Vision based solutions for autonomous navigation

Karunsekera, H. Hasith Ruchiran

2020

Karunsekera, H. H. R. (2020). Vision based solutions for autonomous navigation. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/146366>

<https://doi.org/10.32657/10356/146366>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 17:33:33 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

VISION BASED SOLUTIONS FOR AUTONOMOUS NAVIGATION

H. HASITH RUCHIRAN KARUNSEKERA

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

2020

VISION BASED SOLUTIONS FOR AUTONOMOUS NAVIGATION

H. HASITH RUCHIRAN KARUNASEKERA

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

14 - 05 - 2020

.....
Date

.....
H. Hasith Ruchiran Karunasekera

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

..... 15 - 05 - 2020

Date

.....

Assoc. Prof. Wang Han

Authorship Attribution Statement

This thesis contains material from two paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Chapter 4 is published as H. Karunasekera, H. Wang and H. Zhang. Energy Minimization Approach for Negative Obstacle Region Detection. *IEEE Transactions on Vehicular Technology* **68-12**, 11668-11678 (2019). DOI: 10.1109/TVT.2019.2948684.

The contributions of the co-authors are as follows:

- A/Prof Wang provided the initial direction and edited the manuscript.
- I designed the study and implemented the idea as a software code, which was used for generating and testing results. I also analyzed the data and prepared the manuscript drafts.
- Mr. Zhang helped me in data collection using stereo cameras and performing calibration of stereo cameras.

Chapter 5 is published as H. Karunasekera, H. Wang and H. Zhang. Multiple Object Tracking with Attention to Appearance, Structure, Motion and Size. *IEEE Access* **7**, 104423-104434 (2019). DOI: 10.1109/ACCESS.2019.2932301.

The contributions of the co-authors are as follows:

- A/Prof Wang provided the initial direction and edited the manuscript.
- I designed the study and implemented the idea as a software code, which was used for generating and testing results. I also analyzed the data and prepared the manuscript drafts.
- Mr. Zhang helped in revising the manuscript drafts.

14 - 05 - 2020

.....
Date

.....
H. Hasith Ruchiran Karunasekera

Acknowledgement

I would like to express my heartfelt gratitude to my dear supervisor, Associate Professor Wang Han, for all his support, guidance and motivation given to succeed in my postgraduate studies.

I am equally grateful to all the members of the Thesis Advisory Committee (TAC) and the other examiners, for allocating their time from busy schedules to review my thesis and to be present at my oral examination. I wish to appreciate the support given by my fellow colleagues especially, Zhang Handuo, Zhou Hui and Mahdi Abolfazli during the duration of my PhD study.

Added to that, I would like to thank my dear wife, parents and family for their understanding and support given to me whenever requested. Finally I would like to sincerely appreciate all the efforts of the research community whose contribution has made the current stand of this area and especially who has made their work and codes publicly available to support the others.

Contents

Statement of Originality	ii
Supervisor Declaration Statement	iii
Authorship Attribution Statement	iv
Acknowledgement	v
Table of Contents	vi
Summary	ix
List of Figures	xi
List of Tables	xiii
Nomenclature	xiv
1 Introduction	1
1.1 Motivation	4
1.2 Objectives	5
1.3 Major contributions of the Thesis	6
1.4 Organization of the Thesis	8
2 Literature Review	9
2.1 Obstacle Detection	9
2.1.1 Positive Obstacle Detection	10
2.1.2 Negative Obstacle Detection	15
2.1.3 Object Detection	18
2.2 Object Tracking	25
2.2.1 Single Object Tracking	26
2.2.2 Multiple Object Tracking	26
2.3 Concluding Remarks	32

3	Detecting Obstacles for Real-time Navigation	36
3.1	Introduction	36
3.2	The Proposed Method	37
3.2.1	Understanding the Drivable Path	37
3.2.2	Class Independent Positive Obstacle Detection	43
3.2.3	Instance-level Segmentation	44
3.3	Results and Discussion	46
3.3.1	Road Slope Accuracy	48
3.3.2	Execution Time	51
3.3.3	Instance Segmentation Performance	51
3.4	Concluding Remarks	54
4	Negative Obstacle Region Detection	55
4.1	Introduction	55
4.2	The Proposed Method	57
4.2.1	Introducing the Energy Function	58
4.2.2	Superpixel Segmentation	59
4.2.3	Defining Region of Interest	60
4.2.4	Initial Negative Obstacle Region	63
4.2.5	Energy Function Initiation and Optimization	65
4.2.6	Negative Obstacle Region Detection	68
4.3	Results and Discussion	68
4.3.1	Parameters tuning	70
4.3.2	Comparison with others	71
4.4	Concluding Remarks	79
5	Multiple Object Tracking with attention to Appearance, Structure, Motion and Size	81
5.1	Introduction	81
5.2	The Proposed Method	82
5.2.1	Matching Cost Computation	84
5.2.2	Track Match and Update	91
5.2.3	Tracking False Negatives for Stable Tracks	92

5.3	Results and Discussion	93
5.3.1	Detections	94
5.3.2	Ablation Study and Parameters Tuning	95
5.3.3	Comparison with Others	98
5.4	Concluding Remarks	102
6	Learning Features for Multiple Object Tracking as an Extension to Detection	104
6.1	Introduction	104
6.2	The Proposed Method	106
6.2.1	Network Architecture	107
6.2.2	Network Training	109
6.2.3	Network Inference	111
6.2.4	Tracking	111
6.3	Results and Discussion	113
6.3.1	Comparison to Baseline	113
6.3.2	Analysis on Running Speed	116
6.3.3	Comparison to Others	117
6.4	Concluding Remarks	120
7	Conclusion and Recommendations	121
7.1	Conclusion	121
7.2	Recommendations for future research	122
	List of Publications	124
	References	126
	Appendix A: Stereo vision methods and derivations	147

Summary

This thesis presents a study on computer vision solutions for autonomous navigation. Among many different functionalities in autonomous navigation, areas covered in this study include obstacle detection and multiple object tracking.

The study on obstacle detection is performed under two main sub-categories, namely, positive and negative obstacle detection. Positive obstacles are the obstacles that lie on the road surface such as vehicles, pedestrians etc., while the negative obstacles are the ones below the road surface such as holes and potholes. The first challenge addressed in this study is to identify the obstacles on the drivable road surface with depth information. The key contribution made in the first part is to propose an efficient framework for understanding the road surface, calculating the road angle, detecting obstacles in class agnostic manner and instance segmentation of objects, using stereo vision. The proposed framework has been tested in the real world with live data on public roads in real time, with different weather conditions and has shown to be effective.

The second part of the study is on negative obstacle detection for the safe navigation of the robot. The key contribution in this section is to propose an energy minimization approach for negative obstacle region detection, using stereo output, colour information and saliency detection. The proposed concept has been tested on different environments such as concrete-road, tar-road and corridor based scenarios. Comparison with the recent work, has shown the effectiveness of the proposed method.

The third contribution of the thesis is to propose an efficient framework for the multiple object tracking task that can achieve real time performance with the state-of-the-art accuracy. Tracking framework is proposed following the tracking-by-detection architecture. Matching cost is calculated by combining grid based color histogram matching, grid based structure matching, predicted object motion matching and predicted size based matching. Proposed framework has gained the expected efficiency, achieving 150+ fps for KITTI data and 27.0 fps and 17.1 fps for MOT 17 training and test data respectively, with comparable accuracy to the

recent work.

Final contribution of the study is to learn good features to track as an extension to an existing detection network. Object detection is the first step of the tracking-by-detection architecture. Hence, learning good features for tracking as an extension to the detection network, helps to reduce the computational complexity of the whole pipeline. Learnt features have resulted in improved tracking accuracy compared to the hand crafted ones (in the above mentioned work), at the cost of reduction in tracking speed. 3D convolutional network architecture is proposed to consider the inter-dependencies in spatio-temporal domains.

List of Figures

1.1	Main Functionalities of Autonomous Navigation	2
2.1	v-disparity and u-disparity map generation	11
3.1	u-disparity based large obstacle removal	38
3.2	Road profile extraction flow chart	39
3.3	Road profile extraction process	40
3.4	Slope calculation flow chart	41
3.5	Slope angle and pitch angle calculation in different road segments .	42
3.6	Surface normal estimation	42
3.7	Flow chart for positive obstacle detection in class agnostic manner.	44
3.8	Obstacle bounding box generating process	45
3.9	Instance level segmentation	47
3.10	Stereo rig mounts and vehicles used	48
3.11	Slope detection experimental results	50
3.12	Instance segmentation results for KITTI data	53
3.13	: CPU and GPU utilization over time by Proposed method and Mask-RCNN method	54
4.1	Geometry of negative obstacle detection	56
4.2	Proposed negative obstacle detection pipeline	58
4.3	Region of interest extraction	61
4.4	Initial Negative Obstacle Region N profiling process	64
4.5	Detection of negative obstacle region	67
4.6	Parameter fine tuning and selection process	72
4.7	Energy minimization in iterations.	73
4.8	RoI based plane fitting process	76
4.9	Negative obstacle region detection examples	77
4.10	Failed cases in negative obstacle detection.	78
4.11	Negative obstacle detection accuracy variation with the distance to the negative obstacle from the camera.	79

5.1	Flow chart of the proposed tracking framework	83
5.2	Grid structure and appearance based history of a track	85
5.3	Grid based histogram matching against single histogram matching .	87
5.4	Linear Binary Pattern (LBP)	89
5.5	Maintaining the track in case of a false negative in detection	92
5.6	Comparison of MOTA changes for different dissimilarity threshold th_{MC} and number of grid cells m	98
5.7	Tracking results on few frames	101
6.1	Network architecture of the proposed method	106
6.2	Network architecture of CenterNet DLA-34	108
6.3	Details of the 3D Network	109
6.4	Images inserting strategy during inference	112
6.5	MOT results on few sequences in MOT16 data	119

List of Tables

3.1	Slope Estimation Accuracy	49
3.2	Slope Detection Consistency	50
3.3	Execution Time	51
3.4	Instance Segmentation Performance	52
4.1	Selected Parameters for Testing	71
4.2	Negative Obstacle Region Detection Results Comparison	74
4.3	Algorithm Execution Time per Frame	78
5.1	Ablation Study in MOT17 and KITTI Tracking Training Sets	96
5.2	Results in MOT17 and KITTI Tracking Test Sets	99
5.3	Comparison of Results on Different Detection Methods on MOT17- 3 rd sequence	102
6.1	MOT Performance Compared to Baseline Method	114
6.2	Different Distance Measures for Baseline	115
6.3	MOT Accuracy for Improvements on MOTeD	116
6.4	MOT Accuracy and Processing Speed based on number of frames processed for tracking at each inference	117
6.5	MOT Performance on MOT16 Test Data	117
6.6	MOTeD vs. MASS Performance on MOT16 Training Data	119

