], orthogonal transforms [78], Markov or conditional random fields [79,80], Gabor filtering [81]. While having a superior performance compared to other methods [8] , computational cost is increased by memory access patterns and calculations done in this step.

Texture based methods are further classified into groups based on the texture correlation level, as shown in Figure 2.12. Correlation level is an important parameter, as correlation area is proportional to the amount of spatial information used.

Test results provided by Sanin et al [8] indicate that, as correlation level increase from pixel to large region, shadow detection performance increases. Large texture based methods offer more independence from object, scene and shadow types. Also robustness to noise and penumbra detection increases.

2.3 OCCLUSION DETECTION AND HANDLING

Occlusion of vehicles is a challenging problem in Vision Based Intelligent Transport Systems. Unhandled occlusion cases deteriorate performance of key vision algorithms such as vehicle count, vehicle classification, lane change detection and violation detection. These algorithms are used to gather critical information needed for traffic management applications of ITS, therefore accuracy of these applications are critical. Also, as higher layer information such as travel speed, queue lengths, and traffic condition assessments are dependent on algorithms such as vehicle detection and classification, any inaccuracy in these algorithms propagates to higher layers. As

a result, various occlusion detection and handling algorithms are employed in ITS applications to improve performance.

Occlusion occurs when an object obscures view of another object from camera point of view. For most ITS applications, camera placement is done under physical restrictions such as height and available space for installation. There are few applications that provide top down view of the road surface and traffic, by setting up visual sensors looking down from a high platform. But due to restrictions imposed by urban environment, this configuration is not widely used, and not discussed considerably in ITS scope. Also for most applications, cameras are placed in such a way that all areas of interest are covered for different vision applications. Combined by these infrastructure related factors, urban traffic management applications mostly focus on intersections, where traffic density is significantly high compared to highway scenarios [82]. Reduced traffic flow speed and increased congestion triggered by high traffic density increase rate of occlusion events drastically. Variation of moving directions and paths of the vehicles, and increased interaction with surroundings in intersections further contribute to this rate [82–84]. Figure 2.14 shows images from most widely used visual sensor setup and configurations. As a result of these factors, occlusion cases between vehicles in areas of interest are mostly inevitable for higher layer applications such as vehicle counting, vehicle classification, vehicle tracking and event detection.

Occlusion events can be grouped into two cases, according to the proportion of blocked part of the occluded objects, which are partial and full occlusion. Full occlusion can be observed when an object is completely or nearly completely obscured by another object. In this case detecting or identifying occluded object is not possible without using extra information [13]. Figure 2.15 shows a full occlusion event with past and previous frames from same observation. Handling this type of occlusion is possible only when information from past or future frames is utilized, which is the case in many tracking applications. Another approach is using multiple sensors and cameras combined with stereo vision applications to detect and handle full occlusion cases [85], but due to increased complexity and costs, monocular vision based applications are preferred.

(a) Rectilinear

(b) Rectilinear

(c) Fiseheye

Figure 2.14: Occlusion cases from different camera setups

Second type of occlusion is partial occlusion, which occurs when occluded object is partly blocked. This means object can still be identified and distinguished visually from the blocking object using information supplied in current frame. This is the case mostly observed when a scene has occlusion event, as full occlusion cases are mostly temporary, and reduce into partial occlusion cases as observation proceeds. High traffic density and camera placement restrictions also increase rate of occurrence, as mentioned before. Also when using blob based moving object detectors, effects of moving cast shadows can generate effect of a partial occlusion on binary masks as in Figure 2.16. Partial occlusions can be solved by utilizing occlusion handling methods, which is the main focus of this thesis work.

Effects of occlusion on performance of vision based ITS algorithms are varying greatly. Some algorithms such as queue length estimation, vehicle presence detection or lane violation detection can be robust to these effects. But when vehicles are required to be detected and identified individually, occlusion becomes problem-

(a) Partial Occlusion (b) Full Occlusion

(c) Partial Occlusion

Figure 2.15: Progress of a full occlusion

atic. Overall performance of low level applications such as tracking, classification or counting deteriorates due to track losses, misdetections and misclassification. Performance of higher layer applications such as path tracing, event detection, flow analysis and traffic control systems are also affected, as they rely on low level applications mentioned.

To overcome occlusion problem and improve performance of vision based ITS applications, several methods of occlusion handling are developed. In early stages of development of ITS systems, several non-visual sensors are used for these applications, such as ultrasonic, infra-red and mechanical sensors. These methods are often invasive for transportation infrastructure and less accurate compared to vision based systems. Use of visual sensors in ITS became widespread, as sensor technology advanced and processing power increased [86]. Occlusion handling became an important and challenging part of ITS applications. While physical restrictions regarding

(a) (b)

Figure 2.16: Binary mask under partial occlusion

camera placement, and challenges in urban traffic scenes enforce an increase of complexity of proposed methods, requirement of real time processing of large amount of data constrains this complexity.

Occlusion handling methods are first addressed as a subset of vehicle tracking approaches [11], as solving occlusions require a priori information. This information can be object model, or past information regarding state of objects in the scene. Therefore, occlusion detection and handling methods are often combined and discussed with tracking algorithms. Main advantage of these approaches is availability of a tracker feedback, which can be used to solve full occlusions and partial occlusions that cannot be solved by current scene information. There are also different approaches, in which occlusion handling methods are used for accurate object detection and classification, without using any tracking algorithm. These algorithms use various object models, motion characteristics or certain object properties as a priori information to be supplied to occlusion handling algorithms. These methods often lack ability to detect and handle full occlusions.

Approaches in literature differ greatly depending on the problem definition and purpose of occlusion handling. Therefore, a distinct classification of occlusion handling methods is not possible. To classify and group proposed solutions, occlusion detection and occlusion solving approaches are examined separately as:

- Occlusion detection and reasoning

- Occlusion solving

2.3.1 OCCLUSION DETECTION AND REASONING

Occlusion reasoning is crucial part of any occlusion handling method. Detection results are used to trigger occlusion solving methods. Therefore, methods achieving this task need to be accurate, as any misdetection or false alarm affect success of whole application due to propagation.

Vision based ITS applications with occlusion handling solution are mainly focused on tracking algorithms. For ITS applications, occlusion is mostly discussed as a problem encountered in tracking algorithms, and often solved by incorporating concept of occlusion into the tracking algorithms [87]. There are also applications which use only current frame information to handle occlusions [14], but majority of works on occlusion handling uses either a tracking algorithm or an object model to associate objects in different frames in time. Therefore, it is necessary to examine these algorithms, which provide inter-frame information, in scope of occlusion detection.

Detection of occlusion event is based on certain features, which can indicate presence of occlusion when observed. These features vary depending on the purpose and scope of the framework in which occlusion handling is implemented. As mentioned before, these features may include results or observations from inter frame object association. In addition, occlusion can be detected without using any inter-frame information, using only information derived from current frame. In this thesis work, occlusion detection methods are grouped into two based on their time independence as following:

1. Inter frame level occlusion detection

2. Intra frame level occlusion detection

2.3.1.1 INTER FRAME LEVEL OCCLUSION DETECTION

INTER FRAME OBJECT ASSOCIATION / OBJECT MODELS

Visual object tracking problem is discussed in many ITS applications. While some higher layer applications such as path tracing and speed measurements are mostly dependent on a tracker algorithm output, applications like classification or vehicle counting are not fully dependent. But for increasing consistency and accuracy of results, object association with tracking or association based on object model is used. In this thesis work, tracking algorithms or object models are not examined in depth, but their role and efficiency in occlusion detection are to be discussed.

Object association is a critical step in occlusion handling, as inter frame level occlusion features can only be extracted by evaluation of object behavior on time axis. Abrupt changes in object models or properties, split merge events are used as features of inter frame occlusion detection.

Most preferred object association method is using tracker feedback. While there are wide range of trackers available, use of blob matching is most common. [83] uses a blob matching algorithm with using an undirected bipartite graph to represent object associations. Vertices in one partition of the graph belong to previous frame blobs, where other partition holds vertices corresponding to current frame blobs. Blob matching is done by finding nearest neighbor or nearest with average centroid. Similar graph based association is also done in [88], but instead of using blob centroid distance as a matching feature, overlap between bounding box areas are used. In [16], a feature vector is extracted from shape properties of extracted blobs like area, perimeter, centroid and orientation. These feature vectors are used in a fitness function, to match with previous frame blobs. In [89] blobs in consecutive fames are linked based on object trajectories, with imposing a path and shape coherence constraint. A similar approach based on blob trajectories with observing change in object shape properties is used in [15]. In [17], blobs are first segmented by subtractive clustering of motion vectors. Then these objects are matched by applying separate thresholds on area change, and deviation from estimated position. Srinivas et al. [87] use a nearest neighbor matching with estimated position, while maintaining a Kalman filter using velocity and position information for estimations. Work proposed in [90] has similar

approach using Kalman filter for position estimations, but introduces road geometry to increase accuracy. In [91], a distance matrix constructed by calculating Euclidean distances between centers of bounding boxes is used. Then a correspondence matrix is calculated, by matching objects with minimum distance, between frames. As this method cannot handle split and merge situations, a merge-split detection procedure based on this correspondence matrix and bounding box areas are developed.

Frameworks proposed in [11, 12, 92] use a generalized deformable model to represent objects. In these works, objects are first detected and segmented. Results of these stages are used for generating an object model for each detected object. For object association, object generated model is checked for any match with models from previous frames. In [10], a rectangular region detection method is used for object detection. Then, Kalman filtering is used to compute associations between detected objects. [93] combines appearance model of detected foreground objects combined with a bounding box distance measure. This distance measure is introduced to avoid abrupt distance change during merge and split events, when using centroid distance as a measure.

Part based models are also used to detect and track vehicles. In [94], a hybrid image template is applied to part based model. Combined with this part modeling, location and scale information is used to construct vehicle model and tracking.

Wu et al. [95] extends conventional Histogram Of Gradients (HOG) into Relative Discriminative HOG (RDHOG) to represent and track vehicles. For estimation of tracking state variables, a two stage particle filter based on two-scale RDHOG is used. Template matching is used to match objects from consecutive frames, based on Bhattacharyya distance of HOG descriptors. Pan et al. [96] also use template matching, but to avoid erroneous matching due to occlusion second stage - Variant Mask Template Matching (VMTM) - based on occlusion analysis Content Adaptive Progressive Occlusion Analysis (CAPOA) is used. In [82], a Spatio-Temporal Markov Random Field Model is utilized to determine state of each pixel. Instead of classifying each pixel independently, algorithm labels blocks of 8x8 pixels.

INTER FRAME LEVEL OCCLUSION DETECTION FEATURES

Inter-frame occlusion detection features can be generalized as state transitions and

events related to associated objects. In this level, occlusion reasoning is mainly dependent on the relation of the objects from consecutive frames.

Most common feature extracted from tracker feedback for occlusion detection is merging and splitting events. A merge event means more than one object is present in a tracked entity, which could not be separately associated into different objects. Presence of occlusion can be deduced from these events, as this type of failure in association can only occur at the starting instant of an occlusion event. A split event means multiple objects in current frame are originated from same tracked entity, and yields information about presence of occlusion in previous frames. Also, state of tracked objects are maintained as state vectors or graph representations, using merge and split event information. Blob matching based association approaches generally provide this information. In methods described in [87, 88, 90, 91, 93], these events are only feature for inter-frame occlusion reasoning. Although a significant feature, methods combining these with more information yield more successful results. [83] Combines merge-split event information with a shape estimator to detect occlusion events. Relationship between blobs from consecutive frames is represented in a bipartite graph. Size increase or decrease in a tracked blob is also used as a feature for occlusion detection.

There are also several methods, which do not rely on information on merge or split events. [15] Introduces trajectory based occlusion event prediction. For each object, trajectory is estimated using Kalman Filter. If distance between estimated center points of two objects becomes smaller than a certain threshold, then an occlusion event is predicted. Trajectory based decision is also supported by observation on change of object mask area. In [82], 8x8 image blocks which are labeled as object blocks are subjected to motion vector analysis. Any difference in motion vectors indicates presence of different objects, with occlusion. [96] Uses backward motion estimation to detect any other object interferes with the Region Of Interest (ROI) of another object.

Association failures between objects from different frames are also used as a feature for detection. Occlusion is considered as reason of failure to match object with any present model, or object. An example of this approach is [16], where objects without

any match with any object from previous frame are assumed to be in partial or full occlusion. In [17], each object is applied a backward and forward matching operation. Any matching failure is assumed to be caused by occlusion. Also subtractive clustering [10, 17] is applied on motion vectors of each object mask. Here, it is assumed that if there is no occlusion, number of clusters should be 1. This value is expected to be more than one in this method, as occluding objects are assumed to have different velocity on image plane, and different motion vectors. [89] Uses similar approach with motion vectors, but supports this method by keeping a trajectory for each object and observing any discontinuities.

Model based tracking approaches also provide features for detection of occlusion in inter-frame level. [11] Uses changes in generalized deformable model parameters as detection feature. In this work, an area ratio is defined and used as in Equation 2.16

$$R_{area} = \frac{Area_{mask}}{Area_{model}} \tag{2.16}$$

For each frame, R_{area} is calculated, and together with width, height and length parameters of deformable model, is used as feature for detection. Main assumption here is, difference between these parameters in consecutive frames become significant, when an occlusion event occurs. Detection is based on this observation, and a threshold is applied to each parameter to permit small changes.

[92] also uses generalized deformable model. An area ratio is also defined here, but different from [11], is the ratio between area of blob and area of its convex hull. For each blob, this value and fitted 3D dimensions of generalized deformable model are maintained. Change in these parameters is observed as a feature for occlusion detection.

2.3.1.2 INTRA FRAME OCCLUSION DETECTION

Intra frame level occlusion handling uses features based only on current frame information. These features are based either on object shape properties, or fitting results of a predetermined object model, as there is no information regarding to behavior of object on time axis in intra frame level.

FEATURES BASED ON SHAPE PROPERTIES

Most of the approaches in literature focus on shape properties of objects, based on certain assumptions as a priori information. These features are extracted from output masks of moving object segmentation functions.

[15] Applies a threshold to object size, and deduce if there is more than one object in segmented blob. Weakness of this method is requirement of accurate object size estimation. Also when variance of vehicle size in traffic is taken into consideration, this method can produce erroneous results with single hard threshold. Also, applicability of this method is further restricted to the scenario in paper- which is observation of scene using a far positioned rectilinear camera. Any application with close positioned rectilinear camera, or a fisheye camera would not produce accurate results, as effects of perspective and fisheye projection further increases variance in object sizes.

[14] Uses binary mask of the foreground objects to create a statistical graph for each blob. This statistical graph is formed by accumulating foreground pixels for each row, and storing these values indexed by row number. Then, a search for discontinuity between consecutive rows is conducted to detect occlusion. This approach can segment two vehicle occlusions, but more complex cases such as more than two vehicles or vehicles with different shape properties induce unpredictable changes in statistical graph.

Methods described in [13, 16, 17] introduce a different and more complex shape feature for intra frame occlusion detection. These methods come up with assumption that, unoccluded vehicle shapes tend to be convex, where objects shapes in partial occlusion tend to be concave. Considering vehicle shapes and possible camera setups in ITS applications, this assumption is reasonable.

In [16], convexity of an object is measured by how well it fits into convex hull constructed from object binary mask, where convex hull is defined as the binary mask constructed from minimum convex set of points to cover object binary mask. A convexity metric is defined as in Equation 2.17, where $Area_v$ represents the area of the object, and $Area_{ch}$ represents the area of convex hull of the object.

$$C = R_A = \frac{Area_v}{Area_{ch}} \tag{2.17}$$

This convexity measure converges to 1, when object mask is perfectly convex. For

occluded objects, area of convex hull increases and this measure becomes less than 1. In [16], presence of an occlusion is detected by applying a threshold, Th_{RA} to this measure. [13] uses same convexity metric, but introduces two classifiers to minimize overall error an overall risk. To minimize overall error, a Bayesian Minimum Error Classifier is used. By comparing probability distributions as in Equation 2.18 partial occlusion is detected.

$$p\left(PartiallyOccluded|C\right) > p\left(Non - PartiallyOccluded|C\right) \tag{2.18}$$

To minimize risk by identifying and segmenting an object as a partially occluded object, a Bayesian Minimum Risk (BMR) classifier is used. Two actions are defined as:

1. Classification as partial occlusion and segmentation.
2. Classification as full or no occlusion. No segmentation.

Then decision based on risk is made as follows: If

$$R\left(A1|C\right) < R\left(A2|C\right) \tag{2.19}$$

Object is classified as partially occluded, where

$$R\left(A1|C\right) = 2 * p\left(NPO|C\right)$$
$$R\left(A2|C\right) = p\left(PO|C\right)$$

[17] also uses assumption of convexity as in [13, 16], but uses different metrics to decide how much object mask is deviated from its convex hull. A quantity called compactness is introduced as in Equation 2.20, where BL is boundary length of object, and AO is area of object.

$$\Gamma = \frac{BL^2}{AO} \tag{2.20}$$

Γ is computed for object as Γ_V and its convex hull as Γ_C. Then a ratio between these two is defined as compactness ratio, Γ_R:

$$\Gamma_R = \frac{\Gamma_C}{\Gamma_V} \tag{2.21}$$

Also Interior Distance Ratio (IDR) is defined to express deviation from convex hull independent of area ratio as in Equation 2.22, where MA is length of minor axis of

current blob, and $Dist_M$ is the maximum Euclidean distance of pixels on blob contour to closest pixel on convex hull contour.

$$IDR = \frac{Dist|M}{MA} \tag{2.22}$$

IDR with Γ_R is used as a feature for occlusion detection. IDR tends to result lower for unoccluded vehicles, as convex shape of the blob is not distorted much. Γ_R tends to result close to 1 for unoccluded vehicles, as Γ_C and Γ_V becomes equal for perfectly convex blobs, where Γ_V is larger for occluded vehicles, as border length increases.

[12] uses camera parameters and 3D deformable model extensively, for occlusion detection. Combining road geometry and camera parameters with fitted deformable model, contour of object mask is analyzed to detect number of objects included in blob. In this work, model parameters are not directly used as a feature, while curvature of the object contour is used as main feature for occlusion detection.

FEATURES BASED ON MODEL PARAMETERS

Object models are used in many applications requiring tracking or classification tasks. For these applications, object models are constructed and are used for detection of objects of interest. In intra frame context, use of adaptive models as used in kernel based trackers are not possible, as this model is constructed and updated using inter frame information. Therefore, only training or rule based models are applicable.

Part based models are used in many object detection and tracking applications. Generally, part based models are used for tracking by detection frameworks. [97] uses part based representation for detection of occlusion in a human tracking application. This segments human object into three segments and detection of a human is achieved by successful association of detected segments. [94] applies similar approach to ITS domain, with several adaptations. Front views of vehicles are used, as this is the most common case in ITS applications. Parts are selected according to their visibility during occlusions, and information they contain regarding to vehicle class. Method divides each object into two parts representing occluded and unoccluded regions. Considering area around license plate is easily occluded, and area around front window is unoccluded most of the time, these two areas used as parts. This way, loss of easily occluded part can be compensated by detection of unoccluded part. Also occlusion events are detected, if there is any missing part.

Another approach is use of object models constructed by rules or training. In [10], vehicle templates are constructed from 3D vehicle models. These templates are used for matching, and objects are detected. Occlusions up to a certain level can be handled by this approach, as long as object can still be matched to a 2D template.

In [98], a generalized deformable model is fitted to each object. In this work, it is assumed that, in presence of occlusion, ratios between model dimensions are substantially different from the norm. Also, area ratio defined in [11] is calculated. While occlusion is not present, this parameter is expected to be closer to 1. Deviation of this parameter from norm is used as a detection feature.

2.3.2 OCCLUSION SOLVING

For occlusion solving, methods are wide range as in the case of occlusion detection. Works on this field are mainly shaped by requirements enforced by higher layer applications. A vehicle classification algorithm often requires more accurate information of object location and shape, compared to a vehicle counting algorithm. For a counting algorithm, it is often expected to handle full occlusion cases, during which a classification is not possible. Therefore, these approaches can be grouped into two groups as following:

1. Partial occlusion solving method
2. Full occlusion solving method

2.3.2.1 PARTIAL OCCLUSION SOLVING

Partial occlusions are main concern of most of the work in literature. As majority of occlusion cases are partial occlusion, and frequency of this event in urban traffic scenes are high, there are many approaches discussing possible methods for partial occlusion handling.

During partial occlusions, occluded object is not totally blocked, and features obtained from unblocked part are sufficient for detection or identification of the object.

Therefore, solutions for partial occlusion problem are developed as part of tracking classification and counting applications in ITS domain.

For partial occlusions, both inter-frame level and intra-frame level features are used extensively. While some methods only maintain identity of occluding objects, majority of the methods examined aims to segment objects from each other. Main reason behind this difference is scope of the framework occlusion handling is done. For example, simple tracking tasks and counting can be achieved by using only vehicle count and identities on a blob, while more complex applications such as classification, path tracing, and violation detection may require segmentation of occluded objects with accurate borders.

MAINTAINING OBJECT IDENTITY

Some work in literature are focused only on eliminating negative effects of partial occlusion, without solving problem without finding identity or object boundaries. One example for this type of solutions is discussed in [83]. In this work, two states are defined for tracker as tracking and prediction only modes. When occlusion occurs, instead of identifying and separating occluding objects, method treats occlusion as an absence of measurement, and measurements are ignored for position and shape filters used for tracking. This way, erroneous object data is disregarded, and object models are kept accurate through occlusion events.

For applications without accurate border extraction requirement, finding and maintaining object identity is sufficient. These methods are often regarded as a solution to problem of tracking objects through partial occlusions. When a partial occlusion occurs, these methods are utilized to ensure tracker has sufficient information of object identities, and continue tracking object until object leaves scene, or is no longer object of interest. Also, erroneous data is disregarded in most of these methods, which increases accuracy of object models or object parameter estimations such as motion characteristics. One example for this approach is [87]. In this work, all objects are associated with a track as T_a, T_b etc. Then for each possible merge event, combination of objects is decomposed to multiple tracks. Then one to one correspondence is checked for each track. This way, correct tracks are matched to each object after split events, by maintaining identity and parameters of occluded objects during

combined track. [91] Also uses a similar approach. Difference from [87] here is in matching procedure after split events. In this work, color distribution of objects is used as a correspondence feature. This information is added into the occluded group, and used when any split even occurs. Distance between two color distributions are calculated using Kullback-Liebler (KL) distance during process. [90] Uses motion estimation based on a Kalman filter to correctly match splitting object. During occlusions, Kalman filter parameters are not updated for occluded objects. Instead, a new Kalman filter is constructed for occluded group and matching is done based on this filter estimation. Split events are handled based on the estimations from filters stored before occlusion event. Another similar approach discussed in [88], maintains identity of tracked objects using an association graph for connected components. This work utilizes a two level tracking algorithm, to increase tracking accuracy through occlusions. First stage is vehicle level tracking. In second level, instead of treating each connected component as an object, method treats these components as regions of tracked objects. Each region is associated to best matching object by evaluation of motion characteristics, to segment occluding vehicles.

MODEL BASED METHODS

Object models are also used as a feature for partial occlusion solving. Different object models are employed for ITS applications, especially for tracking and classification frameworks. In [94, 98] part-based object models are used for object classification in congested traffic conditions. These methods are based on identification and classification of vehicles from their visible parts. This enables framework to process occluded vehicles, if sufficient part information is obtained from current frame. Therefore, occlusion is handled without use of any segmentation approach.

Object models based on vehicle shape are also widely used for occlusion handling purposes. A 2D object model is used in [10], for detection and tracking of the vehicles. In this work, videos from a single calibrated camera placed at a few meters above ground are used. 2D projections of 3D vehicle model in different camera tilt angles and object orientation angles are constructed as in Figure 2.17. In this work, occlusion handling is achieved indirectly, by achieving accurate model match during partial occlusions up to certain level.

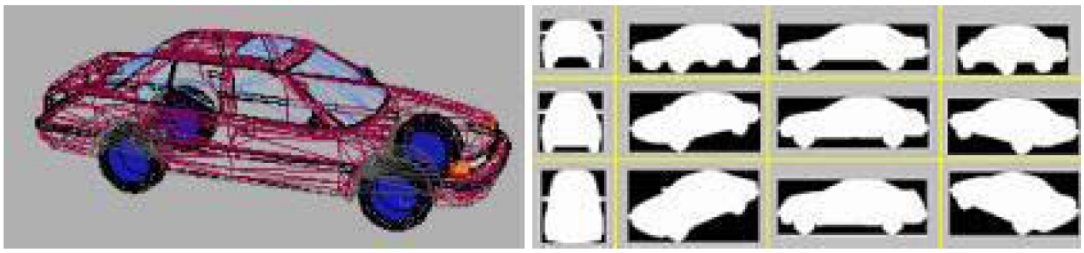


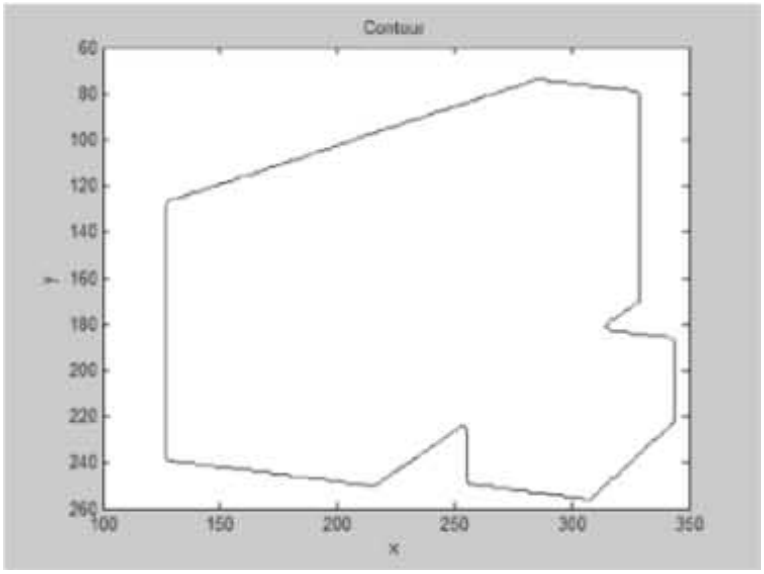
Figure 2.17: 2D vehicle model used in [10]

[93] also uses model matching to handle partial occlusions. An appearance based model is used with split-merge information. When a merge event occurs, appearance models of merging objects are used to estimate location and depth ordering of each object. This is achieved by using a maximum likelihood classifier for each pixel to determine which model is most likely to have produced that pixel.

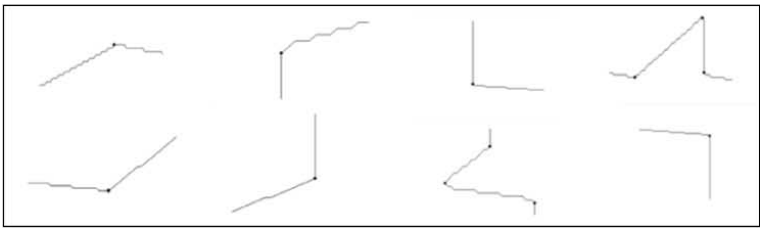
Another model based approach is use of generalized deformable models for occlusion handling. [11, 12] are examples of this approach. In this approach, a Generalized Deformable Model (GDM) is fitted into each connected component. Then this model is decomposed to individual objects, using different segmentation and decomposition methods. In [11], contour of the fitted model is analyzed for decomposition. Curvature points are extracted from the filtered contour of the model, as in Figure 2.18. Then model is decomposed into 2D projections of modified 3D rectangular prisms. A 3D model and position information for each object in occluded group are obtained as in Figure 2.19. [12] Improves this method by implementing projection of GDM into 2D using camera and road parameters. Also, a resolvability index is calculated for each occluded group. Then for each vehicle in group, GDM is resolved by recovering missing lines of vehicle model geometrically, using the non-occluded lines. A result of this algorithm can be seen in Figure 2.20.

CUT BASED METHODS

For applications with accurate object segmentation requirement and low execution time, cut based object splitting methods are proposed. These methods are widely used for intra frame object segmentation. For ITS frameworks, which involve foreground detection and object mask generation at any point of framework, these methods are



(a) Borders of 3D model projection



(b) Detected curvature point on contour

Figure 2.18: Extraction of curvature points , retrieved from [11]

preferred as they have low computational cost compared to model based approaches. These methods mostly use 2D geometric properties of the occluded object masks for segmentation. A cutting line or cutting region is calculated to separate and isolate occluding objects. Most of methods based on this approach in literature, depends on some geometrical assumptions regarding to object shape. Also, none of the works examined for this thesis work, proposes a solution to handle cases involving vehicle groups with more than two occluding objects.

Several methods and features are discussed to determine cutting line or region. A cutting region is defined in [89], based the critical assumption that if two vehicles are occluding, they must have different speeds. Therefore, if variance of motion vectors in a region is greater than a threshold, then motion vectors inside an object can be clustered into two regions. Then dilation operation is applied on these two regions,

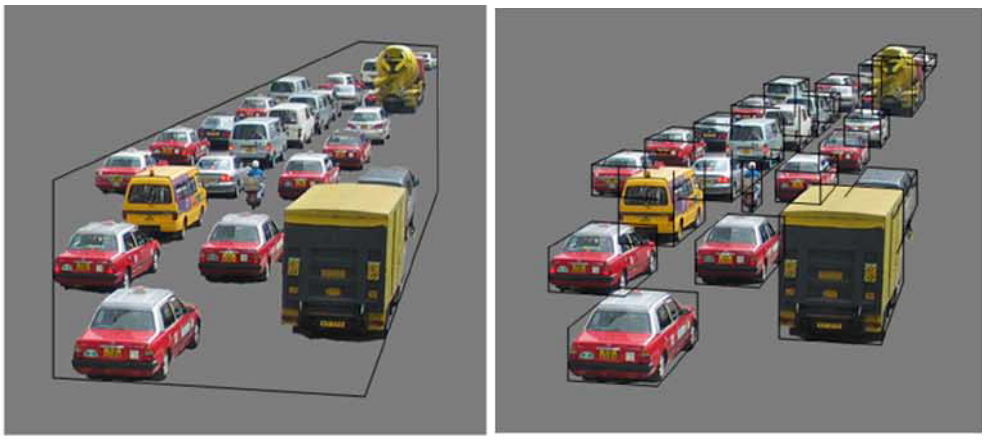
(a) Original Image (b) Segmented Objects

Figure 2.19: GDM based segmentation, retrieved from [11]

and overlap area is defined as cutting region Figure 2.21. This method is different from other cut based methods discussed in this work, as it utilizes motion vector information instead of geometric properties of object mask.

Methods based on a cutting line are more common in literature. Several methods are discussed in literature to find two points forming the cutting line to separate occluding objects. These methods are using shape and contour of the object blob as feature. A statistical graph is formed in [14], based on horizontal projection of object mask Figure 2.22. Then graph is checked for any sudden increase or decrease. Points defining cutting line are chosen on object contour according to this observation. In [15], three types of cutting lines are defined as vertical, horizontal and diagonal. Approach here is to find cutting line with minimum length, if occlusion is detected. To find vertical line, on each contour point, vertical distance of contour to top and bottom sides of bounding box are calculated. A vertical line at minimum of this value is defined as cutting line Figure 2.23. Similar approach is used for horizontal and diagonal lines.

Assumption that object shape is convex unless object is occluded, is used also for solving partial occlusions, as it was used for detecting if occlusion is present. In works, which use this assumption, dividing points are selected as points on object contour, at which deviation from convex hull is maximum. Deviation is evaluated differently at each work. In [13], deviation is considered as distance from convex hull. After convex hull and difference image is calculated as in Figure 2.24, two



(a) Detected Group of Objects (b) Segmented Objects

Figure 2.20: GDM based segmentation in a crowded scene, retrieved from [12]

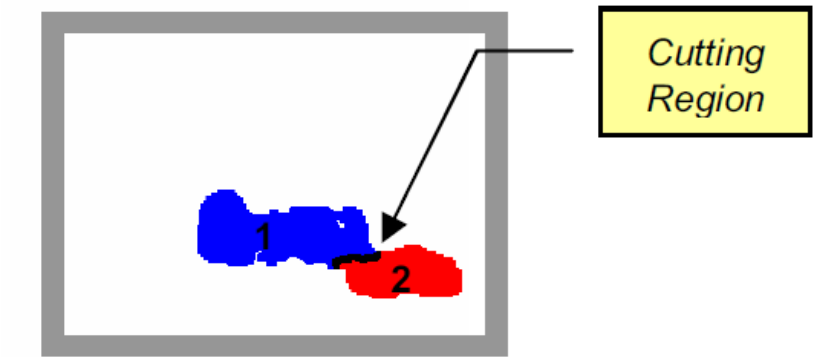


Figure 2.21: Formation of cutting region, retrieved from [13]

largest parts of the difference image is selected. Then for this two parts, vertices of convex hull is calculated. For each set of vertices, vertex with maximum distance to object convex hull is selected as cutting point. After points are determined, cutting line is removed from object mask. [17] uses a similar approach. First, a single cutting point is defined as in Figure 2.25. As in [13], selection is based on the distance of point from object convex hull. Then for every tilt angle, cutting lines are formed and convex distance Con_D is calculated for split objects as in 2.23, where IDR is defined in Equation 2.22, and Γ_R is defined in Equation 2.21. By minimizing this parameter, cutting line is found.

$$Con_D = IDR^2 + (1 - \Gamma_R)^2 \tag{2.23}$$

(a) Binary Mask (b) Statistical Graph

Figure 2.22: Calculation of statistical graph, retrieved from [14]

Considering computational cost of method discussed in this work, a different approach is proposed in [16]. In this work, it is assumed that maximum deviation from convex hull is observed at the points at which curvature function given in Equation 2.24 is equal to zero. Object contour is first smoothed to avoid false positives. Smoothing is applied until exactly at two points curvature function results zero. Then these two points are selected as vertices of cutting line Figure 2.26.

$$\kappa(u) = \frac{\dot{x}(u)\,\ddot{y}(u) - \ddot{x}(u)\,\dot{y}(u)}{\sqrt{\left(\dot{x}(u)^2 + \dot{y}(u)^2\right)^3}} \tag{2.24}$$

2.3.2.2 FULL OCCLUSION SOLVING

Full occlusion solving methods are generally dependent on inter-frame occlusion detection features, as occluded object is blocked and no feature can be extracted during full occlusion. As no feature from the object can be obtained from current frame, these methods are used to maintain identity, or detect presence of occluded objects.