Nomenclature

δ Small constant

$3Drp$ 3D road profile

$3Drp_seg_j$ 3D road profile of j^{th} road segment

α Angle to the end of a negative obstacle from the camera

β Constant value

θ_n Negative obstacle visible angle from the camera set on ego-vehicle

$app(i, j)$ Appearance based dissimilarity between i^{th} and j^{th} objects

AUC Area Under Curve

b Baseline of the stereo rig

bb Bounding box

bg Backgorund

C Small integer constant

c Column number for a given matrix

C_p Large penalty

$C_{clrdiff}$ Constant value

$CFTP$ CFTP superpixel segmentation mechanism

$clrdiff$ Distance in the RGB colour domain

cls Object class

$corr$ Correlation

$Cosine$ Cosine distance

D Feature distance

d	Disparity value
$d(fm, fn)$	Feature distance between fm and fn feature vectors
d_{max}^{bb}	Maximum disparity value for a given bounding box
d_{min}^{bb}	Minimum disparity value for a given bounding box
d_r^{rp}	Disparity value of refined road profile at row r
d_r	Disparity value corresponding to the peak intensity value for each row of the v-disparity map
d_{max}	Maximum number of disparity levels
$d_{r,c}$	Disparity value at point (r, c)
d_{th}	Threshold value for comparing disparity values that belong to road
$d_{u,road}$	Disparity value corresponding to the road in u^{th} image row
<i>Dark</i>	Darker appearance based negative obstacle detection method
<i>det</i>	Detection
$dis(fm, fn)$	Dissimilarity between feature vectors fm and fn
$dist$	Maximum distance from the plane to a given point to classify whether that point belongs in the plane or not
E	Energy
fg	Foreground
fm	Feature vector of an object in the frame at time t
fn	Feature vector of an object in an adjacent frame
H	Histogram
h	Image height or the number of rows in image
h_{neg}^{vis}	Visible height of the negative obstacle

h_{bb}	Height of a bounding box
h_{min}	Minimum height of an obstacle to consider it as a large obstacle
h_{neg}^{min}	Minimum height difference to declare a point belongs to a negative region
H_{veh}	Height of the camera set on ego-vehicle from the road
HSV	Hue-Saturation-Value colour space
i	Variable that holds numerical values
id	Object identity
$init_rp$	Initial road profile
$iou(i, j)$	Size difference based dissimilarity between i^{th} and j^{th} objects
j	Variable that holds numerical values
k	Total number of superpixels in a region of interest
l	Number of concatenated histograms kept in the memory for each track
L_{tri}	Triplet Loss
$loc(i, j)$	Location based dissimilarity between i^{th} and j^{th} objects
m	Number of cells in the grid
$MC(i, j)$	Matching cost between i^{th} and j^{th} objects
N	Initial Negative Obstacle Region
n	Total number of pixels in a region of interest
N_b	Batch size
n_f	Total number of elements in the feature vector
n_m	Total number of pixels in the m^{th} grid cell
N_s	Surface normal vector
n_{dst}	Value used to normalize the distance measure

N_{sail}	Salient road region
N_{st}	Potential negative obstacle region from 3D information
$occl$	Occluded grid cell
P	Probability
p	Pixel in a image
$p(u, v)$	Pixel at image (u, v) coordinates
p_{obs}	Pixel that belongs to an obstacle
p_{road}	Pixel that belongs to the road
$Plane$	Plane fitting based negative obstacle detection method
$Plane_RoI$	Plane fitting on the region of interest based negative obstacle detection method
PR	Precision-Recall
$pred$	Prediction
$Proposed$	Proposed method in the chapter
q	Consecutive number of frames that track is undetected
R	Range to a negative obstacle from the ego-vehicle
r	Row number for a given matrix
R_{hrz}	Threshold number of rows to stop refined road profile
R_{low}	Threshold for number of rows in the v-disparity image to search for the initial point of the rp
ra	RoI Align
rd	Road
re	Reshape

ROC	Receiver Operating Characteristics
rp	Refined road profile
rp_{init}	Initial road point in refined road profile
s	Superpixel
SAE	Sum of Absolute Error
SAE	Sum of Squared Error
$sim(fm, fn)$	Similarity between feature vectors fm and fn
$SLIC$	SLIC superpixel segmentation mechanism
sm	Similarity
$str(i, j)$	Structure based dissimilarity between i^{th} and j^{th} objects
t	Frame at time t
th_h	High threshold used for hysteresis thresholding
th_l	Low threshold used for hysteresis thresholding
th_{det}	Threshold to qualify as a valid object by the detection network
th_{iou}	Threshold for overlap of objects for track association
th_{MC}	Threshold for matching cost for track association
u	x direction in image coordinate system
u_{span}	Connected line in the refined u-disparity map
u_{RoI}	Region of Interest in u-disparity image
$u_{th}(d)$	Threshold value to segment the u-disparity map to identify peaks for a given d value
v	y direction in image coordinate system
v_{disp}	v-disparity image

v_disp_{th}	Threshold value to compare the disparity difference in adjacent rows in rp calculation
$v_{r,c}$	Intensity value of a point at (r, c) in the v-disparity image
W	Weight parameter
w	Width of a negative obstacle
w_{bb}	Width of a bounding box
X	Distance to a point in the real world in the X direction
x_{bb}	x direction image coordinates of bounding box center
Y	Distance to a point in the real world in the Y direction
y_{max}^{bb}	Rightmost y direction value for a given bounding box
y_{bb}	y direction image coordinates of bounding box center
Z	Distance to a point in the real world in the Z direction
ADAS	Advance Driver Assistance System
BING	Binarized Normed Gradients
CF	Correlation Filter
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DEM	Digital Elevation Map
FN	False Negatives
FP	False Positives
FPS	Frames Per Second
GPS	Global Positioning System
GPU	Graphical Processing Unit

HOG Histogram of Oriented Gradients

IDS Identity Switches

IMU Inertial Measurement Unit

KITTI KITTI benchmark dataset

LBP Linear Binary Pattern

LBPH Linear Binary Pattern Histogram

LSTM Long Short-Term Memory

ML Mostly Lost

MOT Multiple Object Tracking

MOT16 Multiple Object Tracking 2016 dataset

MOT17 Multiple Object Tracking 2017 dataset

MOTA Multiple Object Tracking Accuracy

MOTP Multiple Object Tracking Precision

MT Mostly Track

RAM Random Access Memory

RGB Red-Green-Blue colour space

RNN Recurrent Neural Network

RoI Region of Interest

ROS Robotic Operating System

SIFT Scale Invariant Feature Transform

SVM Support Vector Machine

TP True Positives

TPR True Positive Rate

UGV Unmanned Ground Vehicle

vSLAM visual Simultaneous Localization and Mapping

Chapter 1

Introduction

Autonomous navigation is the task of traversing an object from one point to another without human intervention. These objects include aerial, ground or sea vehicles. Focus of this study is on the autonomous navigation of ground vehicles, which are also known as Unmanned Ground Vehicles (UGVs). Driver-less cars, cleaning robots, warehouse robots, surveillance robots are among the many interesting applications of UGVs. While, there exists industry level autonomous robots which navigate in controlled environments carrying out various tasks, fully autonomous vehicles on roads is still a challenge. Main challenges include, human safety, presence of various human activities and various dynamic aspects associated with the environment. However, Tesla, Google Car, Uber have shown great advancements on autonomous navigation on roads, amongst these challenges.

Collision free autonomous navigation of a UGV is a combined task of different functionalities as indicated in the Fig. 1.1. These Functionalities are;

1. Obstacle Detection - Identifying obstacles in the environment that can collide with the robot.
2. Object Detection - Detecting and classifying important objects such as humans, vehicles and etc. depending on the application.
3. Object Tracking - Maintaining a unique identity for objects over time for meaningful understanding of the objects
4. Object Motion Prediction - Predicting the motion of the tracked objects. This will help the robot to better perceive the environment, anticipate possible risks and take actions before hand.
5. Action Prediction - Predicting the immediate next action of objects, mainly for humans to avoid collision or to limit the damage

6. Simultaneous Localization and Mapping - Localizing the robot in the environment and building a map of the surroundings to aid in navigational tasks.
7. Path Planning and Obstacle Avoidance - Building a feasible and safe route for the robot to navigate from one place to another considering all above information.

Figure 1.1: Main Functionalities of Autonomous Navigation

In order to achieve the objectives of above tasks, a robot needs to sense the environment. For this purpose, robots rely on different sensor inputs. Main sensors used to perceive the environment include;

1. Laser/Lidar - Provides a 3D point cloud of the scene with depth information. Measurements from laser are of high accuracy. Lidar emits infrared light waves to obtain these information.
2. Radar - Similar to lidar, radar provides point cloud information but is less in accuracy in the short range comparatively. But radar is cheaper than lidar and has a higher sensing range. Radar uses radio waves instead of light waves.
3. GPS - It's an acronym for Global Positioning System and used to localize the vehicle in a global map.

4. IMU - Stands for Inertial Measurement Unit where it is useful to measure the velocity, position changes and roll/pitch/yaw information of the ego vehicle.
5. Sonar - Sonar is similar to lidar or radar but used in very short range sensing, mainly to aid in avoiding collision with near by obstacles.
6. Vision/Camera - Camera is a passive sensor which does not emit any signal for measurements. Cameras have the ability to sense colour information in comparison to other sensors.

Among the aforementioned sensors Vision/Camera based sensing offers multiple advantages such as;

1. Being a low cost sensory option in-comparison to sensors such as lidar.
2. Being a non-invasive and passive sensor, power consumption of a camera is very low
3. A camera captures a lot of information of the environment, such as colour, objects, object types and depth.

On the other hand, computational power has increased rapidly over the recent past with faster and multiple core CPUs and powerful GPU arrays. These advancements have enabled scientists to run more complex algorithms to extract information from sensor inputs. Especially, computer vision has benefited greatly from these hardware advancements, as multiple algorithms are required to extract the large amount of information present in an image. Furthermore, recent advances in machine learning have contributed towards many breakthroughs in computer vision applications, attracting much attention towards the field. Therefore, vision has become a mainstream sensor input for autonomous navigational tasks.

Though there are multiple advantages, extracting information from images captured by a camera is still a very challenging task. Compared to other sensor systems, vision systems are still not robust enough. Even though there are accurate methods proposed, the operational speed is still not desirable for real time fast moving UGVs. Therefore, extracting meaningful information from vision input is a developing area in research, with many challenges to address.

1.1 Motivation

In the early stages, the main objective of autonomous navigation is to navigate the robot from one point to another, without colliding with obstacles present in the environment. Therefore, laser and radar sensors were the main source of sensory input, as they provide accurate depth information about the objects in-front of the robot. However, truly autonomous navigation requires to understand the objects in the environment and make advance decisions. A vehicle travelling at high speeds should not suddenly apply brakes for a plastic bag on a road, even though plastic bag is detected as an obstacle. Such decisions are made very often, when humans are driving. Humans rely on their vision system to understand the environment around them. Data captured from cameras, contain information to better understand the environment, than that of a laser sensor. Furthermore, cameras are an inexpensive option in comparison to laser sensors. Therefore, vision is becoming a main source of sensory input for autonomous navigation applications. This was a strong motivational drive to conduct the research in vision based autonomous navigation applications.