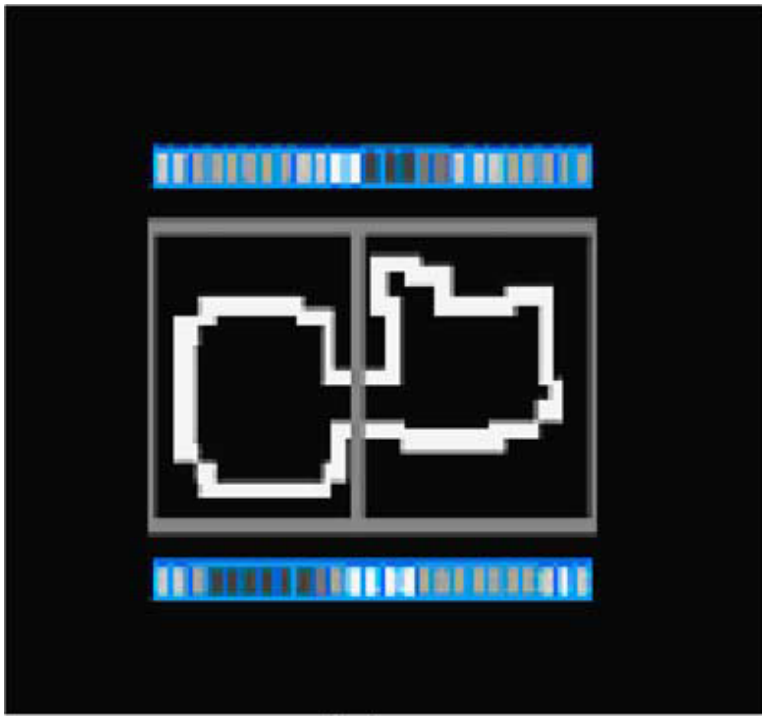


Figure 2.23: Vertical cutting line, retrieved from [15]

Figure 2.24: Convex hull and cut point extraction, retrieved from [16]

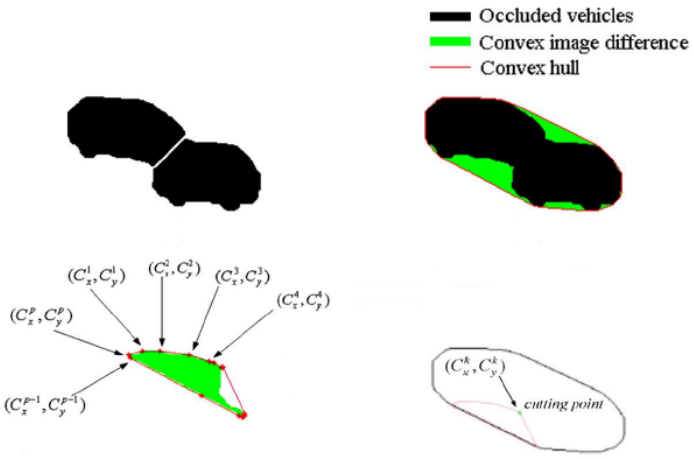


Figure 2.25: Iterative cut progress, retrieved from [17]

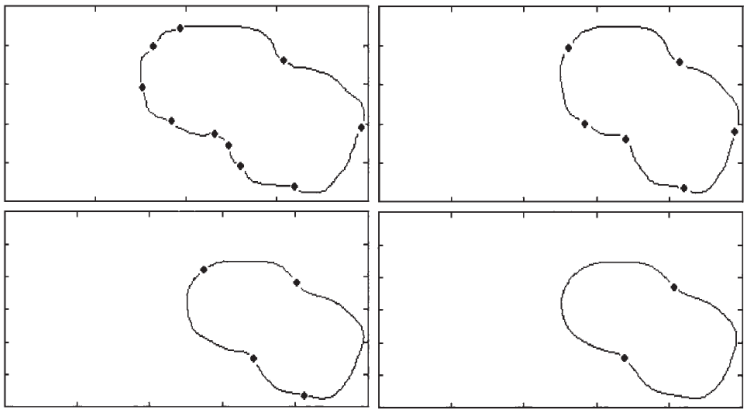


Figure 2.26: Zero crossings of curvature points, retrieved from [16]

CHAPTER 3

PROPOSED SOLUTION

In this chapter, a multi stage framework to detect vehicles at intersections automatically in presence of occlusions is proposed. Design steps and implementation details for the proposed solution are presented.

Performance issues related to the occlusion presence in ITS applications are discussed in several works in literature. Decrease in accuracy of object detection and tracking propagates to higher level tasks, which are often statistical or event based data in commercial ITS applications. It must be stated that, the solution proposed in this work is restricted to object detection problem in occlusion presence. In this sense, tasks like tracking and classification are not in scope of this work, and will be considered as higher level tasks, as the proposed framework aims to increase performance of these tasks by providing accurate data on objects in scene.

Vision based ITS applications are mostly solutions proposed to handle subtasks of a larger traffic management task. Therefore, accuracy and reliability are key features, as risk of causing accidents or disrupting traffic flow imposes a low fault tolerance. Apart from offline analysis applications, which are often used for detailed analysis based on collected data, most applications related to management have a real time constraint. Real time applications must guarantee response within a limited time interval, controlled by deadlines [99]. Satisfying these constraints become more challenging, as many ITS applications are implemented on field hardware, which have computational power limitations due to costs and infrastructure limitations. A Speed-Accuracy Trade-off [100] must be considered, as both computational power and fault tolerance is limited. Proposed work in this thesis work is designed considering Speed-

Accuracy Trade-off. At each stage, methods to handle subtasks are evaluated for both accuracy and computational cost.

Another restriction on the scope of this work is on environmental properties. Low level vision tasks have restrictions and variations on performance, based on target environment properties. One example can be failure of background subtraction based algorithms, when scene is observed with a non-stationary sensor unit. Therefore, the solution proposed in this work is designed based on a widely utilized surveillance scenario in ITS applications. Stationary cameras installed above road level, overlooking intersections are used.

In an intersection, there may be many different objects while object we are interested in are vehicles, which have different movement patterns depending on traffic flow. Static objects either belong to scene, or out of interest when tasks like tracking or counting are performed. Therefore, moving objects are objects of interest for our case, and their detection is a crucial step. Static camera placement and Speed-Accuracy Trade-off must be considered at selection of a reliable object detection method. Due to their low cost and parallelization potential, moving object detection based methods have significant advantage compared to model matching based methods. Also, modularity must be considered as object detection is generally used as first stage of most applications. MOD based methods have better modularity, as they do not require a predefined model, training set or template set.

Moving object detection becomes a challenging task, when illumination changes, shadows or irregular movement patterns are present in the scene. Also, object detection errors propagate to higher level tasks, reducing overall performance of these tasks. Once pixels belonging to moving objects are detected, results must be refined using different methods to handle errors introduced by these properties of objects and scene.

Shadow and highlight effects are significant in urban intersection environments, due to presence of different light sources, and objects blocking illumination in some regions of the scene, like trees, poles or buildings. Also cast shadows are present generated by both sun and lighting installations. Therefore, a method for shadow/highlight detection and removal, or an object detection method resistant to these effects is

needed to increase accuracy.

Detection of objects from mask of moving objects is the step where erroneous outputs due to occlusion are generated. Occlusion cases as in Figure 2.16 are common in urban intersection environment, and a connected component analysis based on foreground mask can cause misclassification of multiple vehicles as one vehicle. Therefore occlusion must be detected and handled to improve detection performance.

In Chapter 2, several different approaches are examined, and grouped according to features and methods they exploit for detecting and solving occlusion events. Considering cost and improvement in detection performance, improved occlusion detection and segmentation methods are proposed.

In Figure 3.1 modules and input/output relations of these modules are shown with low detail. Proposed method first uses a Vehicle Detection module based on moving object detection, to find object candidates. Then Occlusion Analysis is performed on candidate objects, with using association data and geometrical properties of objects. Based on analysis result, Segmentation is done, and association object data is updated.

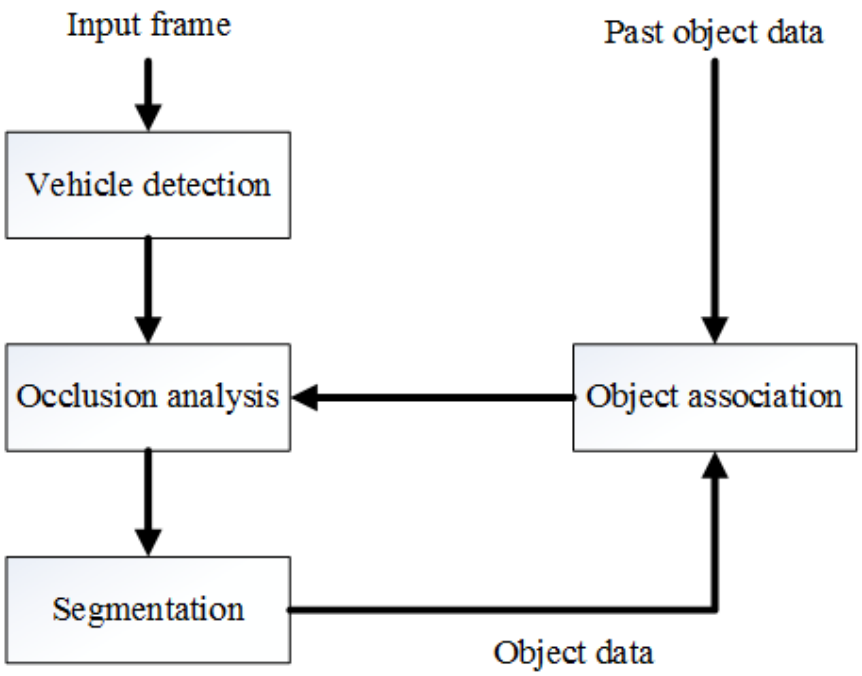


Figure 3.1: System overview

In following sections, these stages will be discussed in detail and methods in literature will be evaluated. For Occlusion Detection and Segmentation, improved methods based on the selected methods from literature will be introduced.

3.1 BACKGROUND SUBTRACTION

In this section, a method to generate accurate foreground mask is proposed. Main objective of this stage, is to provide stable and accurate mask of the pixels belonging to the objects of interest in the scene.

As the traffic flow in an intersection is highly variable, vehicles have often irregular movement patterns. These patterns include cruising, waiting, moving slowly, turning and different combinations of these behavior. Therefore, a vehicle does not necessarily move in whole duration of the observation. Therefore, background model generation is required for accurate detection. Although having lower computational cost and memory requirements, frame differencing based methods are ruled out, due to issues addressed in Chapter 2, Section 1. Figure 2.1 illustrates response of frame differencing to different movement patterns.

There are many methods for background subtraction, which are examined in detail at Chapter 2, Section 1.2. Selection of a base method for this work is done considering several aspects of examined methods, including qualitative and quantitative results.

As segmentation process, which will be explained in detail later in this chapter, utilizes object geometry, obtained from binary mask. Therefore, method is expected to preserve object properties. Two successful approaches, proven with extensive benchmarks [1], are implemented and evaluated. First method is SAGMM, which is based on parametric density estimation with various adaptation and learning improvements compared to base method GMM. Development of this algorithm is shown on Figure 2.6. SAGMM has a lower accuracy compared to recent non parametric based methods evaluated on CDNET [1]. However, computational cost is low, and there is a huge potential for parallelization, as method has a pixel based nature. This means, same operations are done for each pixel, while pixel each being independent from each other. This means, parallelization can be achieved with high speed-ups. Also,

memory access is contiguous, which means a coalesced memory access is possible. This is a major advantage, as high cache utilization is enabled, while time spent for data retrieval from memory decreases. This property of GMM is exploited in a CUDA based work [18], with a significant speed-up as shown in Figure 3.2. Considering this parallelization potential, SAGMM is considerably low cost, considering acceptable detection performance it offers.

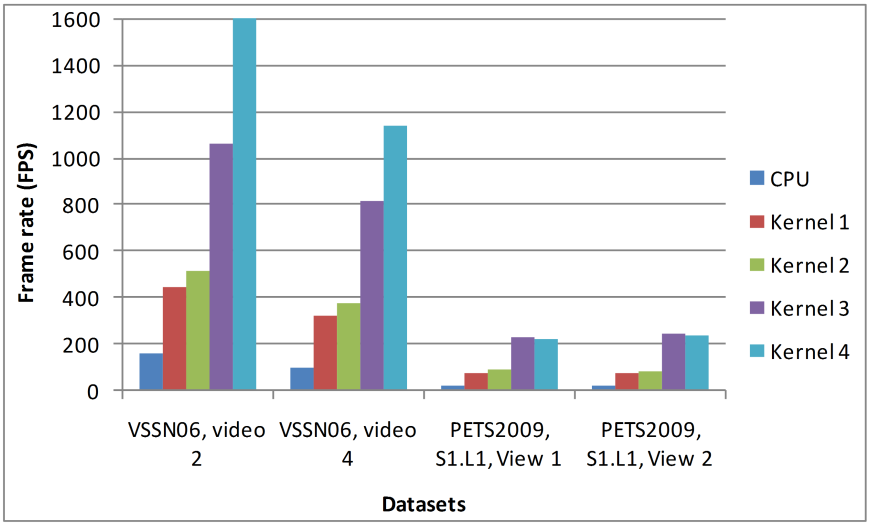
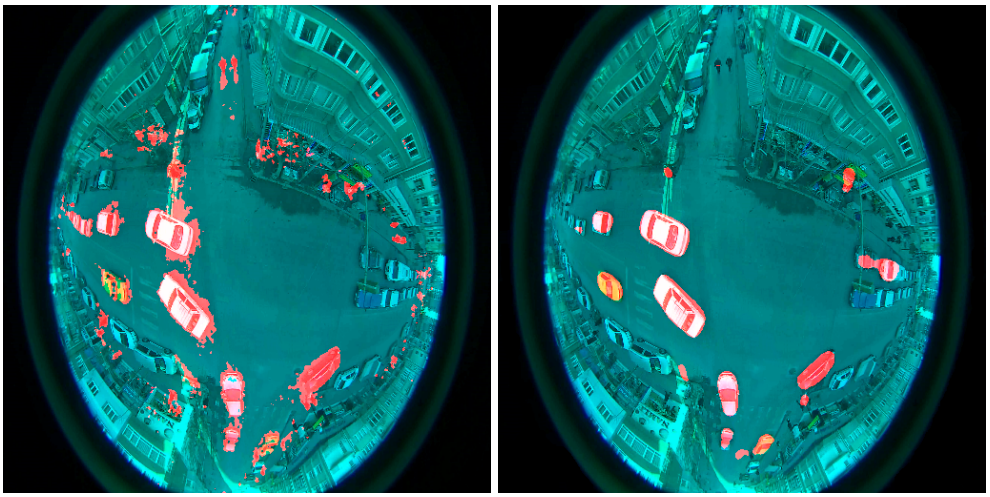


Figure 3.2: Speedups obtained using CUDA, retrieved from [18]

Second method evaluated for use in framework, is SuBSENSE. In benchmarks conducted on CDNET datasets of 2012 and 2014, method has proved itself with superior performance in most metrics and in different scenarios, compared to many modern background subtraction approaches. Although, computational cost is higher than most pixel based methods, as SuBSENSE utilizes LBSP descriptors, which are expensive to sample and compare. Also, sampling pattern shown in Figure 2.8 causes a reduced parallelization potential. This is due to non-contiguous memory access during sampling of LBSP, which causes non-coalesced access.

To decide the method which will be part of the framework, two methods are tested and results are evaluated quantitatively. As segmentation method is based on geometrical properties, high boundary adherence must be achieved. Therefore, outputs are evaluated to deduce which method produces binary masks that preserves object geometry most. Some preliminary results as shown in Figure 3.3 show one of the

cases which SAGMM performs poorly compared to the SuBSENSE.



(a) SAGMM (b) SuBSENSE

Figure 3.3: Preliminary results on background subtraction. Overlays of the foreground masks on the frame.

As SAGMM does not use spatial relationship between local pixels, some parts of the objects which have similar intensity values compared to the background model are misclassified as background. This effect can be observed in the wind shields and vehicle hulls which have similar RGB intensity values with the road surface at background. As shown in Figure 3.3, SuBSENSE overcomes this problem. LBSP descriptors used in pixel level contains local texture information, therefore unlike pixel based GMM, SuBSENSE is more robust to similar color values between background and foreground regions. Apart from that, as GMM is pixel level, due to eliminate errors, and to compensate missing local spatial information, some post processing steps are needed before using binary mask. These steps are median filtering of binary mask to eliminate single pixel foreground regions caused by noise, and morphological closing operation to connect erroneously partitioned blobs. Even after these steps, a higher boundary adherence is achieved with SuBSENSE.

Considering performance and cost of both algorithms, SuBSENSE is selected as moving object detection algorithm to detect vehicles. It must be stated that most important factor in this decision is the fact that Occlusion Detection and Segmentation methods, which will be described in detail in next sections, are highly sensitive to the errors in object geometry. Therefore, to prevent propagation of error to these stages, SuB-

SENSE is selected. Accuracy and stability advantage of SuBSENSE surpass lower computational cost offered by SAGMM.

3.2 SHADOW DETECTION

Negative effects of moving cast shadows were explained in Chapter 2, Section 2. As regions under shadow tend to preserve texture information, SuBSENSE is able to discriminate background and foreground robustly in soft shadow presence [4]. When shadow strength increase performance of both SAGMM and SuBSENSE degrade. SAGMM does not utilize texture information, therefore a significant intensity change caused by shadow leads to misclassification. For SuBSENSE, as intensity decreases to the point where LBSP based texture information loses accuracy, misclassification rate increases dramatically. Therefore shadows are misclassified as foreground, which can be seen in Figure 3.4.

To increase robustness against shadows, a shadow detection stage is proposed. A detailed performance evaluation of existing methods is done by Sanin et al [8] shows that texture based methods have better performance compared to other algorithms. Problem here is, as shadow strength increases, texture correlation performance also decreases. LBSP descriptor comparison actually exploits similar information with small region (SR) texture correlation. Therefore a large region (LR) texture correlation based method is required. The algorithm proposed in the survey [101] is evaluated and selected for shadow removal. This can be explained in 3 main steps:

1. Detection of candidate regions
2. Correlation of candidate regions with background model
3. Classification of candidate regions as foreground or shadow region

In first step, candidate detection is done on existing foreground mask based on a weak shadow detector. Chromaticity and intensity values of pixels are used for candidate selection. This step is expected to find largest possible regions with shadow candidates. The chromaticity based method Cucchiara et al [7] proposed is used, with

relaxed thresholds. This way, largest possible regions are tagged as shadow candidates. A connected component analysis is done on this mask, for defining regions. To avoid regions with both foreground pixels and shadow pixels, connected components are split from the edges which are present in foreground only.

In second step, textures corresponding to the each candidate region are correlated between frame, and background model. Texture correlation is based on gradient. To calculate gradient direction and magnitude, Equation 3.1 and Equation 3.2 is used.

$$|\nabla_p| = \sqrt{\nabla_x^2 + \nabla_y^2}\,, \tag{3.1}$$

$$\theta_p = arctan_2\left(\frac{\nabla_x}{\nabla_y}\right)\,, \tag{3.2}$$

Where ∇_y and ∇_x are directional gradients, and $arctan_2(.)$ is a modified $arctan(.)$ to allow gradient direction to be treated as a circular variable. To avoid noise, a gradient magnitude threshold is applied on each pixel. Then gradient direction difference is calculated between current frame and background ,as in Equation 3.3:

$$\Delta\theta_p = arccos\left[\frac{\nabla_x^F\nabla_x^B + \nabla_y^F\nabla_y^B}{\sqrt{\left(\nabla_x^{F2} + \nabla_y^{F2}\right)\left(\nabla_x^{B2} + \nabla_y^{B2}\right)}}\right]\,. \tag{3.3}$$

Where F and B are foreground and background respectively. Gradient direction correlation between F and B is estimated as in Equation 3.4:

$$c = \frac{\sum_{p=1}^{n} H(\tau_a - \Delta\theta_p)}{n}\,, \tag{3.4}$$

Where τ_a is angular difference threshold, and $H(.)$ is unit step function. c gives the fraction of pixels with similar gradient direction in F and B. At third step, this fraction is thresholded to decide if a region is foreground or shadow.

Resulting method for foreground mask generation is shown in Figure 3.5.

3.3 VEHICLE OBJECT DEFINITION

After foreground mask is obtained, a connected component analysis is done to detect vehicle blobs. A vehicle class with following properties is defined:

1. ID
2. Contour
3. Centroid
4. Area
5. Association Data

Each blob with area above a static threshold, is used to define a vehicle object. Contour and centroid of the blob are calculated, as they are required in Occlusion Detection and Vehicle Association.

3.4 VEHICLE ASSOCIATION

The association scheme proposed in this work is kept simple, as it is required to demonstrate performance increase using intra frame level occlusion detection. Occlusion detection and segmentation approach proposed in this work aims to improve performance of methods requiring accurate vehicle detection in any stage. This means, most tracking applications can benefit from utilization of this method, as they require detection of vehicles at track initialization stage.

Association provides information regarding presence of a merge event in a detected vehicle or vehicle group. Also, by keeping track of split and merge events in lifetime of each object, estimation of number of vehicles in a group is provided. A foreground object with multiple vehicles means these vehicles are occluded, partial or full.

Association block in this framework can be replaced, as there are wide range tracking algorithms, which can generate similar information with greater accuracy. Applica-

tions depending on blob trackers [83, 102] are most suitable for this purpose, as they generate blob based object data.

For association scheme in this work, following statistics are calculated and kept for each object group:

1. Contained Vehicle Objects (Members)
2. Position and velocity estimates for each member
3. Merge or split event data

A two stage association method is implemented. First, a position estimation is done for all vehicles from previous frame. In this step, estimation is done using last known position and velocity. Velocity is assumed to be unchanged in consecutive frames, and position estimation is done based on this assumption. However, estimate error increases, if object is occluded for long time intervals. A validity flag is used for each parameter. This flag is set to false, if vehicle is occluded more than a certain number of frames.

Matching is done if position estimate is valid, and is within a certain proximity of an object in current frame. Another proximity threshold is used to match objects based on distance between centroids, if velocity and position estimations are not valid.

Past frame objects without match are processed in second stage of association. In this stage, each unmatched past frame object is associated using similar proximity threshold and distance between centroids. There are five possible events which will be used as inter frame occlusion detection feature :

1. New Object : Unmatched objects from current frame object list are treated as new vehicles entering the scene. A new vehicle object is generated.
2. Missing Object : Unmatched objects from past frame object list are treated as vehicles leaving scene or failed detections. Vehicles associated with these objects are not used in future frames.

3. Merge Event : If an object from current frame list is associated to more than one objects from previous frame list, then a merge event occurs. Positions and velocities of merging vehicles are recorded for use in position estimation.

4. Split Event : If an object from past frame list is associated to more than one objects from current frame list, then a split event occurs. Using estimated positions of vehicles inside object, each vehicle is assigned to the closest of the split objects.

5. No Change : No merge split event occurs, and object is successfully matched to a single object from past object list.

An overview of the object association stage can be seen in Figure 3.6.

3.5 OCCLUSION DETECTION

In this section, the occlusion reasoning method proposed in this framework is explained in detail. Methods in literature, often address occlusion problem in context of higher level tasks such as tracking or classification. Therefore, various methods with different features, performance and computational cost are proposed. In this thesis, as stated in previous sections, both performance and computational cost is considered during evaluation of existing algorithms, and design of the proposed solution.

Occlusion detection relies on both inter frame and intra frame information in most existing applications. This two set of features are independent from each other, and utilization of both is expected to improves detection rate. Therefore, two methods for each information set is proposed. Occlusion is detected, if any of these two methods result with occlusion detection.

The method proposed in this work, relies on the fact that if occlusion is present between two moving objects, then a single connected component exists in foreground mask, including both objects. It must be noted that blobs of multiple unoccluded objects may be connected due to misclassification of background pixels. Either way, segmentation is needed for splitting connected blobs, and both situations will be regarded as occlusion cases.

An overview of the proposed occlusion detection stage is shown in Figure 3.7, and details are explained in next sections.

3.5.1 INTER FRAME OCCLUSION DETECTION

Inter frame occlusion detection features are extracted from the relations of objects and motion characteristics obtained from between different frames. Most methods in literature utilize relations and events of objects in temporal axis.

As blobs with multiple objects are regarded as occlusion cases, and segmentation method relies on splitting them, inter frame relations between vehicle blobs can be used for occlusion reasoning.

In this work, a blob association scheme based on simple position estimation is used. Each blob is regarded as an object, which may contain single or multiple vehicles. Blobs with multiple vehicles are considered as occlusion events, and number of vehicles in a blob is the feature used for intra frame occlusion reasoning. Number of vehicles in a blob is determined and updated in object association stage each frame. Therefore, this stage provides necessary information for intra frame occlusion detection. Blobs with multiple vehicles from current object list are considered as under occlusion, and Segmentation is applied if possible.

3.5.2 INTRA FRAME OCCLUSION DETECTION

In this section, an occlusion reasoning method based on only current frame information is proposed. Results of intra frame feature based occlusion analysis will be used in conjunction with intra frame detection results, which is a simply logical OR operation.

Model based approaches examined in Chapter 2 Section 3 are not selected for this framework, although they are successful with satisfactory results as seen in [11, 12]. Main reason behind this, as these model based approaches are highly dependent on scene geometry, and object properties. Also, they rely on a sequence of stages with moderated computational cost, to construct and analyze the constructed model. With

this complexity, it is not possible to provide a lightweight solution to object detection problem. Also, models are highly dependent, and must be updated for different camera placements.

Some part based models are also used for vehicle detection, but they are specialized for a very specific scene geometry. This restricts the environment and scene geometry for the solutions based on this approach. Therefore, they are ruled out for occlusion detection in this work.

Shape properties are widely used for occlusion reasoning in recent works. Ranging from simple area thresholding to detailed analysis of shape properties like contour curvature or shape convexity, these methods often offer occlusion detection with low computation cost compared to model based approaches. Detection is based on certain assumptions about geometrical properties of object shapes.

Area thresholding is simplest form of this methods. Although very easy to implement, this approach cannot provide desired results, as object sizes vary depending following factors:

- Different size of different class of vehicles
- Different size with varying distance from camera

In the proposed solution to intra frame occlusion detection, area thresholding is utilized as a candidate detection method. If area of a binary blob is larger than a scene dependent static threshold, blob is considered as a candidate for occlusion presence, and further analysis is done. Considering cost of other methods are significantly high compared to area calculation, and connected component analysis is already done on binary foreground mask, this step has practically negligible cost. Also, cost further operations are avoided for non-occluded objects.

Second feature used for occlusion detection is convexity of binary blob. Underlying assumption here is that vehicles without occlusion have a highly convex shape, and occlusion disrupts this convexity. First part of assumption mostly holds for vehicles in ITS surveillance applications. Unoccluded objects appear in convex shapes unless their shape is irregular, which is the case for some trailer trucks and boat towers. Sec-

ond part of this assumption also mostly holds, while there are rare occlusion cases with connected shape of objects are convex. Also, full occlusion cases cannot be detected, as shape of the front object is mostly undistorted. Considering these observations, convexity is a strong feature for occlusion with low computational cost, and selected as main feature for intra frame occlusion detection in the proposed solution.

There are several metrics proposed to measure how convex an object is. Most popular approach is using ratio between object area and its convex hull as a measure. As illustrated in Figure 3.8, when object is occluded, there is a significant area difference between object contour and convex hull. Considering this observation, area ratio defined in Equation 2.17 is used in this work.

To calculate convex hull of a vehicle, first contour is of binary connected component is extracted. Then algorithm to find convex hull of a 2D point set proposed by Sklansky [103] is used for convex hull extraction. Since algorithm has $O(n)$ complexity, and number of 2D points in a typical binary blob contour is small, cost of this operation is negligible. For implementation, C++ code provided by Open Source Computer Vision (OpenCV) [104] library is used.

A static threshold is defined to detect possible occlusion events. This value is calculated for each candidate filtered by area thresholding. It must be stated that, area threshold is set low, in order to detect maximum number of occlusion events. False positives detected here do not effect accuracy, as they are eliminated in Segmentation part.

3.6 SEGMENTATION

3.6.1 EVALUATION OF EXISTING METHODS

Segmentation of occluded vehicles based on convex hull analysis is discussed in literature. Works of Zhang et al [17] and Heidari et al [13],are examined in detail, as they exploit difference between convex hull and object contour for segmentation.

Heidari et al [13] proposed a method that simply cuts an occluded blob into two parts,

with a dividing line defined by dividing points. To select two points, a set of points is constructed from deepest point in each convexity defect.

While this method can solve occlusion cases with two vehicles, blobs formed by occlusion of three or more objects cannot be handled. Also, convexity defects may appear irregular, depending on the vehicle shape and accuracy of binary mask. Selecting closest two of deepest point set may cause dividing lines which does not divide actual vehicles.

Zhang et al [17] uses an iterative method to find cutting points that define dividing line. Instead of finding two points, they first find point on object contour with maximum distance to convex hull. Then for each angle in 360 degree range, they draw a line to segment blob. After each division, they recompute convex distance given in Equation 2.23. This approach has a high computational cost as for each iteration, object contours, convex hulls, area ratio and IDR must be recalculated, for each blob. Also blobs formed by occlusion of three or more objects cannot be handled.

3.6.2 PROPOSED METHOD OVERVIEW

Proposed method is inspired by methods mentioned in previous section, and can be considered as an extension to the methods proposed by Heidari et al [13] and Zhang et al [43]. Thess methods are used as base because of the low execution speed they offer. As the proposed solution aims to provide a lightweight vehicle detection tool, keeping execution time low while increasing overall performance is key. Major advantages of the proposed method are robustness to shape irregularities, and ability to handle occlusions with three or more vehicles.

An overview of the proposed segmentation stage is shown in Figure 3.9, and details are explained in following sections.

3.6.3 CONVEXITY DEFECTS ANALYSIS

In this part, convexity defects for occluded blobs are extracted and analyzed. Convexity defects are connected pixels between convex hull and object contour, which

are illustrated in Figure 3.10.

We are interested in depth of each pixel on object contour, which is the orthogonal distance to the convex hull of the blob. In base method, only point with maximum depth is analyzed, restricting candidate points for each defect to a single point.

In the proposed work, persistent maxima of the depth along the contour is calculated. For this purpose, Persistence1D algorithm, which calculates all extrema and their persistence in pairs. A persistence threshold, τ_p is which is set to 0.4 in this work, is used to eliminate noise and maxima points with low persistence. This way, for each convexity defect, more than one maxima points can be defined. These maxima points are actually candidate points, and segmentation will be completed by forming a cut between matched candidate points.

It must be noted that, each convexity defect has its own point set. This is important, as the method proposed in this work matches points from different convexity defects.

For each blob, which is subject to occlusion detection analysis, convex hull and contour is already extracted. Depth is calculated as in Equation 3.5

$$D_{Np} = \frac{|\,(y_2 - y_1)\,x_p - (x_2 - x_1)\,y_p + x_2 y_1 + y_2 x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}} \tag{3.5}$$

Where D_{Np} is calculated depth, N is index of the depth in blob, p is position along the object contour, (x_1, y_1) and (x_2, y_2) are starting and ending points of the defects on convex hull. Start and end points of the defects are marked with red in Figure 3.10.

3.6.4 CUT POINT SELECTION

After candidate points are generated, pairs of matched points must be formed. A matching procedure is done with following steps:

1. A N by N score matrix is formed, where N is number of candidate points in a single blob.

2. For each candidate, a match score M_s is calculated, but only with candidates from other defects. Matrix in first step is filled.

3. To detect matches, a segmentation score threshold τ_s is defined. An array of matching pairs are constructed, and pairs with score higher than τ_s are stored.

In base method, match score is based only on distance between two candidates. Although a strong feature for simple occlusion cases where two convex shapes overlap, this metric is not sufficient for complex cases. There may be more than two objects, and binary masks may include misclassified regions which may result with many convexity defects, as illustrated in Figure 3.11.

Also, in most cases, ideal match points are from two convexity defects, which tend to extend towards each other, from the convex hull. This means extending direction of a defect can also be used as a feature in matching process. A case, in which this behavior can be seen is shown in Figure 3.12. Figure illustrates a 3 vehicle occlusion scenario, with candidate points are marked with A,B,C and D. If only distance between two points are used as match criteria, then resulting matches would be A, B and C, D, while correct match set isA, C and B, D. To prevent this, another metric which measures the similarity of the extension directions must be introduced.

To calculate match score M_s, between candidate points, first two metrics are defined. Two metrics, M_d and M_o are defined and calculated for this purpose.

M_d is based on the distance between two candidate points, and indicates how much distance cutting line covers within the segmented blob. Instead of using 2D Euclidean distance directly, M_d is calculated as in Equation 3.6.

$$M_d = exp\left(M_{SC}\left(-\left(dist - T_d\right) / \left(CW\left(D_1 + D_2\right)\right)\right)\right) \tag{3.6}$$

Where M_{SC} is smoothing coefficient, $dist$ is 2D distance between candidate points, T_d is minimum distance threshold, CW is the weight coefficient, D_1 and D_2 are depths of candidate points. T_d is introduced to boost effect of distance in decision, when two points are very close. Sum of D_1 and D_2 in denominator is added to have adaptability to changing blob size. For larger blobs, depth values will be larger. Thus

same metric value will be obtained, even there is a larger distance between candidate points. Exponentiation is used to increase decay of the metric with increasing distance.

M_o is defined to measure how aligned two convexity defect areas are with each other. First an angle, θ_n is defined for each point, which is the deviation angle between these two lines:

1. Line between two candidate points
2. Line between current candidate point, and closest point to candidate on convex hull

These two lines intersect at the candidate point, and deviation angle between them is calculated for both candidate points. θ_1 is calculated as in Equation 3.7.

$$\theta_n = \arctan(|\left(S_p - S_n\right) / \left(1 + \left(S_p S_n\right)\right)|); \tag{3.7}$$

Where S_p is slope of the first line, and S_n is the slope of the second line.

Then M_o is calculated as in Equation 3.8:

$$M_o = \left(\left(1 - \cos^2\left(\theta_1\right)\right)\left(1 - \cos^2\left(\theta_2\right)\right)\right) \tag{3.8}$$

Where θ_1 and θ_2 are deviation angles for two candidate points evaluated. As two convexity defects become oriented with low deviation angles, M_o becomes close to zero. Note that M_d get close to 1 when distance between two points decrease. Values close to the 1 in both metrics indicate that selected candidate point set is good match.

After these two metrics are calculated, they are combined into M_s, which is actually segmentation score. For each possible candidate match, this value is evaluated as in Equation 3.9, and thresholded with τ_s to decide if they are matched. Formulation of M_s is done to boost effect of extreme values for both metric within range of $[0, 1]$. This means, extremely close candidate point pairs, and pairs with well aligned convexity defects with reasonable distance yields a higher M_s.

$$M_s = M_d M_o^2 + (1 - M_d M_o)\, M_d \tag{3.9}$$

3.6.5 CUT PATH GENERATION

After matching point set is generated, blobs are segmented to smaller ones, using cuts between matched points. In base method, a dividing line is formed between two points, which is basically removal of pixels along this line, from the binary map. This approach have two major drawbacks. First one is, dividing line does not overlap with the actual border between vehicles. This may result with both segmented objects have a small part of each others hull, after segmentation. Second drawback is, if a pixels are removed from mask, then a connected component analysis must be performed again, for object detection. Although, cost of this operation is not very significant, it can be avoided.

To solve first problem, which actually degrades boundary adherence of object detection, a edge based solution is proposed. Instead of using a dividing line, a path between two points which along the edges of the image is used. As borders between the objects can be exposed with edge detection methods, a dividing path through these edges may result with better boundary adherence.

To detect edges, Sobel Operator [105] is chosen, as 3 by 3 separable filter is low cost with satisfactory edge detection.