The amount of data captured, even by a normal camera is of great value. However, arriving at meaningful decisions from that information, in real time is a challenging task. The challenge is in finding a good compromise between speed, accuracy and robustness of the system. While obtaining a very accurate result is time consuming, a fast method may not be robust enough. Illumination variations, weather conditions and different scenarios (such as urban navigation, indoor navigation and terrain navigation) impose challenges on the robustness. Conducting research on all the components of autonomous navigation, (as shown in the Fig. 1.1) is beyond the scope of this research. Therefore, two primary components, namely (1) obstacle detection and (2) multiple object tracking, are studied in this research. Current challenges in these two primary components are listed as below.

1. Obstacle detection can be done very accurately using laser scanners. However, vision based systems have to rely on stereo cameras for depth sensing. Challenges and limitations of using stereo vision are as listed below, which are studied in this work.

- (a) In general, stereo based depth information provides dense information in the near field of view, while in the far region information become sparse.
 - (b) Stereo output provides 3D information for each pixel in the image. Categorizing these 3D points as meaningful objects is a time consuming task.
 - (c) Stereo vision based holes and pothole detection is a rarely studied area and detection accuracy could be improved further.
2. Multiple object tracking is an important task for motion prediction, surveillance, action prediction etc. Limitations and challenges that motivated to conduct this study on object tracking are as below.
- (a) Most research limits the tracking task to one object type, such as pedestrian tracking or vehicle tracking. However, several object classes are present in navigational applications, which needs to be tracked in a single framework.
 - (b) Navigation is a real time application, which require a good trade-off between tracking accuracy and speed. Multiple object tracking is a growing area with much room for improvement in terms of both accuracy and efficiency.

1.2 Objectives

The main objective of the study is to propose vision based frameworks that can achieve real time performance, while being able to compete with the state-of-the-art accuracy. Three major objectives of the study are listed as below.

1. Detecting positive obstacles (such as humans, vehicles, trees etc.) using stereo vision and achieving meaningful segmentation for the resulting 3D points. In order to handle the sparse information from stereo vision in the far range, two stereo sets are proposed such that, one stereo set with lower focal length to see the near range and another stereo set with a bigger focal length to see the far range.

2. Combining stereo information and pixel intensity information to detect negative obstacles such as potholes and ditches on the robot path. Moreover, energy minimization approach is proposed to combine stereo and colour information for the task of negative obstacle region detection.
3. Performing multiple object tracking in real time with satisfactory results. Tracking-by-detection framework is used combining several traditional features, with an emphasis on efficiency. Later, a deep learning based appearance feature is explored to achieve better performance.

1.3 Major contributions of the Thesis

Major contributions of this study can be summarized as listed below.

1. Obstacle detection is an important task for an autonomous navigation system. In this work stereo vision based drivable path detection, class agnostic obstacle detection and instance level segmentation method is proposed. Drivable path is detected following u-disparity and v-disparity maps. In addition with the estimation of the road plane, slope measurement is done using 3D information from stereo vision. Obstacle detection method is proposed by removing the points on the road plane and detecting obstacle points in a class agnostic manner. Furthermore, real-time instance segmentation method is proposed combining the information from the above class agnostic obstacle detection and deep learning based object detector. The method has been tested with four different stereo camera configurations on real world data on driving scenarios. With realistic data collected on the public roads, in different type of weather conditions, the proposed framework is shown to be effective for operating under real time requirements.
2. The main focus on obstacle detection research has been on detecting positive obstacles such as vehicles, pedestrians and anything above the ground plane. An equally important task is to detect negative obstacles such as holes, ditches etc. that lie below the ground surface, which has not been studied extensively in literature. In this work detection of such holes and potholes is proposed as an energy minimization problem. Stereo information combined

with saliency is used to initialize the energy function and colour information is used to optimize the result. The optimized result is then subjected to hysteresis thresholding to arrive at the final negative obstacle region. Experimental results show that the proposed method can successfully detect negative obstacles and run in real time (in the range 14-20 Hz) when using GPU for stereo matching. The proposed method was tested under three configurations with corridor, concrete-road and tar-road scenes. The proposed method perform well compared to the recent works.

3. Objective of multiple object tracking (MOT) is to assign unique track identities for all the objects of interest in a given video sequence. Tracking-by-detection is the most common approach used in addressing MOT problem. In this work, a method is proposed to address MOT by defining a matching cost, based on object motion, appearance, structure and size. Several traditional features were combined to achieve a simple, efficient yet effective tracking framework, that can perform tracking in a class agnostic manner. The proposed architecture can track objects from different classes simultaneously. The appearance and structure based dissimilarity measure is calculated by matching histograms following a grid architecture. Motion and size for each track is predicted using the information from track’s history. These dissimilarity values are then combined and used in the Hungarian Algorithm, in the data association step for track identity assignment. In addition, a method is introduced to address any false detections in stable tracks. The proposed method runs in real time following an online approach. Proposed method is evaluated in both MOT17 benchmark data-set for pedestrian tracking and KITTI benchmark data-set for vehicle tracking using the same system parameters to verify the robustness of the proposed method. The method can achieve state-of-the-art results in both benchmarks.

4. Multiple object tracking (MOT) make use of appearance features and location features. Deeply learnt appearance features provide high tracking accuracy compared to hand-crafted ones at increased computational time. On the other hand, object detection is the initial step when using tracking-by-

detection framework. Therefore, computational complexity of both detection and tracking needs to be considered together for real-time systems. In most of the literature, two networks are used to extract features for the detection task and tracking task. Hence, in this work, a method to learn robust features that are suitable for tracking is proposed, as an extension to a detection network. The proposed method assists in implementing the tracking module at minimal additional computational complexity. Furthermore, appearance features are learnt using a triplet loss in a fractional manner, opposed to the additive manner used in most of person re-identification work. The proposed method is tested in MOT16 human tracking dataset and has achieved satisfactory results, specifically being amongst the top five published online methods.

1.4 Organization of the Thesis

The remainder of the thesis is organized as follows.

In Chapter 2, a literature review is presented on obstacle detection and object tracking. Obstacle detection research is presented under three headings as positive, negative obstacle detection and object detection. Under object tracking, literature is reviewed based on single and multiple object tracking methods.

Chapter 3 presents positive obstacle detection and associated steps in detail, followed by experimental results. Chapter 4 presents the proposed negative obstacle region detection methodology with results and discussion.

Proposed multiple object tracking framework using traditional features is explained in Chapter 5 with results on public benchmarks. In Chapter 6, a deep learning approach is proposed to learn suitable features for tracking, as an extension to an object detection network. Finally the conclusion is presented in Chapter 7, summarizing the research work presented in this thesis with recommendations for future studies.

In Appendix A stereo vision methods and derivations are introduced, as stereo vision is used in both Chapters 3 and 4. Appendix includes, a brief literature review on stereo matching, theory on 3D reconstruction, details on stereo camera setup, camera calibration and stereo matching implementation details.

Chapter 2

Literature Review

Recently many research have been conducted in vision based autonomous navigation components. In this study the main focus is on obstacle detection and object tracking aspects for driving scenarios. Therefore, in this chapter related literature has been reviewed and presented under the above mentioned categorization. UGVs use different sensing methodologies such as laser, radar, vision etc. [1], to understand the environment around them. Literature on obstacle detection and object tracking is reviewed mainly on vision based research work because the focus in this study is on vision sensor (monocular or stereo camera input) based object analysis.

2.1 Obstacle Detection

An obstacle is an object that disturbs the navigation of a vehicle. Hence, obstacle detection is important for the safe navigation of the ego-vehicle, to avoid any collision with them. Obstacles present in the environment can be either positive obstacles or negative obstacles [2]. Positive obstacles are the most intuitive and common class of obstacles, which is defined as an obstacle that lies on top of road surface or the ground plane. Examples for positives obstacles present on roads are pedestrians, vehicles, plants, buildings etc. Many studies have been conducted in positive obstacle detection. Among those, vehicle detection and pedestrian detection are the most widely studied areas. A review on vision based vehicle detection research is presented in [3, 4] while pedestrian detection related research is reviewed in [5]. Negative obstacles are the obstacles that are below the ground surface. Examples for negative obstacles include holes, ditches, potholes etc. Another categorization for obstacles could be based on the class of that object belongs to. These object classes can vary depending on the application and it can include both positive and negative obstacles. Therefore, when reviewing the literature, research on obstacles detection in autonomous driving scenarios can be analyzed under three main cate-

gories. These categories are (a) positive obstacle detection, (b) negative obstacles detection and (c) object detection.

2.1.1 Positive Obstacle Detection

All the objects that lie on top of the ground plane belongs to the super class of positive objects. Therefore, vehicles, pedestrians, pavement, trees & bushes, animals, traffic & other poles, buildings and etc. are the sub-classes that falls under positive objects in driving scenarios. Detecting all these obstacles in a class agnostic manner is important to avoid colliding with them. One major advantage of detecting obstacles in a class agnostic manner is to identify objects that are a potential threat to the vehicle, but does not belong to a common object class. For an example, some logs on the roads may not be considered as a common object class in a driving scenario, but can have a severe threat to the safety of the vehicle and passengers inside.

Research in obstacle detection are summarized as monocular camera based and stereo vision based approaches, by the reviews presented in [3, 4]. In monocular camera approach, either appearance or motion features are used. This constrains obstacle detection task to detecting obstacles belonging to specific object classes or dynamic obstacles [3, 4]. Research that detect obstacles in a class agnostic manner, mostly rely on stereo vision. Stereo vision based methods makes use of the 3D information calculated from the two views when detecting obstacles. When there is a binocular view of the same scene, stereo vision can be applied to generate a disparity map, using the stereo matching methods discussed under Appendix A. Then 3D information can be calculated from the disparity map and stereo camera calibration results. The advantage of a stereo camera system is that it can make use of any obstacle detection method discussed earlier, where the only constraint is the computational complexity (hence the latency) to obtain the result. Stereo vision based methods can be divided into three main headings, namely (1) road surface estimation based methods (2) disparity flow and optical flow combined methods and (3) direct clustering of 3D points.

2.1.1.1 Road surface estimation based methods

Clustering 3D points (that can be calculated from stereo vision), can provide a meaningful segmentation of obstacles. In almost all of the work associated with road scenes and stereo systems, researchers try to estimate the ground or the road surface and then go on to finding the obstacles on the road surface. Road surface approximation is done based on the v-disparity image which was first introduced by [6]. Firstly, the v-disparity map construction is explained as below. Here $\mathbf{v}$ is the v in the (u, v) image co-ordinate system. When the disparity map is obtained from stereo matching, every pixel is assigned with a disparity value. This disparity value is in the range of $(0, d_{max})$ and the disparity image corresponds to the pixels in the rectified image with horizontal epipolar lines. v-disparity is constructed by scanning along each row of the rectified image and incrementing each disparity bin by one, when a corresponding disparity value occur in that scan line. This process is repeated for all rows in the rectified image. This is like constructing a histogram of disparity for each rectified image row. Therefore the v-disparity map is a 2D map with dimension of width being equal to maximum disparity value d_{max} and height being equal to image height (number of rows in the image) v . This process can be visualized by looking at the Fig. 2.1.