After edges are detected, a path along these edges must be calculated. For this purpose, a minimum cost path finding algorithm is needed. Dijkstra's algorithm is a popular approach [106], and widely used in many similar problems. Considering the fact that the path we are trying to find is actually close to the dividing line with some deviation, path finding operation can be optimized by "guiding" the search along this direction. Therefore, a modification of Dijkstra's algorithm, which is named A star (A*) [107] Search algorithm is selected. A* introduces a heuristic based guidance to the search process. This way a better performance compared to exhaustive methods are obtained, and algorithm become popular in real time strategy games, used for unit path finding [108].

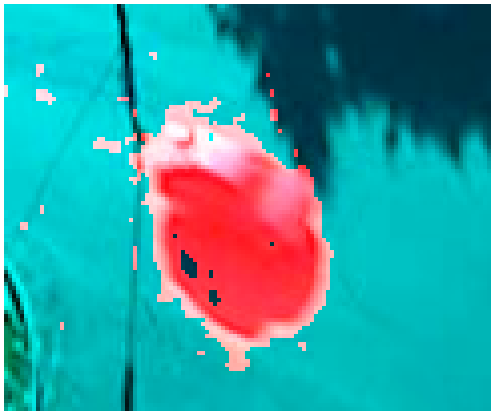
To run A* algorithm, first a graph with nodes and connections must be formed. For our case each pixel is considered as a node, and has connections to the 8 pixel around them. To guide path through pixels with edge presence, node weights are adjusted. First, edge strengths are normalized to [255-0] range, with 0 being the strongest edge value. Then these values are set as node weights. Heuristic to guide search is defined as 2D Euclidean distance between two points. This way, search is guided to the direction of the line connecting two points.

Figure 3.13 shows results of dividing line and dividing path methods for segmentation. It is clear that, edge based path follows border between vehicles closely, thus achieving a higher boundary adherence.

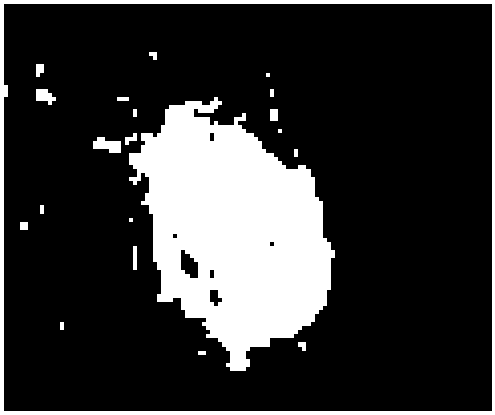
For the cases, in which A* search cannot find a path, dividing line approach is utilized. An overview of the path extraction stage is shown in Figure 3.14.

To solve second problem of the base method mentioned before, all operations on objects are conducted on contours. First, after segmentation path is found, contour of the segmented blob is cut into two parts from cut points. Then each part is concatenated with segmentation path, yielding two separate contours. Segmented object is removed from the current list of objects, two new objects are formed using contours extracted and object data for the next frame is updated.

(a) Vehilce with hard cast shadow

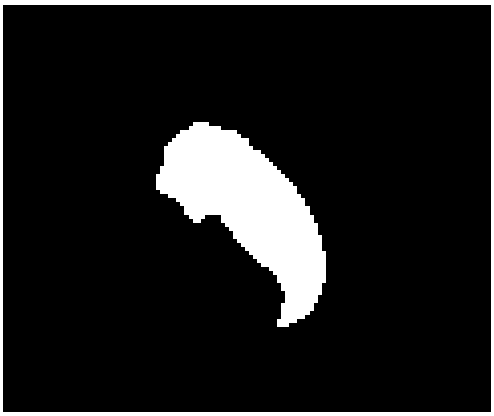


(b) SAGMM mask overlay



(c) SAGMM mask

(d) SuBSENSE mask overlay



(e) SuBSENSE mask

Figure 3.4: Comparison of performance under hard shadow effect.

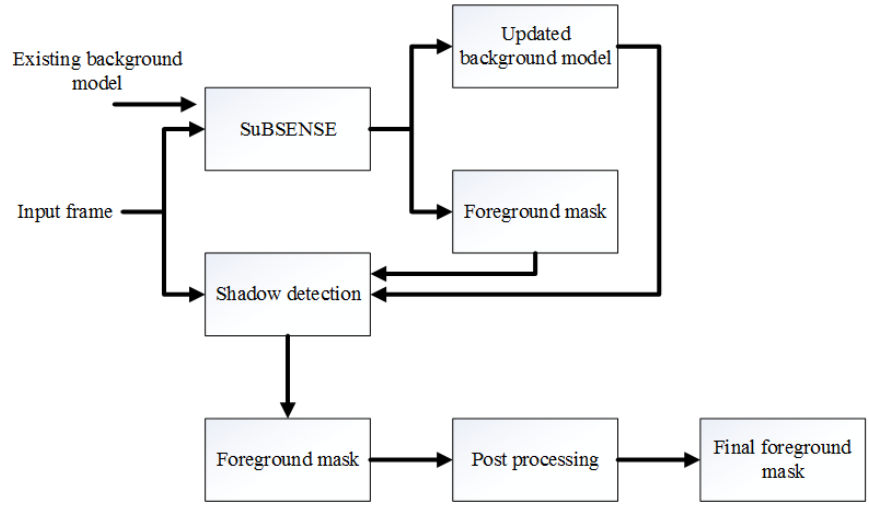


Figure 3.5: Overview of foreground mask generation

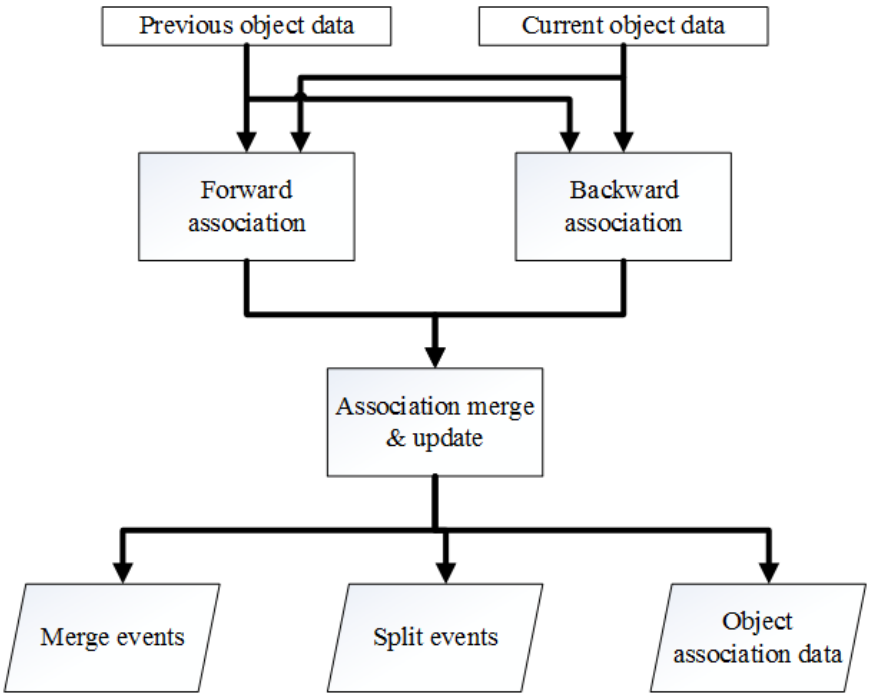


Figure 3.6: Overview of object association

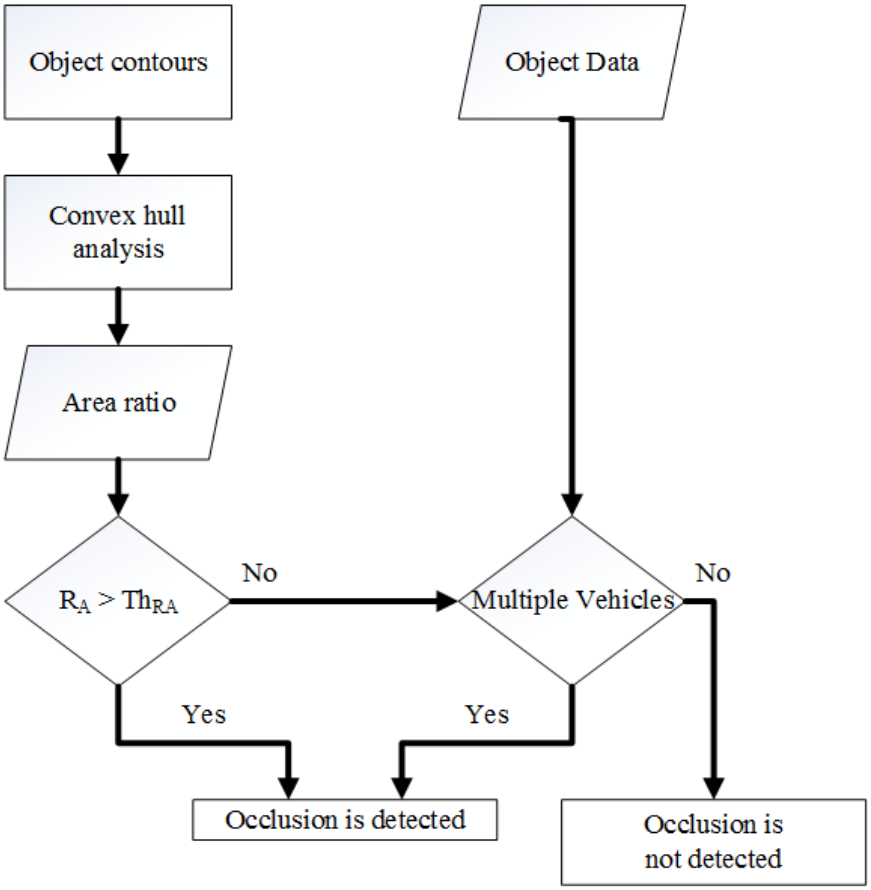


Figure 3.7: Overview of the object occlusion reasoning method proposed

(a) Occluded Vehicles

(b) Mask and Convex Hull

(c) Unoccluded Vehicle

(d) Mask and Convex Hull

Figure 3.8: Convex hulls, and convexity defects for occluded and unoccluded vehicles. Defects are marked with red.

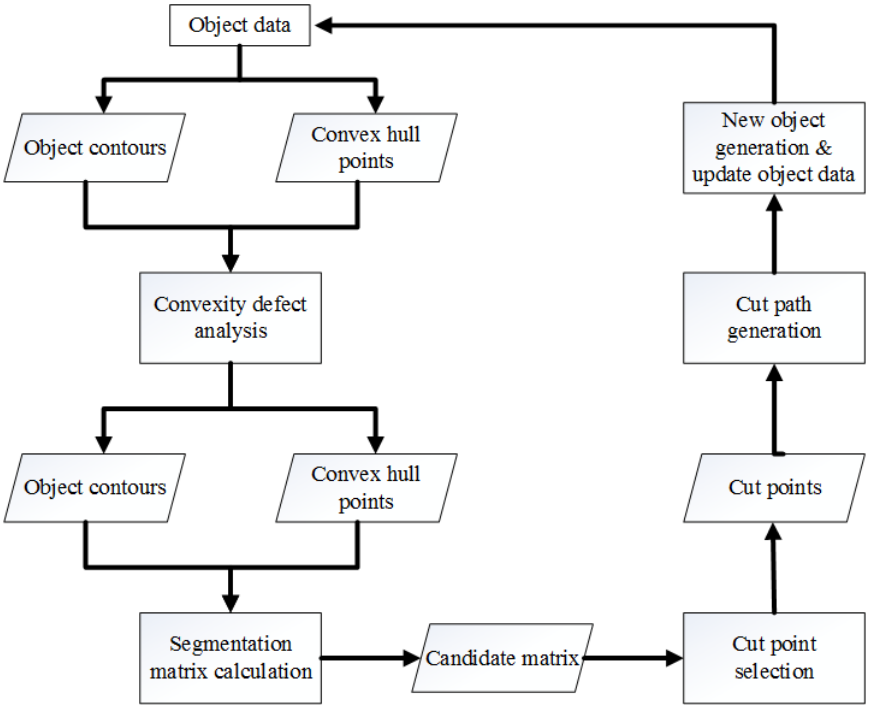


Figure 3.9: Overview of segmentation stage

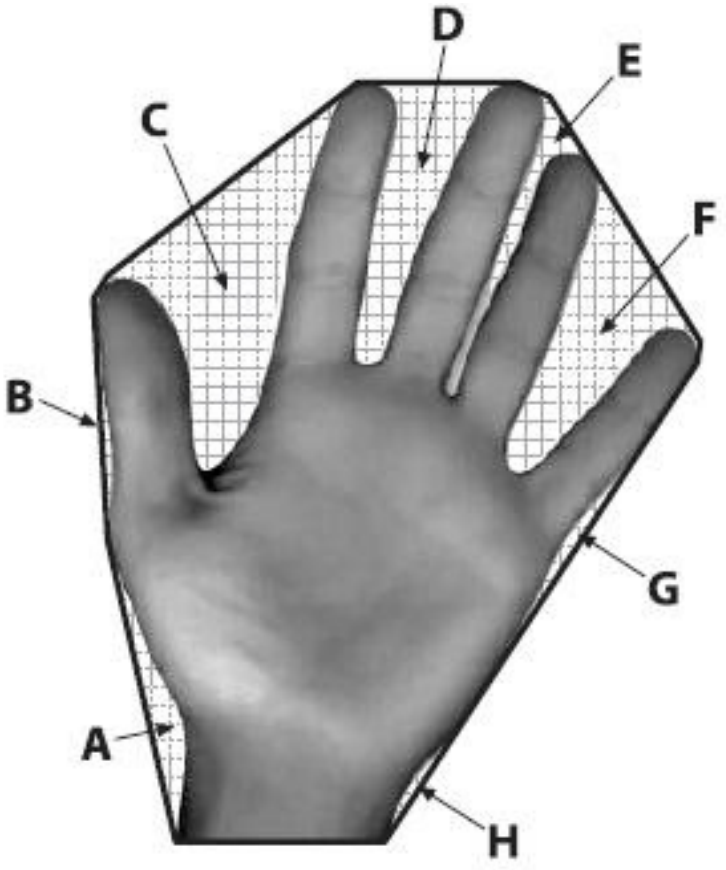


Figure 3.10: Object(hand), convex hull and convexity defects. Convexity defects are marked with letters. retrieved from [19]

(a) Original Image (b) Binary Mask

(c) Convex Hull and Defects

Figure 3.11: Multiple convexity defects with three vehicle occlusion case

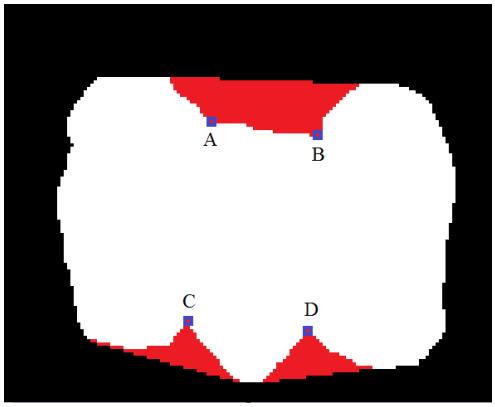


Figure 3.12: Candidate points in a three vehicle occlusion case

(a) Dividing line (b) Edge based dividing path

Figure 3.13: Dividing line and dividing path approaches

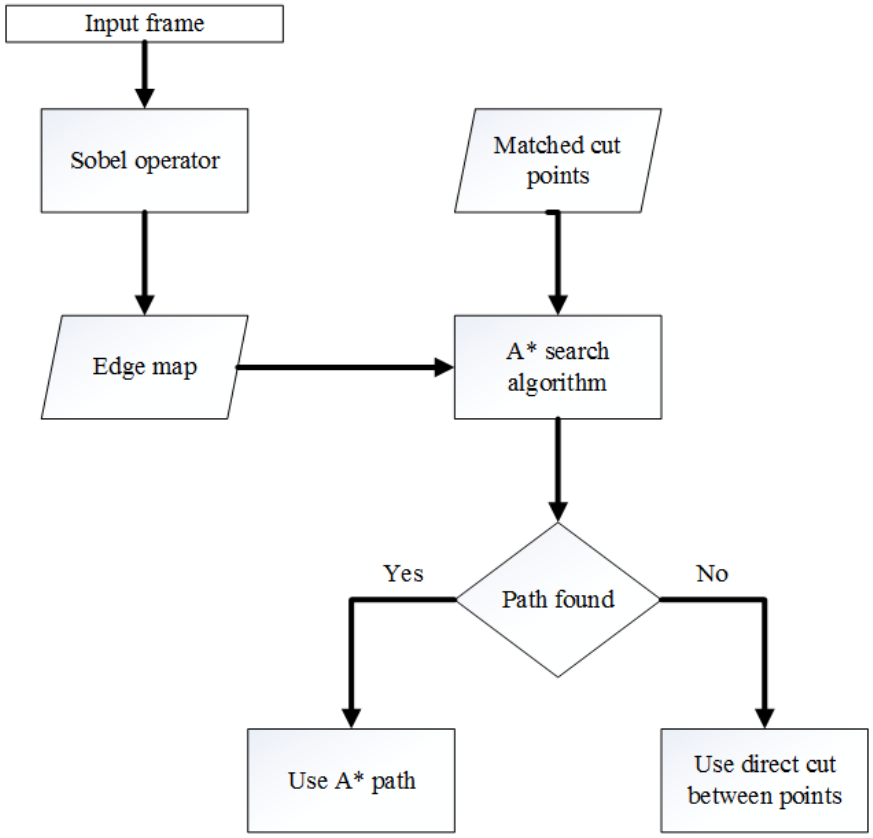


Figure 3.14: Overview of object association

CHAPTER 4

EXPERIMENTAL WORK

4.1 EXPERIMENTAL WORK OVERVIEW

Implementation of the proposed solution is first done on MATLAB in subsystem level, to obtain primitive results. Libraries from other sources in C++/C languages are imported to this environment.