Figure 2.1: v-disparity and u-disparity map generation. Top: u-disparity map, Left to right: Rectified left image, Disparity image, v-disparity map

Given a road scene and a horizontal stereo rig, majority of the pixels in a given image row belongs to the road at the same distance. Thus the majority of the disparity values along that line should be the disparity value corresponding to the road. Furthermore, this disparity value will be the most incremented disparity bin

in that line, which can be clearly identified in the v-disparity map in that line as the peak. Also note that the disparity value is inversely proportional to the depth. Therefore pixels corresponding to the near depths to the camera (usually the bottom part of an image) will get higher disparity values. Disparity value will decay when you go up the image, as a point higher up the image usually is at a higher depth. That is the reason, that v-disparity map is visible with a line as per the Fig. 2.1. This is the line that corresponds to the road or the ground surface. Along this road line, there are vertical lines in the v-disparity image, which corresponds to the obstacles on the road surface. Usually an obstacle will have same disparity values the area it spans in the image. So when the v-disparity is constructed, it is visible as a vertical line standing on top of the road line.

In a similar fashion u-disparity map can be constructed for the u axis of the image. u-disparity image will have the dimensions such that its width will be the number of columns in the image u and the height will be the maximum disparity value d_{max} . In contrast to v-disparity, u-disparity map is a collection of histograms of disparity values for each column. In u-disparity map, any feature about the road will not be visible as you are scanning along a column on the disparity image and the disparity values will be distributed from 0 to d_{max} . On the other hand since the obstacles have a similar disparity for some row and column area, it will give rise to certain peaks in the u-disparity. Therefore, peaks in u-disparity will corresponds to the disparity values belonging to an obstacle in that column.

In order to identify the road, dominant line in the v-disparity map should be identified. In [7], authors use Hough Transform to detect the road surface from the v-disparity map. For planar roads, road line in the v-disparity map can be successfully approximated by a straight line. However, for non-planar road conditions, road line in the v-disparity map is a curve. In [7], non-planar road approximation is done using piece-wise linear curve fitting using C highest Hough Transform values. Once the road profile is extracted obstacles are detected by scanning each line in disparity image. For a given scan line, if the disparity value corresponding to the road is $d_{u,road}$, any other disparity value along that scan line is marked as a potential obstacle. This is an example for class-independent obstacle detection. Authors in [8] also uses Hough Transform to detect planar road surface

from v-disparity. Then they use u-disparity to detect disparity ranges for different pedestrians and use that information with road line to calculate bounding boxes for pedestrians. Later on these human hypotheses inside bounding boxes are verified with a SVM classifier trained on HOG features.

In [9], authors present a non-parametric way of identifying the road profile. The method they have proposed is fast and has the ability to detect planar and non-planar roads using the same methodology. Authors first compute the u-disparity map from the disparity image and identify obvious obstacles from u-disparity peaks. Then the disparity values corresponding to these large obstacles are removed from the disparity map, which is then used to construct the v-disparity map. Then using this v-disparity map authors identify the road profile in a non-parametric manner by scanning the v-disparity map row by row, and detecting the peak disparity value at each line. They make use of an additional constraint from the observation that the road regions in the near vicinity will be located in the lower part of the image, to identify the initial road point. Hence, the peak disparity value on the scan line that is on top of the previous scan line, should be less than or equal to the peak disparity value corresponding to the previous scan line, as the disparity value is inversely proportional to the depth. Developing on top of this road detection method, same authors have presented a method to detect and track obstacles in [10]. In this paper, authors first remove the road surface and anything below the road are initially considered as obstacles. Then they make use of the u-disparity map to categorize these points into different obstacles. This is done by identifying the peaks regions from the u-disparity map using an adaptive connected component labeling method. Each of these regions corresponds to a positive obstacle in the scene.

In addition to detecting the road surface based on u-disparity and v-disparity method, some authors try to model the road surface. In [11], authors follow a similar method, where the road surface is modeled using a quadratic function based on the rectangular Digital Elevation Map (DEM) cells generated from the 3D information from the stereo images. One height is recorded for each DEM cell. DEM cells are classified either as road DEM cell or obstacle DEM cell, by the fact that obstacle DEM cells having a much larger 3D points density than that of a

road DEM cell. In addition, there are several deep learning based methods for segmenting road scenes as presented in [12] and [13]. Whilst deep learning based methods have higher accuracy, real time performance is yet to be achieved.

2.1.1.2 Disparity flow and Optical flow combined methods

There are number of work done to mainly detect dynamic obstacles by combining disparity information and optical flow information. Authors in [14] present an idea of detecting dynamic and static obstacles which they term as 6D vision. Their idea is that using the information from stereo images in more than two consecutive frames to calculate (X, Y, Z) 3D positions and $(\dot{X}, \dot{Y}, \dot{Z})$ 3D motion information and cluster these 6D information to find out static and dynamic obstacles. Authors in [15] finds dynamic obstacles using stereo input and calculating sparse feature based disparity flow and optical flow in subsequent images. Then the feature points that has similar optical and disparity flow are clustered together to generate class independent dynamic objects. In [16], authors make use of disparity flow and stereo flow to calculate ego motion and then find the dynamic obstacles by compensating the ego motion.

2.1.1.3 Direct clustering of 3D points

In some works, researchers try to detect the obstacles only by using the 3D information presented from the disparity map calculation. In [17], authors use a blob extraction method at each disparity level and then use a SVM classifier on those blobs to classify them into several obstacle categories. This is based on the assumption that objects can be represented as surfaces in the real world and can be distinguished from the background of the scene.

Success of all stereo matching based methods depends on the accuracy of the disparity map and the accuracy of the camera parameters obtained from the calibration process. In the review by [4], authors mentioned number of papers that update extrinsic camera parameters online because there can be relative changes occurring in the stereo camera rig due to vehicle vibrations and windy conditions.

2.1.2 Negative Obstacle Detection

Negative obstacle detection is not studied to the extent of positive obstacles, due to lack of such obstacles in normal life and the unavailability of benchmark data sets to work with. Negative obstacle detection has mainly been studied with data from off-road robot navigation. Literature on off-road negative slope or negative obstacle detection can be organized in multiple ways. In [18], authors provide a review on potholes detection methods by categorizing them into (a) vibration based methods, (b) 3D reconstruction based methods and (c) image processing based methods. However, in addition to these three categories, (d) thermal image based methods and (e) multiple sensor fusion methods are used in pothole detection in literature. Therefore, in this study the literature will be reviewed in these five categories.

2.1.2.1 Vibration based methods

Vibration based methods are mainly used to identify the potholes present in roads for the purpose of reporting them for maintenance. Therefore, they detect potholes when the vehicle is already on one and thus are not useful in avoiding such obstacles. In [19], authors identify locations that may have potholes based on accelerometer statistics. Ultrasonic sensor measurements are used to detect potholes in [20] based on the depth to road and pothole. In [21], authors train a classifier using accelerometer and gyroscope data to classify road as smooth or potholed.

2.1.2.2 Thermal image based methods

Thermal imaging is used to detect negative obstacles as they remain warm compared to the surrounding, especially at night. In [22], authors use thermal images to find such warm places to detect holes while verifying it with missing stereo data. Authors in [23] use the same thermal imaging approach and use colour to only consider potholes in the ground. Thermal images are used to train a Convolutional neural network (CNN) in [24], to classify which thermal print is potholed or not.

2.1.2.3 3D reconstruction based methods

Notable amount of research is done following 3D reconstruction based negative obstacle detection. These methods can further be categorized into (b.1) laser based and (b.2) stereo vision based methods. Laser points are used to find the difference in the elevation between neighbouring points to detect bumps and elevated pedestrian crossings [25]. Authors in [26, 27] use the aerial view in contrast to the normally used horizontal view and calculate height to the ground, to differentiate between ground and potholes. This method is applicable mostly to aerial vehicles, as the ground vehicles depend on the horizontal view of the road. In [2], positive and negative obstacles are differentiated using LIDAR, by calculating a height-length-density measure. Authors in [28] use the same sensor and detect potholes and holes, by detecting a transition to occluded 3D voxels after few visible voxles. Support vector machine (SVM) trained on 3D LIDAR points has also been proposed for negative obstacle detection in [29]. Authors in [30] use laser images to detect deformed laser pattern to detect potholes and subsequently train a neural network to classify the severity of them.

Methods that use stereo vision based 3D information to detect potholes are categorized as stereo vision based methods. In [31], authors utilize height difference from adjacent grid cells to define negative regions. Since the grid map is based on stereo information, far range detection is difficult due to the sparse sensor data. Authors in [32] use 3D information from stereo vision to predict the the road surface as a quadratic function and detect potholes as the points that are lower than the road surface. This method has been enhanced in [33], by proposing a data sampling method to increase the quality and reduce the computational complexity of road surface fitting to make it suitable for real time applications. Additionally tracking it over frames has helped authors to reduce false positives. In [34], road surface fitting method in the disparity domain is proposed, without projecting the points to 3D space and mark the points below the road surface as pothole candidates. Thresholding, morphological opening and connected component analysis are used as post processing steps to remove noise from these candidates and to arrive at the final negative obstacle points. Later, authors improve this method in [35], by integrating multiple frames to build a digital elevation map (DEM), compensating

for ego motion. The holes (no data cells) in the time integrated DEM is filled using bilinear interpolation and then local minima are located. Region growing method is applied on the local minima to arrive at the final pothole regions.

2.1.2.4 Image processing based methods

Literature that use 2D image information from monocular cameras are categorized as image processing based methods. Authors in [36], propose an image processing based method to detect potholes on roads under several assumptions such as potholes are darker than the surrounding area, most potholes being elliptical and texture inside a pothole being much coarser and grainier than the texture in the surrounding area. In [37], potholes are detected following three main steps. Firstly the image (only contain road and potholes) is segmented based on histogram thresholding, which is then subjected to series of filtering and thresholding operations to improve the segmentation to a candidate. This candidate and background is compared to make the final decision as to classify it as a pothole or not. In [38], authors select a region of interest (RoI) on road using vanishing point and generate candidates using a histogram threshold. These candidates are filtered by a set of hierarchical decisions based on candidate size, shape, intensity and contour. If these filtered candidates are found in more than one consecutive frames they are categorized as potholes or cracks. In [39], histogram based thresholding on RoI, followed by a shape based false positive elimination method is proposed. RoI is defined by processing the intensity values in RGB space. Three linear binary pattern (LBP) based cascade classifiers are used in [40] to categorize images with negative obstacles in to potholes, fatigue cracks and longitudinal cracks. In [41], least squares SVM (LS-SVM) and neural network (NN) based classifiers are compared using features extracted from integral projection and steerable filter outputs of 200 images, for the purpose of classifying images with potholes. LS-SVM has performed better than NN method. LS-SVM trained (on 1.000 images) on image texture features is used in [42], for classifying images with potholes. In [43], SVM is trained on bag of words (using scale-invariant feature transform - SIFT) to classify images with potholes, which is then subjected to a graph based clustering and intensity based thresholding to localize the potholes.

2.1.2.5 Multiple sensor fusion methods

Authors in [44] use multiple sensor data and train several classifiers on 3D information and texture to find negative regions in the scene. Authors make use of visual images, infrared images and laser. In [45], radar patterns are analyzed to detect presence of potholes, which is then combined with image data to accurately segment them. In [46], authors train a deep NN to semantically segment the potholes using 6000 test images. During test stage, pothole regions detected by the network is verified comparing the depth (from laser point cloud) of the road and the pothole.

Most of the methods in literature (vibration based, image processing based and thermal imaging based) are developed to identify the location of negative obstacles for road maintenance purposes. Thus they mainly focus on classification while some localize them in 2D image space. However, it is important to avoid such negative obstacles in autonomous navigation applications, such as autonomous cleaning robots (sidewalks, pavement and road cleaning). Therefore, the main focus in the research is on 3D reconstruction based methods using stereo images.

2.1.3 Object Detection

Object detection aims to identify an object or multiple objects on a given image and to localize each object as to where they appear in the image. Localizing object means to find out where the object lies in the image, which can either be identifying the bounding box coordinates that cover the object or pixel-wise segmentation of the object. There are two types of pixel-wise segmentation methods in literature, which are (a) Instance Segmentation and (b) Semantic Segmentation. Semantic segmentation is the task of assigning each pixel in the image with a class label. Whereas, in ‘Instance Segmentation’, in addition to assigning an object class, instance identity is also attached to every pixel. Therefore, two objects (instances) of the same class can be separately identified using ‘Instance Segmentation’. Hence, semantic segmentation is more popular in the segmentation field, while instance segmentation is more popular in object detection field. In object detection, each object is classified in to a pre-defined multiple classes (e.g. car class, pedestrian class and etc.) along with localization information.

Object detection related literature is reviewed under three categories, namely (a) single class object detector, (b) objectness score based single detector for multiple object classes and (c) deep learning based single detector for multiple object classes.

2.1.3.1 Single class object detector

Early literature on object detection in road scenarios focuses on one object class (such as vehicle detection or pedestrian detection or pavement detection) at a time. Vehicles and pedestrians are the mostly studied object classes in the early days of research because of their greater importance to advanced driver assistant systems (ADAS) . Authors in [47] and [4] provide a good review on existing vision based vehicle detection methods while [5] and [48] review methods for pedestrian detection.

Almost all the work in this category follows a proposal based object detection pipeline. Authors in [4] propose a framework to categorize all vehicle detection algorithms as a two step process, (1) hypothesis generation and (2) hypothesis validation. They then categorize hypothesis generation step in to three methodologies as, (1) knowledge based (2) stereo based and (3) motion based. Then the hypothesis validation in to two methodologies as, (1) template based and (2) appearance based. It is convenient to use such a process when you know the object class, which in this case is vehicles. But for a general case where you need to detect all the possible obstacles, the above mentioned hypothesis validation step becomes a problem because it is no longer a simple binary classification such as vehicle vs. non-vehicle. However, hypothesis generation methodologies are a very good way to analyze different object detection strategies in literature. Thus, literature is analyzed in a similar way, where instead of knowledge based it is termed as feature based methods.

In [47], authors categorize all vehicle detection methods to two sections as (1) monocular and (2) stereo. Under each section the methods are further categorized as (1) appearance based and (2) motion based. In [5], authors review the literature on pedestrian detection using a pipeline structure with six steps, where similar to [4], they also use hypothesis generation and validation steps which are termed as

object classification and verification/refinement. In [48], authors only review algorithms that use individual monocular images for pedestrian detection in a sliding window approach. Therefore these algorithms fall under feature based methods.

Considering above factors, in this study detection methods are discussed under two headings, (a) feature based methods and (b) motion based methods.

Feature based methods are the methods that make use of features to detect objects. These methods are usually followed by a verification/classification step. Intensity information, Colour information, Edge information, Texture descriptors, Histogram of Gradients (HG), Histogram of Oriented Gradients (HOG) Haar-like features, Shape descriptors, Symmetry, Corners, SIFT, SURF are examples for common features used in the literature explained in the reviews done by [47, 4, 5, 48]. Out of these features, according to the review done by [48], HOG is the best feature for pedestrian classification, when considering single features. A combination of several features provide better results than using a single feature. Most of the algorithms that falls under this category perform a binary classification in the hypotheses verification step. Most common classifiers used in the literature are SVM and AdaBoost classifier.

Motion based methods are based on estimating the motion for every pixel in the image and categorising them accordingly. According to the review done by [47], motion based methods are more commonly used in stereo camera systems than in monocular camera systems, when it comes to vehicle detection. Calculation of optical flow is computationally expensive, hence in some literature sparse optical flow is calculated for specific corner like features as mentioned in [4]. While most of the pedestrian detection algorithms are based on feature based sliding window type methods, one work that integrate features and motion estimation (using optical flow) has given the best results in the analysis done by [48].

Analysis in [47] suggests that most of the work with regard to motion based vehicle detection involves Optical Flow estimation and some other work involves adaptive background modelling techniques. These background modelling techniques work well for stationary camera. However, the background is not static for a moving camera. Hence the background modeling is mostly used in traffic flow

estimation scenarios.

Optical flow is the apparent motion of the pixels between two consecutive frames due to motion in either the camera or the scene. It is a 2D vector for images. Optical flow is developed based on the brightness consistency assumption. That is, when a pixel $p(u, v)$ at time t having a brightness of $I(u, v, t)$ is moved by du small distance in u direction and dv small distance in v direction between the two frames with a small time difference of dt , brightness value $I(u, v, t)$ and brightness value of the pixel $I(u + du, v + dv, t + dt)$ are equal. Refer to (2.1).

$$I(u, v, t) = I(u + du, v + dv, t + dt) \quad (2.1)$$

When considering the first order Taylor Expansion on (2.1), (2.2) can be obtained.

$$I(u, v, t) = I(u, v, t) + \frac{\partial I}{\partial u} du + \frac{\partial I}{\partial v} dv + \frac{\partial I}{\partial t} dt \quad (2.2)$$

By dividing (2.2) using dt , (2.3) can be obtained which is known as the optical flow equation.

$$I_u \dot{u} + I_v \dot{v} + I_t = 0 \quad (2.3)$$

where $I_u = \frac{\partial I}{\partial u}$, $I_v = \frac{\partial I}{\partial v}$ and $I_t = \frac{\partial I}{\partial t}$ are spatiotemporal brightness derivatives of the image pixel. $\dot{u} = \frac{du}{dt}$ and $\dot{v} = \frac{dv}{dt}$ are the motion of the pixel. In optical flow, intention is to find the motion vector $\vec{p} = (\dot{u}, \dot{v})$. Therefore the problem with (2.3) is that there are two unknowns $\dot{u}, \dot{v}$ and one equation. In order to address this problem there are number of solutions presented in the literature which have been evaluated in [49]. The famous Lucas-Kanade method addresses this challenge considering neighbourhood consistency. Assumption made here is that the neighbouring pixels will also have the same motion as with the pixel in consideration. Neighbourhood window considered in Lucas-Kanade method is a 3×3 . This will result in nine equations from the (2.3) to solve for unknown $\dot{u}$ and $\dot{v}$. So a least squares fitting method can be used to obtain a better solution. Another famous method is the global energy minimization method called Horn-Schunck method where the assumption is that optic flow is smooth across the image and tries to minimize an energy function with data term as in (2.3) and a smoothness term over a neighbourhood.

In [50], authors perform sparse optical flow to calculate the ego motion of the vehicle. Then they use outlier rejection criterion to refine the ego-motion and use this flow to find the feature points that are not consistent with the ego-motion and cluster those points to obstacles. [51] and [52] are some recent work on monocular based optical flow calculation methods. Both of these algorithms follow a global energy minimization approach.

Motion based methods have a major disadvantage because it cannot be used to detect static obstacles as there will not be any difference in the motion parameters. Another point to note is that generating a dense optical flow is very time consuming and can affect the real time performance. Even though it is time consuming, generating a reliable optical flow from a moving camera is not an easy task. Optical flow is also susceptible to errors when the relative difference between two adjacent frames are too much than the motion vector search range.

2.1.3.2 Objectness based single detector for multiple object classes

In the recent years class independent object proposal detectors has been developed. Most common approach in the past was to use a sliding window approach and apply a classifier trained on multiple classes to each sliding window at different scales and try to classify, whether the sliding window contains any object in it. This method still has the problem of only being able to detect pre-defined multiple object classes. Method is also very time consuming because of the need to run a computationally expensive classifier on large number of sliding windows per image. These problems were addressed by [53], using a generic objectness measure instead of a time consuming classifier. However, this method takes around four seconds per image, but it is very efficient over the existing methods that uses a classifier. Authors in [53] use some properties of an object, such as a well defined boundary, difference in appearance from the background and the salient nature of an object in a window, to define an objectness measure. Inspired by this, recent work has achieved much faster algorithms such as, [54] and [55]. In [54] authors use an edge based objectness measure to detect objects in an image and typically requires about 250 ms per image. In [55], authors present a method to identify the presence of an object in a given window. This is achieved by re-sizing that window to a small fixed

size window and looking at the norm of the gradients in that small window. Authors in [55] name this method as Binarized Normed Gradients (BING) and claims to be working at very high speeds of 300 frames per second. Authors in [56] introduced a method called selective search, to generate class independent bounding boxes by filtering put unnecessary boxes, using different strategies similar to region growing segmentation tasks.

2.1.3.3 Deep learning based single detector for multiple object classes

Almost all the state-of-the-art object detection methods are based on deep learning networks. These networks have the ability to localize the position of the object in the image and to predict the probability of that object belonging to a particular object class. In the recent past, object detection research has been able to achieve very good results owing to these methods. All deep learning based methods were truly made possible with the large amount of labeled training data and specialized hardware (GPUs) with large scale parallel processing capabilities. Literature related to deep learning based object detection can be categorized as (a) Two stage object detectors, (b) Single stage object detectors, (c) Heat map based object detectors and (d) semantic segmentation.

Two stage object detectors first try to identify bounding boxes containing an object and then classify each of them in to different classes, while regressing the bounding box. As the first step, it is required to generate object proposals, where a separate object detector (this can be an objectness based single object detector) is used to generate these object proposals. Then a separate CNN is used on each image patch to extract deep features and train a SVM (Support Vector Machine) to classify them in to different classes. This method was called R-CNN [57]. Later, this is stream lined to use a CNN once to generate features and used concept call RoI Pooling to generate a fixed number of features for differently sized regions of interest. Furthermore, a fully connected neural network was used to replace the SVM classifier along with a bounding box regression head to fine tune the region proposals further. This method is called as Fast R-CNN [58]. However, these two methods rely on object proposals generated from a traditional

object detector in the first step. Because of this, the speed of the overall process was adversely affected. Then Faster R-CNN [59] was introduced to replace the traditional object detector with a regional proposal network, integrating all the steps in to one network. Later, Mask R-CNN [60] introduced RoI Align method to replace RoI Pooling to generate much better detection results. Furthermore, Mask R-CNN introduced a segmentation head to generate a mask for each object in the bounding box. This concept is known as instance segmentation.

In general, two stage object detectors generate accurate results compared to other methods. Faster R-CNN [59] generated the best detection results in COCO [61] 2015 detection challenge while Mask-RCNN [60] outperformed all the methods in COCO [61] 2016 detection challenge.