After design and development phase is complete, solution is implemented in C++ language using Visual Studio 2012 IDE. Following open source libraries are used in this implementation:

- OpenCV 2.4.10 [109]
- OpenGL 4.2.0 [110]
- freeGLUT 2.8.1 [111]

OpenGL and freeGLUT libraries are used only for visualization purposes, and does not take part in functional part of the implementation.

For implementation in C++, following source codes are used after adaptations to the VS2012 IDE with updates and modifications:

- GMM implementation from BGSLibrary 1.9.2 is updated to the SAGMM [51].
- Shadow detection source codes provided by A. Sanin [8] are used as base for shadow detection functions.

- LBSP implementation provided by P. St-Charles is used after minor updates, and provide basis for LBSP feature descriptors used in SuBSENSE and LOBSTER.
- SuBSENSE implementation provided by P. St-Charles is used as a basis. Updates to increase speed, and remove memory leaks are done.
- LOBSTER implementation provided by P. St-Charles is used as a basis. Updates to increase speed, and remove memory leaks are done.
- OpenCV function "convexityDefects(...)" is used as a base for functions calculating candidate points.
- A* implementation of Justin Heyes-Jones [112] is used as a base for A* search used in cut path generation. Graph structure, node structure, graph construction and traversal parts of the code is updated to work with increased efficiency with 2D edge map.

Tests and time measurements are done on a PC with Intel I7-4072MQ @ 2.20 GHz CPU, and a GeForce 820M GPU. Two different software configurations are defined in order to determine and compare optimization potential of different methods.

- Software Configuration 1 (SW1) :
  - Implemented compatible to 32-Bit Architecture (x86 Instruction set)
  - Single thread, single core
  - No OpenMP support
  - No Single Instruction Multiple Data (SIMD)support
  - No GPU support
- Software Configuration 2 (SW2) :
  - Implemented compatible to 64-Bit Architecture (x86 Instruction set)
  - Multi core (4) multi threaded operation is enabled
  - OpenMP support enabled, OpenMP directives are used in parts of the code, which are suitable for parallelization.

  - No SIMD (Single Instruction Multiple Data) support
  - No GPU support

Quantitative and qualitative results with computation times for each sub stage of the solution are evaluated in following sections. For each sub stage, performance is compared with existing alternatives.

Also, a detailed analysis for computational cost and execution speed is done. For each function block, following metrics are calculated:

- Average of Execution Time (AET): Mean of the execution time, calculated by dividing total time spent on the function by the number of function calls. Indicates expected time function requires to complete.
- Maximum of Execution Time (MET): Maximum value of the execution time. This statistic is critical for real time applications. To guarantee a real time operation, this value must be considered before using a function.
- Variance of Execution Time (VET): Variance calculated from all individual execution times of the function. This metric indicates dependency of execution time to input data.

Microsoft Visual Studio 2012 IDE provides a wide range of tools for benchmark and profiling purposes, which are utilized for calculation of above metrics.

Different datasets are used for evaluation of different stages of the proposed solution. List with video details are given in Table 4.1.

For video sequences Seq 1-4, number of vehicles and occlusion events are counted manually for each frame, to construct ground truth. Also, vehicle pixels are tagged with different values for different vehicles, to construct a ground truth for segmentation accuracy. This marking is done once per 25 frame. Duration and frame rate of sequences are as following:

- Seq1 : 10 FPS, 2450 frame, 4:05 min

Table 4.1: Video Sequences used for testing phase

	Description	Source	Supplementary information	Stage Used
Seq 1	Top view of an intersection, fish-eye lens	Recorded	Ground Truth for vehicle count, and some vehicle silhouettes	Occlusion Detection and Segmentation
Seq 2	Top view of an intersection, fish-eye lens	Recorded	Ground Truth for vehicle count, and some vehicle silhouettes	Occlusion Detection and Segmentation
Seq 3	Top view of an intersection, fish-eye lens	Recorded	Ground Truth for vehicle count, and some vehicle silhouettes	Occlusion Detection and Segmentation
Seq 4	Top view of an intersection, fish-eye lens	Recorded	Ground Truth for vehicle count, and some vehicle silhouettes	Occlusion Detection and Segmentation
highway	Top view of a highway	CDNET	Ground Truth for foreground, background and shadow pixels	Background Subtraction
traffic	Top view of a highway with camera jitter	CDNET	Ground Truth for foreground, background and shadow pixels	Background Subtraction
boulevard	Top view of a road with camera jitter	CDNET	Ground Truth for foreground, background and shadow pixels	Background Subtraction
streetlight	Top view of a highway	CDNET	Ground Truth for foreground, background and shadow pixels	Background Subtraction
tram cross-road	Top view of an intersection with tram	CDNET	Ground Truth for foreground, background and shadow pixels	Background Subtraction
turnpike	Top view of a highway	CDNET	Ground Truth for foreground, background and shadow pixels	Background Subtraction
tunnelexit	Top view of a tunnel exit	CDNET	Ground Truth for foreground, background and shadow pixels	Background Subtraction
intermittent pan	Top view of a road, PTZ camera	CDNET	Ground Truth for foreground, background and shadow pixels	Background Subtraction

- Seq2 : 10 FPS, 2250 frame, 3:45 min
- Seq3 : 10 FPS, 2400 frame, 4:00 min
- Seq4 : 10 FPS, 2400 frame, 4:00 min

4.2 BACKGROUND SUBTRACTION

Background subtraction is the first step in the object detection procedure proposed in this work. A binary mask of foreground pixels is computed using subtraction of background model from the current frame. As occlusion detection and segmentation stages utilize vehicle geometry, precision of this stage is crucial.

For moving object detection, two methods are selected from wide range of available methods; [51], and [4]. Implementation is done based on open source C++ codes, and OpenCV library.

4.2.1 PERFORMANCE MEASURES

Benchmarking tools provided in CDNET [1] are extensively used in recent works on change detection, and most state-of-art algorithms are evaluated by same metrics and methodologies provided in their benchmark guide. Therefore, quantitative performance analysis is of background subtraction methods are done using same metrics. Following metrics are calculated for each algorithm, in different videos.

- TP : True Positive
- FP : False Positive
- FN : False Negative
- TN : True Negative
- Re (Recall) : TP / (TP + FN)
- Sp (Specificity) : TN / (TN + FP)

- FPR (False Positive Rate) : FP / (FP + TN)
- FNR (False Negative Rate) : FN / (TP + FN)
- PWC (Percentage of Wrong Classifications) : 100 * (FN + FP) / (TP + FN + FP + TN)
- F-Measure : (2 * Precision * Recall) / (Precision + Recall)
- Precision : TP / (TP + FP)

MATLAB script provided on CDNET [1] is used to calculate results obtained in this work.

For each method, a qualitative analysis is also performed. Metrics defined above describe overall pixel based performance in this stage, independent of effect on performance in later stages. As occlusion detection and segmentation method is dependent on binary mask geometry, two algorithms with similar metrics can affect further stages differently, depending on the distribution of the error.

4.2.2 DATASET

Evaluation of background subtraction algorithms requires video datasets with different properties. For this work, moving object detection is done on intersection environment. For accurate evaluation, videos recorded in similar scenarios are preferred.

Another important point in dataset selection is existence of ground truth, as metrics described in previous section are calculated based on them. CDNET datasets of 2012 and 2014 offer a wide range of scenarios with ground truth masks. Therefore, this dataset used for subsystem level quantitative analysis.

For qualitative analysis, selected videos from CDNET dataset are used, which are listed in Table 4.1.

Table 4.2: SuBSENSE results from CDNET [1]

	Re	Sp	FPR	FNR	PWC	F-Measure	Precision
SuBSENSE 2012 dataset	0.8281	0.9938	0.0062	0.1719	1.5447	0.8260	0.8576
SuBSENSE 2014 dataset	0.8124	0.9904	0.0096	0.1876	1.6780	0.7408	0.7509

Table 4.3: Results obtained from selected videos of 2012 and 2014 datasets of CDNET

	Re	Sp	FPR	FNR	PWC	F-Measure	Precision
SuBSENSE	0.8364	0.9934	0.0066	0.1636	1.7425	0.8692	0.9046
SAGMM	0.7727	0.9853	0.0147	0.2273	2.9377	0.7845	0.7966

4.2.3 EXISTING EVALUATIONS

A detailed evaluation of SuBSENSE with results are provided on CDNET. Average rankings of SuBSENSE among evaluated methods based on given metrics are, 4.14 for 2012 dataset and 5.29 for 2014 dataset. Detailed results are given in Table 4.2, as reported on CDNET website.

SAGMM is not evaluated on CDNET, and an evaluation with similar detail does not exist. Authors [51] provide a detailed comparison with ZHGMM [49], but calculated metrics and video dataset are different. Therefore, metrics given in previous section have to be calculated for SAGMM, with same dataset used for SuBSENSE.

4.2.4 RESULTS

For quantitative evaluation, videos with moving vehicles from the datasets provided in CDNET are used. Using MATLAB script provided in website, metrics are calculated form output of two algorithms, Table 4.3. Thresholds are inherited from original implementation for SuBSENSE, and from parameter set of base method in CDNET for SAGMM [49].

Quantitative comparison of two methods indicate that SuBSENSE is more successful, with better results for all metrics. Obtained results for SuBSENSE have slight

difference from existing evaluation in CDNET, as whole dataset is not used.

Two algorithms are also run on different intersection videos for qualitative evaluation.

SuBSENSE, has a considerably better performance compared to SAGMM in all cases. Apart from quantitative results, SuBSENSE is better at object localization. Exploitation of spatial information by utilization of LBSP descriptors increase spatial consistency of the binary mask generated. In contrast, SAGMM does not use spatial information between local pixels, and resulting performance difference can be seen in Figure 4.1 and Figure 4.2. When two masks are compared, it is evident that SAGMM requires certain post-processing steps for eliminating single pixel misclassification, and achieving a local spatial consistency between pixels. Also, SuBSENSE has shown superior performance in both hard and soft shadow areas. This is expected as LBSP is mostly unaffected by intensity change caused by moving cast shadows, while intensity is only feature for pixel classification in SAGMM.

SuBSENSE has also another major advantage, which is adaptation to dynamic scenes. For weather conditions like rain or snow, scene variance greatly increases, therefore variance of the estimated PDF of the background model also increases. For SAGMM, this case become problematic, as after adaptation, this often leads to misclassification of foreground pixels as background, due to increased variance of Gaussian modes. But for SuBSENSE, different adaptive thresholds are used if scene variance is high, which enables accurate detection in regions with high variance.

Another important performance difference is in accuracy of object shape data. A commonly encountered problem which degrades this accuracy, is misclassification of wind shields of vehicles as background. Due to close intensity values of wind shield and road surface, wind shields are often classified as background. This becomes problematic, as, misclassification of this area can lead to false positives in occlusion detection. This situation can be seen in Figure 4.1. As seen, binary mask generated using SuBSENSE is more accurate, and misclassification in SAGMM output has a potential negative effect on whole system. Due to missing area, there is a significant difference between object and its convex hull, which degrades performance of occlusion detection and segmentation.

Table 4.4: Execution Time statistics for two algorithms

	SW1			SW2		
	AET	MET	VET	AET	MET	VET
SuBSENSE	145 ms	221 ms	3436 ms^2	92 ms	120 ms	264 ms^2
SAGMM	76 ms	84 ms	22 ms^2	29 ms	35 ms	2.281 ms^2

While having a superior accuracy, SuBSENSE has a significantly higher execution time. This is expected, as computation of LBSP descriptors have non coalesced memory access with low cache utilization. Pixel-wise operations access values of neighboring pixels, while pixel-wise operations are independent in SAGMM.

Measurements of execution times for 512x512 RGB videos are given in Table 4.4. As seen in results, SAGMM has a higher a speed-up of 2.62 between SW1 and SW2, while SuBSENSE has only 1.576. This indicates a higher optimization potential, which means optimization via parallelization yield better results in SAGMM compared to SuBSENSE.

Both methods have significant variance in execution time, dependent on the object motion in scene. More time is spent in frames with more foreground pixels. Maximum values are observed when scene is crowded with many moving vehicles, for both algorithms.

4.3 SHADOW DETECTION

For shadow detection, texture based methods are selected considering their proven performance [8]. C++ implementation provided by A.Sanin [113] is used as a base. Large Region Texture (LRTex) and Small Region Texture (SRTex) Methods are tested.

For quantitative performance analysis, results reported in [8] are used. Qualitative analysis is done by observing and discussing possible negative effects of misdetections. Algorithms are also tested in two software configurations, and execution time statistics are analyses.

4.3.1 PERFORMANCE MEASURES

Two metrics are used by Sanin et al [8], to evaluate performance of shadow detection algorithms. First of these metrics is Shadow Detection Rate, which is same as Recall(Re) defined in previous section. Second is Shadow Discrimination Rate, which is same as Specificity(Se). Metrics are calculated as in Section 4.2.1, but some definitions change:

- TP : True Positive : Shadow pixel classified as shadow
- FP : False Positive : Non-shadow pixel classified as shadow
- FN : False Negative : Shadow pixel classified as non-shadow
- TN : True Negative : Non-Shadow pixel classified as non-shadow

4.3.2 EXISTING EVALUATIONS

In [8], several shadow detection algorithms are evaluated using a 7 video dataset. Quantitative results reported are shown in Figure 4.3. Also, effects of selected shadow detection algorithms on the performance of a tracking application is analyses. Results show that, LRTex method is most successful with most improvement in track accuracy and precision while SRTex method comes second.

Execution times are also analyses in same work. On a 32-bit Intel CPU with 2.6 GHz clock speed, SRTex method had an average of 211.93 ms execution time. This time is reduced by using simpler version of texture filter, to 70.30 ms. LRTex has a better execution time in same hardware, with an average of 21.78 ms.

4.3.3 RESULTS

A quantitative analysis is done to detect possible effects of shadow detection stage on occlusion detection and segmentation. As both methods are texture based, performance decreases when shadow strength increase. This is observed in both algorithms,

Table 4.5: Execution time analysis for LRtex and SRTex methods

	SW1			SW2		
	AET	MET	VET	AET	MET	VET
LRTex	106 ms	185 ms	1537 ms^2	72 ms	123 ms	532 ms^2
SRTex	156 ms	247 ms	6934 ms^2	127 ms	196 ms	4081 ms^2

while LRTex is slightly better at classifying hard shadows. However there is a noticeable performance difference in shadow discrimination. When SRTex method is used, more vehicle parts are classified as shadows. This leads to the similar problem addressed in previous sections, which is inaccurate object geometry in binary mask. LRTex method is more robust to these effects, as features are extracted from largest possible regions.

An execution time analysis is also done for both algorithms. Results can be seen in Table 4.5. LRTex method has a significant advantage in terms of speed. Also, both algorithms have high variance in execution time, as their operation is fully dependent on the number of foreground pixels in given foreground mask.

It must be also noted that, although shadow detection is used as a step to increase accuracy of foreground mask, execution time is comparable to the background extraction. Reason behind this is the cost of calculation of directional gradients where costly trigonometric functions are used.

4.4 OCCLUSION DETECTION

Occlusion detection stage is tested in two configurations. In first configuration, a simple area ratio based thresholding approach is utilized. This part of the proposed work uses same area ratio definition with base methods [13, 43], given in Equation 2.17. Second stage is inter frame occlusion detection, where association data is used for reasoning. It is expected that, true positive rate would increase with exploitation of inter frame information.

First performance measure for occlusion detection is detection rate of multi vehicle occlusions. Unlike base methods [13, 43], multi vehicle occlusions with more than

two occluding vehicles can be handled in the proposed work. Second is measure is the ratio of false positive occlusion detections. Main reason behind presence of false positives, is erroneous binary mask, which received from moving object detection stage.

4.4.1 EXISTING EVALUATIONS

Existing methods in literature often used ratio of detected true positive occlusions to total number of occlusions, to evaluate success of occlusion detection. In [16], detection rates are given and Receiver Operating Characteristic (ROC) curves for partial occlusion detection is calculated for different methods. In results of [13, 43], only detection rate is used for measuring performance. Results are heavily dependent on static threshold enforced on area ratio, therefore ROC curve analysis is required to measure this dependence.

4.4.2 DATASET

As dataset in the base method [13, 43] is not provided, a new dataset to evaluate both methods is formed. For 4 different video sequences, following steps are done to construct a dataset:

- Definition of Region of Interest (ROI)
- Determination of occlusion events in sequence inside ROI (N)
- Determination of occluded vehicles in a group inside ROI (TOV)
- Determination of vehicle count in each frame inside ROI (TV)

4.4.3 RESULTS

For each video sequence, following statistics are calculated based on comparison of detected events and ground truth:

Table 4.6: Occlusion Detection results compared with Ground Truth for both methods

	Base Method [13]				Proposed Method			
	N	TP	FP	FN	N	TP	FP	FN
Seq1	234	196	20	38	234	214	20	20
Seq2	171	133	18	38	171	150	18	21
Seq3	226	190	16	36	226	219	16	7
Seq4	125	101	9	24	125	107	9	18

- TP : True Positive
- FP : False Positive
- FN : False Negative
- TN : True Negative
- N : Total number of occlusions in video sequence. Number is obtained after manual counting of independent occlusion events with at least three frame persistence.

Proposed method performed better in all sequences, yielding better Recall and Precision values. Results are shown in Table 4.6. Area ratio threshold R_A is set to 0.8.

In order to show improved accuracy, and reduced dependence to the static threshold R_A, a ROC curve is calculated for both methods. To obtain ROC curve, area threshold R_A is set to values between 0.0 to 1.0. Result can be seen in Figure 4.4.

Execution time for area ratio calculation is observed to be lower than 200 μs. Therefore, detailed analysis is not conducted for this part.

4.5 SEGMENTATION

Segmentation stage is the final part of the proposed solution. In this part, both base method [13] and the proposed method are implemented and tested. Work of Zhang et al [43] is not implemented, as iterative nature of their algorithm is not very efficient for real time applications.

Quantitative evaluation is mainly based on the number of vehicles after segmentation. Vehicle count after a correct segmentation should match with ground truth. Therefore, evaluation is done based on this comparison. However, as both base method and the proposed method do not utilize a multi frame memory, temporal consistency is low. Therefore, instead of checking successful segmentation each frame, a unique occlusion event is counted as segmented correctly, if segmentation result matches with ground truth at least 3 consecutive frames. Following metrics are defined for quantitative evaluation :

- N_o : Number of correctly detected occlusion events. Method is evaluated these cases.
- T_S : Number of successfully segmented events
- F_S : Number of events with failed segmentation

An accuracy measure is defined to evaluate selection of cut path. If vehicles are separated from incorrect points or path, resulting vehicle masks contain pixels from different. This is illustrated in Figure 4.5. Therefore, to measure success of segmentation accuracy, a ground truth is formed by marking pixels of each individual vehicle under occlusion in certain frames. This way, improvement achieved by using a cut path based on edge strength can be measured. To calculate segmentation accuracy, following metrics are defined:

- A_T : Calculated for each segmented vehicle. Equal to the number of pixels of the largest group with same mark in segmented object. For ideal case, this value is equal to the area of the vehicle in ground truth, which means perfect segmentation.
- A_N : Area of each segmented blob
- S_C : Segmentation score, calculated as $100 * A_T/A_N$

Dataset used in this part is the same dataset used in previous section. Seq 1-4 listed in Table 4.1, are used as a ground truth of vehicles is extracted for them. Results are illustrated for each video sequence, and for both methods in Table 4.7.

Table 4.7: Segmentation results for both methods. These results are obtained by segmenting TP occlusion events, using the proposed method for occlusion detection.

	Base Method [13]				Proposed Method			
	N_oN	TS	FS	Average S_C	N_o	TS	FS	Average S_C
Seq1	214	146	68	57	214	181	33	81
Seq2	150	82	78	54	150	135	15	79
Seq3	219	120	99	44	219	194	25	84
Seq4	107	66	41	48	107	92	15	78

Table 4.8: Execution time analysis for Segmentation stage

	SW1			SW2		
	AET	MET	VET	AET	MET	VET
Proposed	7795 μs	10240 μs	5.049 ms^2	3994 μs	5106 μs	0.968 ms^2
Base [13]	2795 μs	3407 μs	0.068 ms^2	1285 μs	1592 μs	0.0622 ms^2

4.5.1 RESULTS

Proposed method performs better at both in ability to solve occlusions, and segmentation accuracy. An exemplary result on test map is shown in Figure 4.5. In Figure 4.5 (c), object segmentation is done with base method. Figure 4.5 (f) shows cut path generated with the proposed method, achieving a greater accuracy. Furthermore, base method fails when more than two vehicles are in same occluded group. Proposed method handles these cases to some extent.

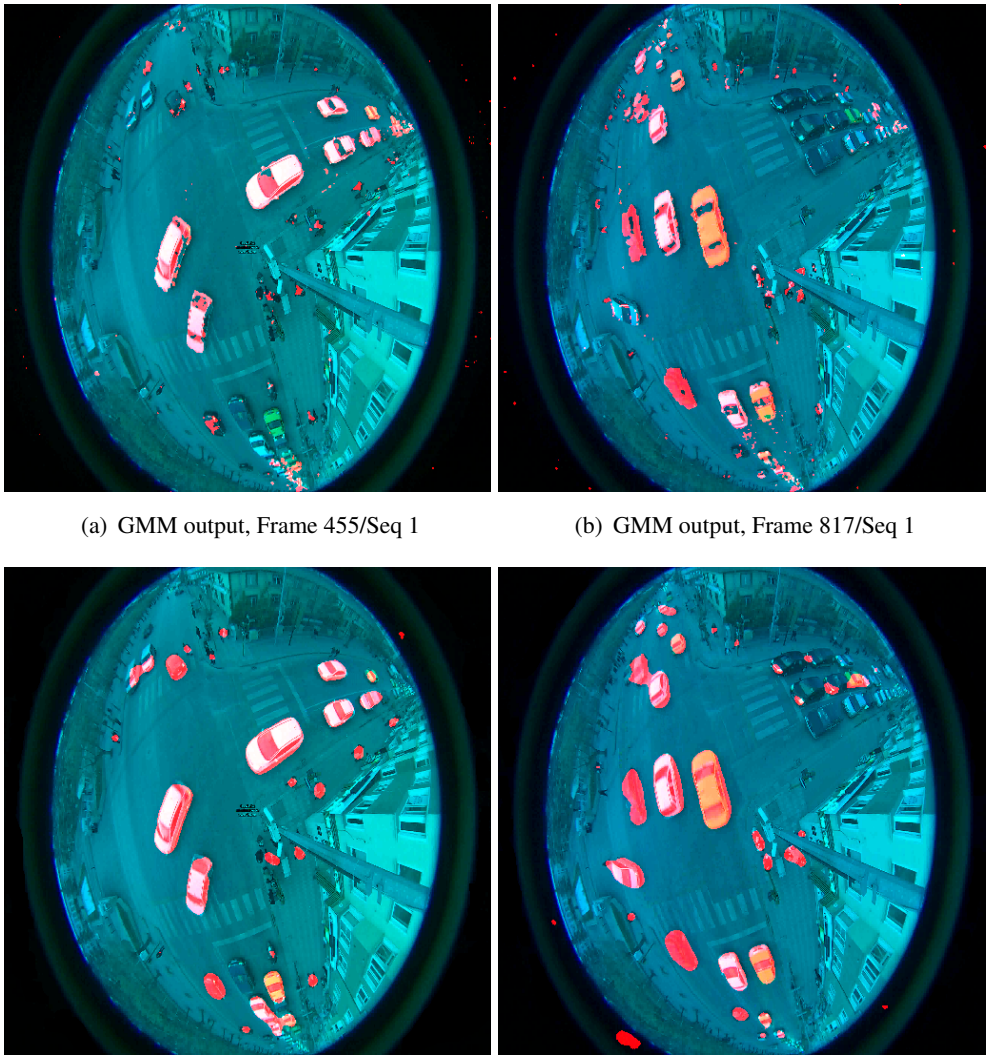
Detailed analysis of execution time is done on both methods. Although more accurate, the proposed method takes longer to segment vehicles. This is mainly caused by A* search algorithm, as shown in Table 4.9. Execution time measurements are shown on Table 4.8, for both methods and different software configurations. It must be noted that, although execution time increases in the proposed method, execution time of previous steps are significantly higher. Therefore, this part is far from being a bottleneck for whole system.

Results indicate that, segmentation part achieved a considerable improvement in accuracy, while maintaining a low execution time, for real time operation. Also, segmentation is not restricted to only groups of two vehicles, which is the case for base

Table 4.9: Detailed execution time analysis of stages of the the proposed method.

	Convexity & Defect Analysis	Candidate Point Generation	Cut Point Selection	Cut Path Generation	New Object Generation
Percentage of execution time spent (average)	% 19	% 9	% 28	% 38	% 6

method [13].



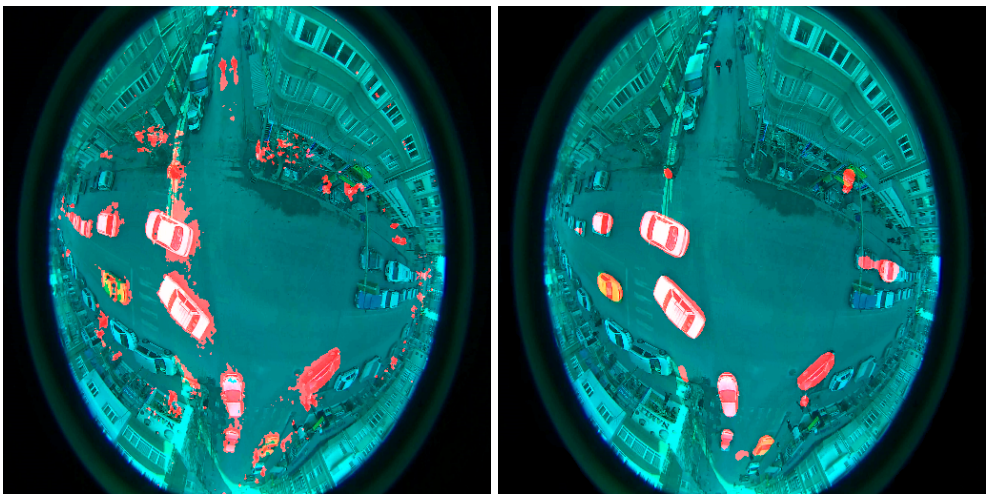
(a) GMM output, Frame 455/Seq 1

(b) GMM output, Frame 817/Seq 1

(c) SuBSENSE output, Frame 455/Seq 1

(d) SuBSENSE output, Frame 817/Seq 1

Figure 4.1: Binary masks generated by SAGMM and SuBSENSE. Foreground pixels are marked in red channel of the image. Notice that vehicle bodies are mostly intact in SuBSENSE masks, while there are holes and missing parts in SAGMM masks



(a) GMM output, Frame 455/Seq 2 (b) SuBSENSE output, Frame 817/Seq 2

Figure 4.2: Binary masks generated by SAGMM (a) and SuBSENSE (b). Foreground pixels are marked in red channel of the image. Notice that SuBSENSE classifies shadow and highlight pixels as background, while SAGMM fails to achieve this for large number of pixels

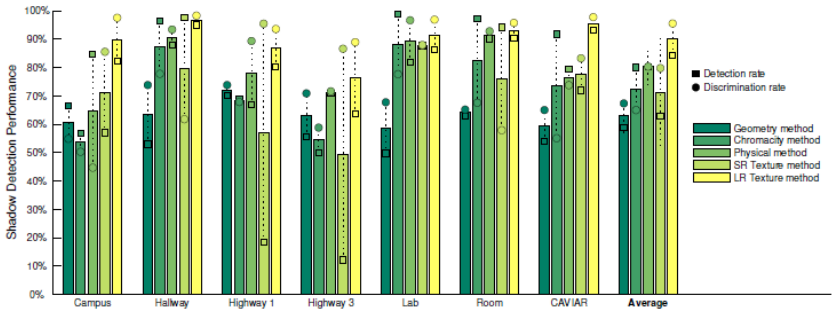


Figure 4.3: Comparison of different algorithms by Sanin et al [8]

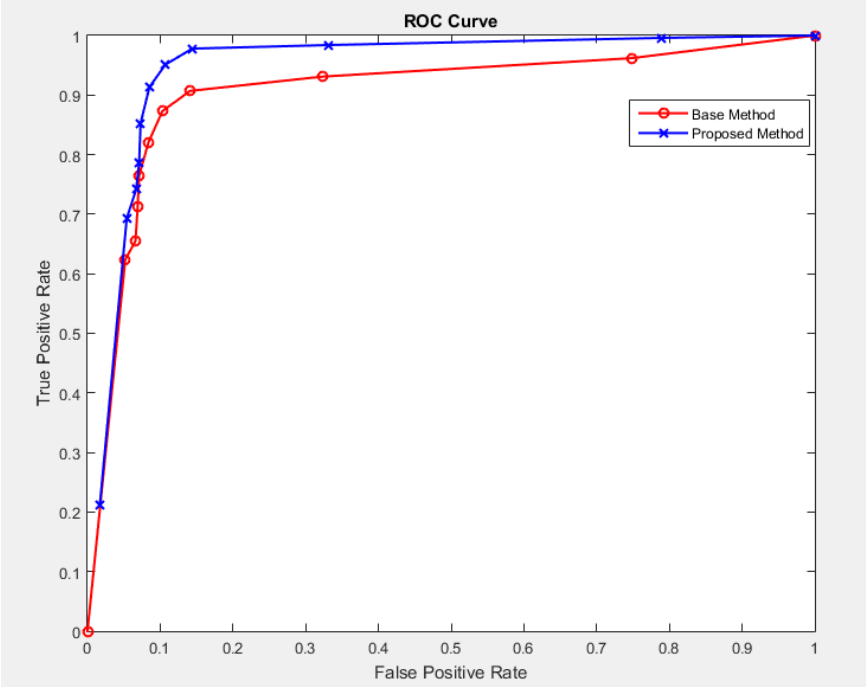
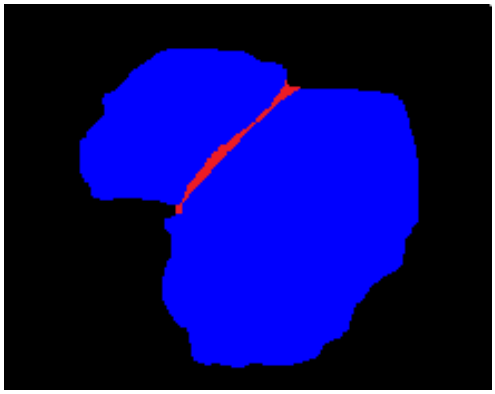


Figure 4.4: ROC curves for base method and proposed method

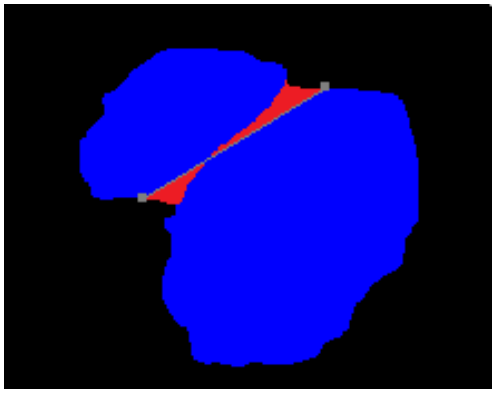
(a) Test Image

(b) Object mask of test image

(c) Cut points and path, base method [13]



(d) Incorrectly grouped pixels, base method [13]



(e) Incorrect cut points and path with markings, base method [13]

(f) Correct points and correct path, proposed method

Figure 4.5: Comparison of segmentation accuracy of base method and proposed method. Incorrectly grouped pixels are marked in red

CHAPTER 5

CONCLUSION

Vision based methods are widely utilized for ITS solutions, with increasing demand. While traffic management become more complex, performance requirements and range of tasks for vision based also increase dramatically. This work is motivated by this trend, and aims to provide an accurate vehicle detection framework, which operates robustly under presence of occlusion.

Results obtained from experimental work show that, method is successful in detection and segmentation of partial occlusion cases. Compared to available methods, object detection performance significantly increased, while maintaining a relatively low execution time.

In moving object detection stage, a recently developed background subtraction method is implemented. As expected, SuBSENSE algorithm achieves better results, compared to the popularly used SAGMM. Spatial consistency introduced by use of LBSP descriptors is a key improvement, as segmentation methods perform better, if shape properties are not distorted due to misclassified pixels. Moreover, illumination invariance of the texture based LBSP feature enabled method to work robustly under different illumination conditions.

Occlusion detection and segmentation part was proposed to close the gap in speed accuracy trade off. Existing methods were either too inaccurate, or very costly in terms of computational power. Result of this study is a light weight framework, with considerable accuracy and ability to handle multi vehicle occlusions.

Experimental results and observations indicate that, the proposed framework handles

object detection under occlusion successfully. Resulting method can be used with tracking applications to increase target detection and tracking performance, while benefiting from the increased association accuracy provided by tracker algorithms.

There are also some notable failure cases, as mentioned in Chapter 4. Segmentation method is mainly based on intra-frame level features. Occlusion cases without any significant convexity defect can occur, as seen in Figure 5.1. Therefore, some detected occlusion cases cannot be segmented using convexity measure. This problem can be solved by using inter frame level data, as intra frame level lacks necessary information. Depending only on intra frame level information is the main reason behind this fail case. Thus, introduction of inter frame features into segmentation stage can be a further work to increase performance of the proposed solution.

(a) Image (b) Foreground Mask

Figure 5.1: Occlusion case without significant convexity defect.

Furthermore, assumption that vehicle shape is convex unless there is occlusion does not hold for some vehicles with uncommon shapes such as crane carriers and vehicles with construction equipment. Vehicles without a convex shape will be detected as occlusion cases, and segmented erroneously using the proposed method. Therefore, these false positive detection and segmentation events should be handled either utilizing inter frame data, or extending assumption on object geometry such that vehicles with uncommon shape can also satisfy.

REFERENCES

[1] Nil Goyette, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, and Prakash Ishwar. Changedetection. net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 1–8. IEEE, 2012.

[2] Nijad Al-Najdawi, Helmut E Bez, Jyoti Singhai, and Eran A Edirisinghe. A survey of cast shadow detection algorithms. *Pattern Recognition Letters*, 33(6):752–764, 2012.

[3] Marko Heikkilä and Matti Pietikäinen. A texture-based method for modeling the background and detecting moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):657–662, 2006.

[4] Pierre-Luc St-Charles, Guillaume-Alexandre Bilodeau, and Robert Bergevin. Subsense: A universal change detection method with local adaptive sensitivity. *Image Processing, IEEE Transactions on*, 24(1):359–373, 2015.

[5] Pawn image. `http://courses.cs.washington.edu/courses/cse458/06au/reading/lighting_tutorial/final_render.jpg`.

[6] J Stander, Roland Mech, and Jörn Ostermann. Detection of moving cast shadows for object segmentation. *Multimedia, IEEE Transactions on*, 1(1):65–76, 1999.