Single stage object detectors try to solve the detection problem in a single stage without the need for an additional regional proposal network as in two stage detectors. Since the two stage object detectors could not achieve real-time performance, single stage object detectors emerged. Yolo [62] is one of the popular work among single stage detectors. In Yolo, image is divided in to a grid and for each grid cell set of anchor boxes are defined, which are associated a confidence score to check whether there's an object inside the bounding box. Also a classification score is also produced for each bounding box. Since an image only needs go through the network once to generate detection results which makes Yolo really fast. In SSD [63], bounding box classification confidence score calculation is done at multiple convolution layers (hence at multiple scales) using 3x3 convolution, generating better results than Yolo at a slower run time.

Heatmap based object detectors gets rid of the concept of anchor boxes and use keypoints based heatmaps instead. This is a most recent direction in object detection research. In a way this could be considered as single stage detector as well. CornerNet [64] was the most initial work in this area, where instead of anchor boxes, two corner points were used to generate heat maps to train the network. For each class there's separate heat map. CornerNet [64] produced very good detection results exceeding the performance of single stage detectors but at a slower frame rate. CornerNet-Lite [65] was introduced to improve the efficiency of

CornerNet [64] by changing the backbone and removing the need to process all the pixels. CornerNet-Lite [65] achieved higher frame rates than Yolo [62] at improved detection accuracy. CenterNet [66] was developed on top of CornerNet [64] where authors used a triplet of points (two corner points and center point) to improve the detection accuracy than the prior, but at a similar speed. In [67], authors present a much faster method which only uses object center points as the heatmap. The method proposed in [67] can achieve higher running speeds and better accuracy than CornerNet-Lite [65] and Yolo [62].

Semantic segmentation based methods tries to segment every pixel in the image to a object category. Semantic segmentation methods are class dependent categorizations implemented based on supervised learning. Usually semantic segmentation follows a encoder-decoder based architecture in the network. Encoder is where you down sample the image and select features and produces the labelling. Up sampling that low resolution representation is the decoder part. Relatively, the important task in this architecture is the decoder as segmentation should adhere to object boundaries correctly. [68, 69] are good examples for deep network based semantic segmentation implementations.

2.2 Object Tracking

Tracking is a challenging problem in many video analysis tasks, where an object defined by a bounding box needs to be identified and maintain a unique identity over all the frames it appears in an image sequence. Tracking research can broadly be divided in to Multiple Object Tracking (MOT) and Single Object Tracking. While MOT assumes object detections as prior knowledge, the later tries to localize and track an unknown object which is only described by the localization information at the first frame. Single object tracking is often known as detection-free tracking or visual object tracking. Video object detection is another aspect of multiple object tracking, however it's normally constrained to two or three objects per frame.

In both single object tracking and multiple object tracking problem, researchers use hand crafted features and filters, or deeply learnt features and filters or a combination of hand crafted and deeply learnt methods.

2.2.1 Single Object Tracking

In recent years, the most prevalent technique in single object tracking area is *discriminative method*, opposed to generative methods. Instead of building an object appearance model based on generative process and without considering the background [70], discriminative trackers are able to distinguish the target from negative samples by learning a classifier, which is more accurate [71]. TLD (Tracking, Learning and Detecting) tracker [72] divides tracking process into three stages (tracking, learning and detecting). Correlation filter (CF) based tracking [73] and kernelized correlation filter (KCF) based tracking [74] can be presented as significant achievements. With the recent breakthrough in deep learning domain, many deeply learned single object trackers were developed [75].

SOT research in recent years follows two main architectures, (a) correlation filter based variants (CF) and (b) Siamese network [76] architecture based methods. With the introduction of MOSSE tracker [73], CF based tracking has attracted much attention and still most of the best methods use CFs in their implementations [77]. In [77], authors use a discriminative CF based on convolution feature activations for tracking problem. CF based methods use peak correlation output to find the best matching location and often use fast fourier transform to increase efficiency by working in the frequency domain. CF is updated online to capture changes in the tracked target and thus is comparatively more time consuming than Siamese based methods. On the other hand, since the deep learning methods are trained offline, it can achieve very high speeds [75]. Among deep learning based methods, SiamFC [78] follows Siamese architecture and achieves very good accuracy at high speeds. In such methods, search region and target are both passed through identical networks (Siamese) to generate features vectors which are then used to learn a function, to find the similarity. SiamRPN [79] is an improvement to the above, which incorporates region proposal network which can achieve very high speeds and generate state-of-the-art results.

2.2.2 Multiple Object Tracking

Multiple Object Tracking (MOT) is an important sub task for better understanding of the road scenarios. MOT is useful for number of applications such as traffic

analysis, surveillance, driver assistance systems as well as automated driving platforms. The proposed work tries to address the MOT problem for automated driving applications. MOT problem is solved using (a) offline or (b) online approaches [80]. While the offline (also known as the global) approaches process the whole image sequence at once, the online approaches address the tracking problem at each local frame. Non-real time applications favour global approach as they provide the highest amount of information as the whole image sequence is available as prior knowledge. On the contrary, online methods are favoured in real time applications such as driving scenarios.

The most prominent approach for online MOT problem is the tracking by detection approach where the tracking framework is provided with multiple object detection results in each frame as a prior. Therefore, all tracking by detection approaches have two main stages, namely (a) detection stage and (b) track association stage. Unlike single object tracking, in MOT new objects can appear (object birth) in the middle of the image sequence (not necessarily in the first frame) and some objects can go out from the frame (object death) after certain number of images. Therefore, it is necessary to use an inherent or exclusive object detection mechanism to detect objects in each frame. In track association stage, detection results from the object detector are assigned to tracks.

Alternatively, in some research work [81, 82, 83, 84] multiple objects initialized at the first frame are tracked across the sequence. This is similar to multiple single object trackers where the object localization is done by searching for the best matching bounding box in a search region. Main challenge in such scenarios is the tracker drifting away from the actual object with time. In such situations object bounding box information from a object detector is used to correct the drift. Just using a tracker can be fast depending on the number of single object trackers initiated (number of objects to track). However, this can be addressed using parallel processing using GPU as well. Since the tracker is running asynchronous to the detector, high performing (yet time consuming) object detector can be used in between several frames to correct the drifting [85]. These methods can be seen as extensions to single object tracking problem.

2.2.2.1 Offline Methods

Some MOT methods are designed for specific object classes, such as multiple human tracking [86, 87, 88] while other methods [89, 90, 91, 92] are class agnostic. Most MOT methods follow tracking-by-detection framework where they rely on an object detector to provide object candidates. Some methods that follow tracking-by-detection framework make use of all the object candidates in the sequence while the others use candidates up to that frame. Former methods are classified as global while the later as online methods. Network flow optimization [93, 94, 87], graph based clustering [95, 86], multiple hypotheses tracking (MHT) [96] and Bayesian filtering based tracking [90] are among the popular approaches in global MOT methods in the recent past. Other than the direct association of detections to tracks, some global MOT methods [90] use detections to tracklets (which are a combination of matching detections in few consecutive frames) and tracklets to track assignment to address long term variations in tracked objects. In [97], Siamese Long short-term memory (LSTM) based network is proposed to provide a similarity score between two trajectories (i.e. tracklets) of constant lengths. Appearance features are extracted from network trained for person re-identification, which are then combined with motion features using a fully connected layer and feed in to a LSTM for temporal integration. Tracklets are generated using Kalman filter prediction and bounding box intersection over union based Hungarian assignment in an online manner.

When calculating the matching cost or dissimilarity between detections in different time frames, object’s appearance and motion are the most common information sources. In appearance based distance calculations, traditional methods such as distance between RGB colour histograms [90] or distance between deeply learned features [86, 87, 95, 96] are used. Point motion matching [90], distance between expected and detected positions [86, 96] and spatio-temporal distance between object bounding boxes [95, 87] are used to define motion based matching costs. In addition, authors in [87] used deeply learned person re-identification to address long term occlusions. As a post processing step, authors in [90] use a change point detection framework to rectify errors in tracklets to track assignment. Some methods [86] use multiple detectors to improve the tracking results, at the

expense of increased computational time.

2.2.2.2 Online Methods

While offline MOT methods mostly use some form of global optimization to associate objects or tracklets to tracks using matching costs from all the frames, online MOT methods operate on information up to current frame. Thus online methods are more suitable for real time applications. Online MOT methods mostly use pair-wise cost to match the detections to tracks.

Similar to global methods, matching cost calculation is based on appearance and object localization information from the detections. Object localization has been used in defining motion based models as well. Authors have used deeply learnt features as well as hand crafted features when calculating the appearance based features. Authors in [98], use RGB histogram of the object as the appearance feature and calculate the similarity as the Chi-Square distance between histograms. In addition, authors also make use of the cosine distance of the spatio-temporal locations. In [99], similarity between a track and object is calculated based on the difference between forward and backward optical flow. Authors in [91] use deeply learnt keypoints as the appearance feature. Furthermore, they make use of the object shape, pose, 2D and 3D localization information in finding the similarity between a track and an object. In [100], authors use deeply learned person re-identification scores and object localization information in a Recurrent Autoregressive Network to calculate pair-wise costs. Authors in [101] use a trained siamese network for historical appearance based matching which specifically help in reducing identity switches when tracking. Additionally motion and size based matching has been used to aid the tracking task. Authors in [102] use discriminative appearance learning method for each track based on the positive samples of the detection and negative samples around it as the appearance model. They further make use of object size and position based spatio-temporal matching and combine these measures in a multiplicative manner.

Triplet loss based convolution neural network is trained to extract appearance features in [103]. Additionally they also used motion and size based similarities combined in a multiplicative manner to calculate the final similarity score to be

used in tracking. Cosine distance is used to calculate the appearance feature based distance. Triplet loss as expressed in (2.4) is used in many person re-identification work [104, 105] for training the weights. (2.4) tries to make the feature distance between the feature vectors with different identities $\min_{\forall fm_{id} \neq fn_{id}} d(fm, fn)$ to be at least β margin higher than the feature distance between the feature vectors with same identity $\min_{\forall fm_{id} = fn_{id}} D(fm, fn)$, where fm and fn are the feature vectors of two objects in two frames.

$$L_{tri} = \max(0, \min_{\forall fm_{id} = fn_{id}} D(fm, fn) - \max_{\forall fm_{id} \neq fn_{id}} d(fm, fn) + \beta) \quad (2.4)$$

Appearance features are extracted in [106] following a Siamese type network trained using triplet loss. But they use a mask to indicate pixels belonging to the object during training and testing, to mitigate errors arising from background and other objects when the object is partially occluded. Mask-RCNN[60] is used to generate the mask. In [107], deep features are extracted from a network trained for human re-identification task as well. However, feature based distance is calculated not only considering the most immediate feature vector but up to 100 previous feature vectors and taking the minimum distance from that. Furthermore, distance between location and predicted location (based on Kalman Filter) is also combined to the pair-wise cost. A Recurrent auto-regressive network (RAN) is proposed in [100] to model the appearance and motion dynamics of each object. When the detector fails, RAN will detect the object, reducing false negatives. For each track RAN is used with shared weights.

It is helpful to understand the interactions between objects when performing the tracking task. In [88], authors train a Recurrent neural network (RNN) to calculate the social interaction model based similarity score, along with two other RNNs to calculate similarity scores based on appearance and motion. Then these three similarity scores are combined using a fully connected network to calculate a final matching score. In [108], authors use a grouping strategy under the observation that people walk in groups in affinity matrix processing. During affinity matrix construction (using pair-wise costs), size based distance, motion based distance and appearance based distances are combined. Appearance features are extracted from the higher layer convolution features from the deep network used for detection

task.

In order to improve the tracking accuracy when following tracking-by-detection framework, one trick could be to improve the detection accuracy. Some work [109, 110] combine detection results from multiple detectors to achieve this. However, when considering the overall system this will lead to inefficiencies in terms of resources and running time. Authors in [111] propose to use the detections from both the object detector and bounding box predicted from each track’s history. The pair-wise cost is defined based on deeply learned person re-identification scores. Detections are filtered using non maximum suppression using a cost define based on detection confidence and track confidence.

In certain situations motion cues can give rise to false matching, such as when the camera is subjected to a sudden motion. This global motion can be compensated by using relative structural motion cues between object bounding boxes [92, 112].

Multiple instances of single object trackers have been employed in [85] for the MOT task. ECO-HC - a hand crafted variant of [113] using Histogram Of gradients (HoG) and Colour Names is used as the single object tracker. A dual network is used to calculate the pair-wise cost, for the tracks and objects that does not get matched by the above tracker. The first network is following a Siamese architecture that compares detection bounding boxes and track history. The output of the first network is used in the second network that follows a Long short-term memory (LSTM) architecture to calculate the matching score.

In addition to the pair-wise cost based online methods, filter update based methods are also used in online MOT problem. In [114], Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter is used to predict the object position and size based location measure, which is used with discriminative correlation based appearance matching cost. In a similar manner authors in [115] use Monte Carlo PHD filter where the appearance feature is in the form of dictionary matching, defined using RGB colour histogram and HoG clustering. Poisson multi-Bernoulli mixture filter is used in [116], where authors use predictions of object 3D co-ordinates, using 3D object co-ordinates learned from a deeply trained network. Extended Kalman Filter variant is used by [117] to track object location in

image coordinates, 3D world (using stereo matching and ego-motion calculation) co-ordinates and object size. In certain work [99], Markov Decision Process is used in online tracking framework to update the state (*active, track, lost and inactive*) of the track at each frame using an appearance based template matching. Template matching is based on dense optical flow matching. Authors in [89] propose an energy minimization framework for tracking where the energy terms are calculated using an appearance model and motion model (using FAST features trajectories measured using optical flow).

Most of the pair-wise methods [91, 88, 99, 98, 102] use Hungarian assignment [118], while some methods [100] use greedy assignment when finding the best match for detections and tracks. However, there is no significant advantage of one method over the other as they both will generate similar results [119], while Hungarian methods yields slightly better results. In [111] hierarchical association is used based on two different pair-wise matching costs.

Even though deep learning based MOT methods [88, 100, 101, 85] can yield accurate tracking results, these methods in generally are time consuming and require a sizable amount of training data in comparison to traditional hand crafted feature based methods. On the other hand, compared to these complex method, some [120] have shown comparable tracking results just by using the overlap between bounding box, as the matching cost in tracking. This has been improved in [121], where a single object tracker is introduced when no object detection is available. These results suggests that there's still much more room for improvement in the multiple object tracking area.

2.3 Concluding Remarks

In this chapter literature is reviewed mainly under obstacle detection and object tracking in line with vision based autonomous navigation. Based on the above review below conclusions are derived.

1. Positive obstacle detection is important for obstacle avoidance. In order to perceive the depth information from vision based system, a stereo camera approach should be considered. Most of the class agnostic positive obstacle detection methods needs to make use of 3D information from stereo vision.

In addition u-v disparity methods are helpful in clustering 3D information in a meaningful manner [6]. In addition to the ability to obtain 3D information, these methods are very cost effective and can achieve very high frame rates. On the other hand, all state-of-the-art object detection methods are based on deeply learnt approaches such as Faster-RCNN[122] and Mask-RCNN[60]. Particularly, Mask-RCNN[60] achieve very high level of accuracy and provide instance segmentation as well. However, the method is resource intensive. Single shot deep network based object detectors such as Yolo[62] can achieve real time performance, sacrificing the accuracy and instance segmentation. By combining such object detector with u-v disparity method can be helpful in obtaining instance segmentation in real time with 3D information on object points.

2. Almost all the methods following stereo vision based 3D information approaches make use of depth and height information to detect negative obstacle region. However, stereo vision based systems can extract visual information in addition to the 3D information in detecting negative obstacles. In [34, 35], authors uses simple image processing techniques as post processing step to improve the results. Literature on image processing based negative obstacle detection methods provide an overview of the methods that use visual information. Most image processing based methods assume that the negative obstacle region are darker than that of its surroundings [36, 37, 38, 43]. However, in some instances the above mentioned assumption is not valid. In such situations negative obstacle region can be observed as salient compared to the surroundings. This can be used to improve negative obstacle detection.

Traditional learning methods outperform NN based methods when training data is not sufficient [42]. In [46], deep learning based NN is used for semantic segmentation with 6,000 test images and output satisfactory results. However, they use a very powerful GPU and still require 250 ms per frame (frame size is not motioned but panoramic view is used). The main challenges for a good deep learning based method includes (a) the requirement for very large training dataset with ground truth, (b) the requirement of a powerful GPU and (c) high power consumption. Generating a sizable dataset

is costly. Furthermore, battery powered real-time robotic applications with limited hardware have difficulties in handling such computationally intensive methods. Hence traditional approach can be favourable in such scenarios. Hence, an iterative energy minimization strategy combining image color information together with 3D information is used in the proposed work for negative obstacle region detection. Details of this method is discussed in Chapter 4.

3. Even though deep learning based MOT methods [101, 85] can yield accurate tracking results, these methods in generally are time consuming and require a sizable amount of training data in comparison to traditional hand crafted feature based methods. Some deep learning methods [111] achieve accuracy as well as real time performance, but still require specialized hardware and consume much power to achieve higher running speed. Such methods cannot be used in processing power limited, battery powered embedded systems used in most robotic applications. Some filter update based methods [114, 115] achieve state-of-the-art accuracy without requiring specialized hardware, but their computational time is not high enough for real time applications. Appearance learning based methods [102] can achieve competitive performance in terms of accuracy at a higher computational cost. Thus, simpler methods are required for real time robotic applications that can achieve a good trade off between accuracy and real time performance. Therefore, a method combining multiple matching cost calculation methods that are hand crafted and computationally less expensive to calculate, is proposed.
4. Hand crafted feature based trackers add lower complexity on top of the tracking-by-detection framework. Overall MOT system that follows tracking-by-detection framework, is required to run an object detector initially to generate detections. All the state-of-the-art detectors are deep learning based, which outperform other traditional object detectors in terms of speed and accuracy. Hence, adding another deep network on top of such a framework for tracking increases complexity of the system as a whole. On the other hand, traditional tracking methods cannot match the accuracy of the exist-

ing deeply learnt re-identification methods. Furthermore, when looking at the multiple object tracking performance metrics of the current state-of-the-art methods in public benchmarks such as MOT16[123], still there is a larger room for improvement compared to detection. Therefore, MOT tracking problem can be approached as an extension to a detection network taking the advantage of deeply learnt features with small incremental complexity and run time.

Chapter 3

Detecting Obstacles for Real-time Navigation

3.1 Introduction

In order to safely navigate from one point to another, the robot should be aware of the obstacles present in the environment and the exact 3D locations of those obstacles. Pure vision based systems rely on stereo vision to perceive 3D information from the binocular view of the same scene. In this chapter, stereo vision based obstacles detection and segmentation framework is introduced that can run in real time.

When detecting obstacles for real-time navigation, obstacles that lie on the path of the robot carries an importance over the other obstacles. Therefore, understanding the drivable path is an essential step in obstacle detection. In addition to identifying an obstacle free driving path, with stereo information, it is possible to calculate the slope of that path. Knowing the condition of the drivable path is helpful in controlling the robot. For example, to accelerate the robot to an upward slope or to reduce the speed on a downward slope is required for the safe operation of the robot. Hence, understanding the drivable path, i.e. detecting the drivable path and calculating the slope, is presented as the first section in this chapter.

Once all the points on the drivable path is identified, the remaining are obstacle points. However, inaccuracies in disparity leads to incorrectly detecting certain areas on roads as obstacles. These can be rectified by segmenting obstacle points to meaningful objects. Therefore, class independent obstacle detection is presented as the second section.

These class independent obstacles belong to different objects. It is important to categorize these obstacles into objects as some object classes such as pedestrians, vehicles, traffic signs etc. carries additional information for a robot, than some classes such as trees, buildings etc. To this end, an object detector is used to classify and detect object in the image. While classification is the task of identifying the object class, detection is the task of localizing (identifying the bounding box of the

object in the image plane) each object. This localization result (bounding box) contains the object as well as the background. Instance segmentation is one step advanced than object detection, which is the task of assigning an instance identity and class label for every pixel in the image. Using both class independent obstacle points and object detection output, instance segmentation can be achieved, which is presented as the third section in this chapter.

3.2 The Proposed Method

In this section the methods are discussed in details. In all of the methods, real time performance is achieved to support navigational applications.

3.2.1 Understanding the Drivable Path

The main objectives of understanding the drivable path is to detect the road and identify the slope of the road. Road extraction is the process of identifying pixels belonging to the road or ground plane. Furthermore, since the proposed method is based on stereo vision, corresponding disparity value is identified for each road row in the image. Road starting row and ending row are identified as well. Road profile is identified using v-disparity and u-disparity maps as per the method proposed in [9] by Wu, Meiqing, et. al. Main reasons for using the proposed method are; (a) simple implementation, thus can produce real time results and (b) ability to provide accurate road approximations even for non-planar road surfaces.

Firstly, based on the disparity map u-disparity map is generated. Since u-disparity map is like a collection of histograms of the disparity for each image column, the peaks in the u-disparity map corresponds to the obstacles. So the u-disparity map is then applied to thresholding to identify the peak and thus the disparity values corresponding to the obstacles. This step is there to help with road profile extraction by removing large obstacles which will otherwise lead to erroneous results in road profile identification with v-disparity. Either a constant or a disparity dependant constant can be used as the threshold to identify peaks. Later is the better choice as objects that are far away corresponds to smaller number of pixels in the image. Thus, threshold values are pre-calculated based on the equation below and stored in a look-up table for faster referencing during

inference time.

$$u_{th}(d) = \frac{h_{min} \times d}{b} \quad (3.1)$$

where b is the baseline, d is disparity and h_{min} is the minimum height of an obstacle to consider it as a large obstacle. Minimum height is a parameter to distinguish large obstacles, thus it is defined as 0.4 m. This parameter is configurable and for the road conditions object with a height of 0.4 m can be assumed to be a large obstacle.