[7] Andrea Prati, Ivana Mikic, Mohan M Trivedi, and Rita Cucchiara. Detecting moving shadows: algorithms and evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(7):918–923, 2003.

[8] Andres Sanin, Conrad Sanderson, and Brian C Lovell. Shadow detection: A survey and comparative evaluation of recent methods. *Pattern recognition*, 45(4):1684–1695, 2012.

[9] Rafael Rodriguez-Gomez, Enrique J Fernandez-Sanchez, Javier Diaz, and Eduardo Ros. Fpga implementation for real-time background subtraction based on horprasert model. *Sensors*, 12(1):585–611, 2012.

[10] Xuefeng Song and Ram Nevatia. A model-based vehicle segmentation method for tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1124–1131. IEEE, 2005.

[11] Clement Chun Cheong Pang, William Wai Leung Lam, and Nelson Hon Ching Yung. A novel method for resolving vehicle occlusion in a monocular traffic-image sequence. *Intelligent Transportation Systems, IEEE Transactions on*, 5(3):129–141, 2004.

[12] Clement Chun Cheong Pang, William Wai Leung Lam, and Nelson Hon Ching Yung. A method for vehicle count in the presence of multiple-vehicle occlusions in traffic images. *Intelligent Transportation Systems, IEEE Transactions on*, 8(3):441–459, 2007.

[13] Vahid Heidari and Mohammad Reza Ahmadzadeh. A method for vehicle classification and resolving vehicle occlusion in traffic images. In *Pattern Recognition and Image Analysis (PRIA), 2013 First Iranian Conference on*, pages 1–6. IEEE, 2013.

[14] Hengjun Yue, Jian Wu, Yanyan Cao, and Zhiming Cui. Research on moving vehicle detection in the presence of occlusion. In *Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on*, pages 514–517. IEEE, 2010.

[15] Roya Rad and Mansour Jamzad. Real time classification and tracking of multiple vehicles in highways. *Pattern Recognition Letters*, 26(10):1597–1607, 2005.

[16] Alberto Faro, Daniela Giordano, and Concetto Spampinato. Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1398–1412, 2011.

[17] Wei Zhang, QM Jonathan Wu, Xiaokang Yang, and Xiangzhong Fang. Multilevel framework to detect and handle vehicle occlusion. *Intelligent Transportation Systems, IEEE Transactions on*, 9(1):161–174, 2008.

[18] Vu Pham, Phong Vo, Vu Thanh Hung, et al. Gpu implementation of extended gaussian mixture model for background subtraction. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2010 IEEE RIVF International Conference on*, pages 1–4. IEEE, 2010.

[19] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[20] HR Kashani and GN Saridis. Intelligent control for urban traffic systems. *Automatica*, 19(2):191–197, 1983.

[21] Bob McQueen and Judy McQueen. *Intelligent transportation systems architectures*. 1999.

[22] Norbert Buch, Sergio Velastin, James Orwell, et al. A review of computer vision techniques for the analysis of urban traffic. *Intelligent Transportation Systems, IEEE Transactions on*, 12(3):920–939, 2011.

[23] Xinping Yan, Hui Zhang, and Chaozhong Wu. Research and development of intelligent transportation systems. In *Distributed Computing and Applications to Business, Engineering & Science (DCABES), 2012 11th International Symposium on*, pages 321–327. IEEE, 2012.

[24] Alan M McIvor. Background subtraction techniques. *Proc. of Image and Vision Computing*, 4:3099–3104, 2000.

[25] Kinjal A Joshi and Darshak G Thakore. A survey on moving object detection and tracking in video surveillance system. *IJSCE, ISSN*, pages 2231–2307, 2012.

[26] A Merin Antony and J Anitha. A survey of moving object segmentation methods. *International Journalof Advanced Research in Electronics and Communication Engineering (IJARECE) Volume*, 1, 2012.

[27] Shireen Y Elhabian, Khaled M El-Sayed, and Sumaya H Ahmed. Moving object detection in spatial domain using background removal techniques-state-of-art. *Recent patents on computer science*, 1(1):32–54, 2008.

[28] Alan J Lipton, Hironobu Fujiyoshi, and Raju S Patil. Moving target classification and tracking from real-time video. In *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*, pages 8–14. IEEE, 1998.

[29] Robert T Collins, Alan Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt, et al. *A system for video surveillance and monitoring*, volume 2. Carnegie Mellon University, the Robotics Institute Pittsburg, 2000.

[30] Md Atiqur Rahman Ahad, Joo Kooi Tan, Hyoungseop Kim, and Seiji Ishikawa. Motion history image: its variants and applications. *Machine Vision and Applications*, 23(2):255–281, 2012.

[31] James W Davis. Hierarchical motion history images for recognizing human motion. In *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, pages 39–46. IEEE, 2001.

[32] Christof Ridder, Olaf Munkelt, and Harald Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Proceedings of International Conference on recent Advances in Mechatronics*, pages 193–199. Citeseer, 1995.

[33] KP Karman and Achim von Brandt. Moving object recognition using an adaptive background memory in time-varying image processing and moving object recognition. *Capellini, Ed*, 2:297307.

[34] K-P Karmann, Achim V Brandt, and Rainer Gerl. Moving object segmentation based on adaptive reference images. In *5. European Signal Processing Conference.*, volume 2, pages 951–954, 1990.

[35] Dieter Koller, J Weber, T Huang, J Malik, G Ogasawara, B Rao, and S Russell. Towards robust automatic traffic scene analysis in real-time. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 126–131. IEEE, 1994.

[36] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[37] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[38] Saeid Fazli, Hamed Moradi Pour, and Hamed Bouzari. Multiple object tracking using improved gmm-based motion segmentation. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, volume 2, pages 1130–1133. IEEE, 2009.

[39] Wei Zhang, Xiang Zhong Fang, and Xiaokang Yang. Moving vehicles segmentation based on bayesian framework for gaussian motion model. *Pattern Recognition Letters*, 27(9):956–967, 2006.

[40] Cristiano Premebida, Gonçalo Monteiro, Urbano Nunes, and Paulo Peixoto. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 1044–1049. IEEE, 2007.

[41] Rui Tan, Hong Huo, Jin Qian, and Tao Fang. Traffic video segmentation using adaptive-k gaussian mixture model. In *Advances in Machine Vision, Image Processing, and Pattern Analysis*, pages 125–134. Springer, 2006.

[42] Peter S Maybeck and Brian D Smith. Multiple model tracker based on gaussian mixture reduction for maneuvering targets in clutter. In *Information Fusion, 2005 8th International Conference on*, volume 1, pages 8–pp. IEEE, 2005.

[43] Wei Zhang, QM Jonathan Wu, Xiaokang Yang, and Xiangzhong Fang. Multilevel framework to detect and handle vehicle occlusion. *Intelligent Transportation Systems, IEEE Transactions on*, 9(1):161–174, 2008.

[44] Piotr Dalka. Detection and segmentation of moving vehicles and trains using gaussian mixtures, shadow detection and morphological processing. *Machine Graphics & Vision International Journal*, 15(3):339–348, 2006.

[45] S Cheung Sen-Ching and Chandrika Kamath. Robust techniques for background subtraction in urban traffic video. In *Electronic Imaging 2004*, pages 881–892. International Society for Optics and Photonics, 2004.

[46] Dar-Shyang Lee. Effective gaussian mixture learning for video background subtraction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):827–832, 2005.

[47] Mahfuzul Haque, Manzur M Murshed, and Manoranjan Paul. Improved gaussian mixtures for robust object detection by adaptive multi-background generation. In *ICPR*, pages 1–4, 2008.

[48] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer, 2002.

[49] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.

[50] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *Computer Vision—ECCV 2000*, pages 751–767. Springer, 2000.

[51] Zezhi Chen and Tim Ellis. Self-adaptive gaussian mixture model for urban traffic monitoring system. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1769–1776. IEEE, 2011.

[52] Anurag Mittal and Nikos Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–302. IEEE, 2004.

[53] Yaser Sheikh and Mubarak Shah. Bayesian modeling of dynamic scenes for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1778–1792, 2005.

[54] Marc Van Droogenbroeck and Olivier Barnich. Vibe: A disruptive method for background subtraction. *Background Modeling and Foreground Detection for Video Surveillance*, 2014.

[55] Marc Van Droogenbroeck and Olivier Paquot. Background subtraction: Experiments and improvements for vibe. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 32–37. IEEE, 2012.

[56] Martin Hofmann, Philipp Tiefenbacher, and Gerhard Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 38–43. IEEE, 2012.

[57] Olivier Barnich and Marc Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *Image Processing, IEEE Transactions on*, 20(6):1709–1724, 2011.

[58] Parisa Darvish Zadeh Varcheie, Michael Sills-Lavoie, and Guillaume-Alexandre Bilodeau. A multiscale region-based motion detection and background subtraction algorithm. *Sensors*, 10(2):1041–1061, 2010.

[59] Yu-Ting Chen, Chu-Song Chen, Chun-Rong Huang, and Yi-Ping Hung. Efficient hierarchical method for background subtraction. *Pattern Recognition*, 40(10):2706–2715, 2007.

[60] Wonjun Kim and Changick Kim. Background subtraction for dynamic texture scenes using fuzzy color histograms. *Signal Processing Letters, IEEE*, 19(3):127–130, 2012.

[61] Guillaume-Alexandre Bilodeau, Jean-Philippe Jodoin, and Nicolas Saunier. Change detection in feature space using local binary similarity patterns. In *Computer and Robot Vision (CRV), 2013 International Conference on*, pages 106–112. IEEE, 2013.

[62] Pierre-Luc St-Charles and Guillaume-Alexandre Bilodeau. Improving background subtraction using local binary similarity patterns. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 509–515. IEEE, 2014.

[63] Thanarat Horprasert, David Harwood, and Larry S Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *IEEE ICCV*, volume 99, pages 1–19, 1999.

[64] Jia-Bin Huang and Chu-Song Chen. Moving cast shadow detection using physics-based features. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2310–2317. IEEE, 2009.

[65] Nicolas Martel-Brisson and André Zaccarin. Kernel-based learning of cast shadows from a physical model of light sources and surfaces for low-level segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[66] Zhou Liu, Kaiqi Huang, Tieniu Tan, and Liangsheng Wang. Cast shadow removal combining local and global features. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[67] Ajay J Joshi and Nikolaos P Papanikolopoulos. Learning to detect moving shadows in dynamic environments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):2055–2063, 2008.

[68] Nicolas Martel-Brisson and Andre Zaccarin. Learning and removing cast shadows through a multidistribution approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(7):1133–1146, 2007.

[69] Fatih Porikli and Jay Thornton. Shadow flow: A recursive method to learn moving cast shadows. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 891–898. IEEE, 2005.

[70] Jun-Wei Hsieh, Wen-Fong Hu, Chia-Jung Chang, and Yung-Sheng Chen. Shadow elimination for effective moving object detection by gaussian shadow modeling. *Image and Vision Computing*, 21(6):505–516, 2003.

[71] Akio Yoneyama, Chia H Yeh, and CC Jay Kuo. Moving cast shadow elimination for robust vehicle extraction based on 2d joint vehicle/shadow models. In *Advanced Video and Signal Based Surveillance, 2003. Proceedings. IEEE Conference on*, pages 229–236. IEEE, 2003.

[72] Henri Nicolas and Jean-Marie Pinel. Joint moving cast shadows segmentation and light source detection in video sequences. *Signal processing: Image communication*, 21(1):22–43, 2006.

[73] Liu Zhi Fang, Wang Yun Qiong, and You Zhi Sheng. A method to segment moving vehicle cast shadow based on wavelet transform. *Pattern Recognition Letters*, 29(16):2182–2188, 2008.

[74] Chia-Chih Chen and Jake K Aggarwal. Human shadow removal with unknown light source. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2407–2410. IEEE, 2010.

[75] Ying-Li Tian, Max Lu, and Arun Hampapur. Robust and efficient foreground analysis for real-time video surveillance. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 1182–1187. IEEE, 2005.

[76] Omar Javed and Mubarak Shah. Tracking and object classification for automated surveillance. In *Computer Vision—ECCV 2002*, pages 343–357. Springer, 2002.

[77] Dong Xu, Xuelong Li, Zhengkai Liu, and Yuan Yuan. Cast shadow detection in video segmentation. *Pattern Recognition Letters*, 26(1):91–99, 2005.

[78] Wei Zhang, Xiang Zhong Fang, and Yi Xu. Detection of moving cast shadows using image orthogonal transform. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 626–629. IEEE, 2006.

[79] Yang Wang, Kia-Fock Loe, and Jian-Kang Wu. A dynamic conditional random field model for foreground and shadow segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(2):279–289, 2006.

[80] Rui Qin, Shengcai Liao, Zhen Lei, and Stan Z Li. Moving cast shadow removal based on local descriptors. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1377–1380. IEEE, 2010.

[81] Alessandro Leone and Cosimo Distante. Shadow detection for moving objects based on texture analysis. *Pattern Recognition*, 40(4):1222–1233, 2007.

[82] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *Intelligent Transportation Systems, IEEE Transactions on*, 1(2):108–118, 2000.

[83] Harini Veeraraghavan, Osama Masoud, and Nikolaos P Papanikolopoulos. Computer vision algorithms for intersection monitoring. *Intelligent Transportation Systems, IEEE Transactions on*, 4(2):78–89, 2003.

[84] Zezhi Chen, Tim Ellis, Sergio Velastin, et al. Vehicle detection, tracking and classification in urban traffic. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 951–956. IEEE, 2012.

[85] Massimo Bertozzi, Alberto Broggi, Alessandra Fascioli, and Stefano Nichele. Stereo vision-based vehicle detection. In *IEEE Intelligent Vehicles Symposium*, pages 39–44. Citeseer, 2000.

[86] Norbert Buch, Sergio Velastin, James Orwell, et al. A review of computer vision techniques for the analysis of urban traffic. *Intelligent Transportation Systems, IEEE Transactions on*, 12(3):920–939, 2011.

[87] Chukka Srinivas, Anthony Hoogs, Glen Brooksby, Wensheng Hu, et al. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 666–673. IEEE, 2006.

[88] Surendra Gupte, Osama Masoud, Robert FK Martin, and Nikolaos P Papanikolopoulos. Detection and classification of vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, 3(1):37–47, 2002.

[89] Chung-Lin Huang and Wen-Chieh Liao. A vision-based vehicle identification system. In *null*, pages 364–367. IEEE, 2004.

[90] Erhan Baş, F Sibel Salman, et al. Automatic vehicle counting from video for traffic flow analysis. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 392–397. Ieee, 2007.

[91] Tao Yang, Quan Pan, Jing Li, and Stan Z Li. Real-time multiple objects tracking with occlusion handling in dynamic scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 970–975. IEEE, 2005.

[92] Nelson HC Yung and Andrew HS Lai. Detection of vehicle occlusion using a generalized deformable model. In *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, volume 4, pages 154–157. IEEE, 1998.

[93] Andrew Senior, Arun Hampapur, Ying-Li Tian, Lisa Brown, Sharath Pankanti, and Ruud Bolle. Appearance models for occlusion handling. *Image and Vision Computing*, 24(11):1233–1243, 2006.

[94] Ye Li, Bin Tian, Bo Li, Gang Xiong, FengHua Zhu, and Kunfeng Wang. Vehicle detection with a part-based model for complex traffic conditions. In *Vehicular Electronics and Safety (ICVES), 2013 IEEE International Conference on*, pages 110–113. IEEE, 2013.

[95] Bing-Fei Wu, Chih-Chung Kao, Cheng-Lung Jen, Yen-Feng Li, Ying-Han Chen, and Jhy-Hong Juang. A relative-discriminative-histogram-of-oriented-gradients-based particle filter approach to vehicle occlusion handling and tracking. *Industrial Electronics, IEEE Transactions on*, 61(8):4228–4237, 2014.

[96] Jiyan Pan and Bo Hu. Robust occlusion handling in object tracking. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[97] Guang Shu, Afshin Dehghan, Omar Oreifej, Emily Hand, and Mubarak Shah. Part-based multiple-person tracking with partial occlusion handling. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1815–1821. IEEE, 2012.

[98] Ye Li, Bo Li, Bin Tian, and Qingming Yao. Vehicle detection based on the and–or graph for congested traffic conditions. *Intelligent Transportation Systems, IEEE Transactions on*, 14(2):984–993, 2013.

[99] M Ben. Principles of concurrent and distributed programming. 2006.

[100] Wayne A Wickelgren. Speed-accuracy tradeoff and information processing dynamics. *Acta psychologica*, 41(1):67–85, 1977.

[101] Andres Sanin, Conrad Sanderson, and Brian C Lovell. Improved shadow removal for robust person tracking in surveillance scenarios. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 141–144. IEEE, 2010.

[102] Ning Song Peng, Jie Yang, and Zhi Liu. Mean shift blob tracking with kernel histogram filtering and hypothesis testing. *Pattern Recognition Letters*, 26(5):605–614, 2005.

[103] Jack Sklansky. Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, 1(2):79–83, 1982.

[104] Gary Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[105] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. 1968.

[106] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[107] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[108] Khammapun Khantanapoka and Krisana Chinnasarn. Pathfinding of 2d & 3d game real-time strategy with depth direction a* algorithm for multi-layer. In *Natural Language Processing, 2009. SNLP'09. Eighth International Symposium on*, pages 184–188. IEEE, 2009.

[109] Opencv library. `https://www.opencv.org`.

[110] Opengl library. `https://www.opengl.org`.

[111] freeglut library. `http://freeglut.sourceforge.net/`.

[112] Justin Heyes Jones. A* search algrorithm implementation. `https://github.com/justinhj/astar-algorithm-cpp`.

[113] A. Sanin. Shadow detection code, 2012.