Figure 3.1: u-disparity based large obstacle removal. (a) u-disparity image, (b) disparity image and (c) large obstacles removed disparity image

Once the peaks in the u-disparity map are detected, the corresponding obstacle can be easily removed from the original disparity map. u-disparity based large obstacle removal process is as shown in Fig. 3.1. Then the v-disparity map is generated from this large obstacle removed disparity map. Road surface can be obtained with v-disparity. It is done based on several observations, (1) maximum intensity value in at least first few rows belongs to the road (i.e. bottom part of the image will mostly constitute from the road), (2) in adjacent rows, the disparity value of the road in the higher row (smaller row number) is always smaller than or equal to disparity value of the road in the lower row (higher row number). Road profile (rp) extraction is started from the bottom of the image. Based on the first assumption the detection process is initiated by identifying the highest intensity of the v-disparity map for the last few rows (R_{low}). Then road profile is approximated by traversing row by row, based on the second assumption. For this purpose, an

initial road profile ($init_rp$) is constructed by just selecting the disparity values corresponding to the peak intensity value for each row of the v-disparity map (d_r). Then if the disparity value of the peak intensity of the row in consideration is equal or lower than (within a threshold (v_disp_{th}) to avoid outliers) that is of the previous row, that disparity is added to the disparity road profile. If no match found, the same disparity as per previous row is considered. Then the process is stopped, once you observe that the road profile is not moving to the left for a threshold number of rows (R_{hrz} extraction). The process is summarized in Fig. 3.2.

Figure 3.2: Road profile extraction flow chart

Each row of the original disparity image is checked with the corresponding disparity value from the road profile. If the disparity $d_{r,c}$ is less than the corresponding disparity from the road disparity profile d_r^p plus a pre-defined small threshold value d_{th} , that pixel in the rectified image is assigned as a road pixel p_{road} otherwise an obstacle pixel p_{obs} . That is, pixel $p \in p_{road}$ if $d_{r,c} < (d_r^p + d_{th})$ or $p \in p_{obs}$ otherwise. Then for $p \in p_{road}$ for which $d_{r,c} > d_r^p$, p is checked whether it was identified as a large obstacle in the u-disparity, if yes p is re-assigned to $p \in p_{obs}$. Small threshold d_{th} is required because there are some road pixels that have higher disparity value than the median road disparity value. These pixels $p \in p_{obs}$ are used in class agnostic obstacle detection. v-disparity based large obstacle removal process is as shown in Fig. 3.3.

Figure 3.3: Road profile extraction process. Left to right: Initial v-disparity, Large obstacles removed v-disparity, Peak points of the v-disparity (in green), Road profile (in green)

Once the road profile is extracted, slope of road plane can be easily estimated with the 3D information. Slope can be calculated in two ways, (a) dense slope calculation - calculating the slope angle with respect to the current position of the ego-vehicle to all road points or (b) calculating slope for different road segments. Since the initial method is too computation intensive and has no real use, later is used. Slope is calculated for three road segments with relative to the vehicle

Figure 3.4: Slope calculation flow chart

position. Road is segmented based on the depth, where for different camera configurations different values are selected. During testing 5m, 8m and 10m have been used. Number of road segments can be changed by changing this depth difference and there is no requirement for a certain number of road segments, for the accuracy of the method. Currently, When the road is not visible (when the view is covered by a large vehicle) slope is not calculated. Road not visible image rows are identified when the v-disparity value for that road row has lower value than a threshold. Based on the road height, slope is calculated following basic trigonometry. Slope calculation is summarized in Fig. (3.4) and Fig. (3.5). Slope angle corresponds to the tangential angle between the height difference of the road point to the road height of the current position and depth of the point.

Furthermore, for each road segment road pitch can be estimated under the assumption that roll and yaw for that road segment is negligible. For this purpose, for each road segment, surface normal to the road is calculated. Surface normal can be calculated as the cross product of two vectors that lie on the road plane.

Figure 3.5: Slope angle and pitch angle calculation in different road segments

Therefore, three points for each road segment, are randomly selected that are not collinear. Then the cross product between the two vectors for the three points are used to calculate the surface normal. Three points for three parts of the road segments, are randomly selected to avoid collinearity on one go. However, sometimes two or more iterations are required to avoid points that are collinear. Surface normal N_s calculation is as in Fig. 3.6.

Plane equation is as;

$$Ax + By + Cz + D = 0 \quad (3.2)$$

Surface normal is calculated as;

$$\begin{aligned} N_s &= \langle A, B, C \rangle \\ &= \vec{OP} \times \vec{OQ} \\ &= \langle (i - a), (j - b), (k - c) \rangle \times \langle (i - d), (j - e), (k - f) \rangle \end{aligned} \quad (3.3)$$

Figure 3.6: Surface normal estimation from three points on a plane. O, P and Q are points on the plane and N is the normal vector to that plane.

Once the surface normal is calculated, pitch angle of that road segment is calculated by taking the angle difference between surface normal and n vector

towards the perfect upward direction. Result of such pitch angle is shown in the right side of the Fig. (3.5).

3.2.2 Class Independent Positive Obstacle Detection

During road profile extraction, 3D points are either classified as belonging to obstacles or road. However, sometimes some small patches of the road are incorrectly classified as obstacles due to errors in disparity matching. These can be avoided by isolating obstacle points into groups. For this purpose method proposed in [10] by Meiqing Wu, et. al is used. Firstly, the pixels $p \in p_{obs}$ are refined by removing the pixels whose 3D points are 2.5 m above the camera rig and 15 m to the left and right sides from the left camera point. The method is based on u-disparity map peaks, where a new u-disparity map is generated based on an updated disparity map which only contains disparity values for refined obstacle pixel set. Every line segment in this u-disparity map corresponds to an obstacle in the image. Hysteresis type thresholding is used to remove the noise in the u-disparity image. High threshold is calculated based on (3.1) and low threshold is calculated as a ratio of the high threshold, to compensate for different object scales due to different depths. Then all the pixels in the u-disparity is categorized into three segments as pixels above high threshold, below low threshold and between high and low thresholds. All pixels below low threshold are ignored, considering them as noise and pixels above high threshold are selected. Other pixels between high and low thresholds are marked as noise if there is no single pixel in the 8-connected neighbourhood that belongs to a pixel above the high threshold, otherwise they are selected as well. Then a connected component analysis is performed based on u_span , which is a segment of the connected line in the threshold applied u-disparity map. Then based on a neighbourhood window, all the u_spans within that window are clustered together to a single segment. Then bounding boxes are generated for these segments, where left and right boundaries are directly found from the two columns of the segments in the threshold applied u-disparity map. Lower band is detected from the row corresponding to the maximum disparity value in the segmentation, which is referred from the road profile, where as upper band is detected by scanning the area bounded by other three bounds corresponding to the disparity range

in the segmentation, in the threshold applied u-disparity image. As an additional constraint, minimum number of pixels are defined to qualify that bounding box as an obstacle. The process is summarized in the flow chart in Fig. 3.7 and results are visualized in Fig. 3.8.

Figure 3.7: Flow chart for positive obstacle detection in class agnostic manner.

3.2.3 Instance-level Segmentation

Even though all obstacle points can be detected in a class agnostic manner, it is also important to categorize these obstacle points in to objects for motion prediction tasks. Furthermore, visual Simultaneous Localization and Mapping (vSLAM) can benefit from the masked objects in two ways. Firstly masking dynamic objects such as vehicles and humans helps in improving ego-vehicle localization estimation as it will then only rely on static feature matching, which is the underlying assumption in localization estimation. Secondly, it can help in improving the loop closure performance because it is easier to identify a revisited place with the absence of dynamic features from vehicles and pedestrians. Otherwise, such dynamic features may result in incorrectly not recognizing the revisited place as there can be a different set of vehicles and humans in the scene.

Figure 3.8: Obstacle bounding box generating process. Top to bottom: u-disparity for refined obstacle points, Threshold applied u-disparity, Segmented u-disparity, Bonding boxes detected

Deeply learnt two step regional proposal based object detectors achieve higher accuracy compared to the deeply learnt single shot detection networks. However, single shot networks can achieve real-time frame rates and thus most suitable for navigation applications. Therefore, Yolo [62] object detector is used in this study. Results from the Yolo network (bounding boxes) with the obstacle points detected above are combined to achieve real time instance segmentation. After positive obstacle detection step, refined obstacle points in the image can be obtained. Using this information, a new u-disparity image is constructed only for the refined obstacle points. In addition, an object detector is run, specifically YoloV3 [124], to generate object bounding boxes with class information, on the left rectified image. Then for each bounding box result (i.e. for each object) from the Yolo network, region of interest from u-disparity (u_{RoI}) is extracted, where the width of u_{RoI} is the same as object bounding box width and height of u_{RoI} is the height of u disparity image. Peaks in this u_{RoI} corresponds to the object in the bounding box and thus a mask can be extracted by selecting the disparity values corresponding to these peaks.

However, sometimes objects are partially occluded by another object in-front (such as far away car occluded from from a near-by car). In this situation simply

using the peaks to generate the mask results in wrongly classifying the partially occluded object as a part of the mask for the object in consideration, as u_{RoI} contains peaks for both the objects. Disparity values (range information) can be used to overcome this problem. Therefore, a maximum disparity value (d_{max}^{bb}) and a minimum disparity value (d_{min}^{bb}) are defined for each bounding box when considering the peaks inside u_{RoI} for each object. These values are defined based on two assumptions, (a) object always lie on the road and (b) bounding box that is occluded by the other has the lower y_m^{bb} of the two bounding boxes (i.e the bounding box behind is always occluded by the bounding box in front). d_{max}^{bb} is defined as the road disparity value corresponding to y_{max}^{bb} of the bounding box based on the first assumption. d_{min}^{bb} is defined as the d_{max}^{bb} of the bounding box behind that object, if there exists such as bounding box or otherwise set to zero. In order to account for errors that may arise from bounding box and road prediction, a certain percentage beyond d_{max}^{bb} and d_{min}^{bb} is searched inside u_{RoI} . Then in order to account for the holes and discontinuities in disparity, resulting object mask is subjected to morphological closing operation. As the final step, a convex hull can be generated for the morphological closed mask, to arrive at the final object mask. Convex hull is the smallest convex that contain all the points inside it. Results from the proposed method are as shown in Fig. 3.9. It is difficult to generate a mask for certain objects that are further away. In such cases, whole bounding box is masked.

3.3 Results and Discussion

The proposed methodologies are tested on real world scenarios with stereo cameras attached to a vehicle and processing the images captured by these cameras in real-time on an embedded system. Methods are tested with two gray scale stereo rigs and two colour stereo rigs when the stereo rigs are mounted either horizontal to the road surface or tilted towards the road surface. One such stereo rig and vehicles used to mount them are as shown in Fig. 3.10. Furthermore, system was able to operate satisfactorily with bright day light conditions as well as on cloudy days.

Figure 3.9: Instance level segmentation. (a) disparity image, (b) Yolo detection output and (c) instance segmentation using proposed method

Figure 3.10: Stereo rig mounts and vehicles used. (a) Contains two stereo rigs with different baselines and different cameras, (b) & (c) vehicles used to mount stereo rigs and data collection.

3.3.1 Road Slope Accuracy

Road slope accuracy was tested on the custom data collected during the experiments as KITTI [125] data is only on planar roads. Slope and angles estimated were evaluated with the slope angles measured. Slope is with relative to the slope of the vehicle position. Slope evaluation was verified under three situations, (a) no slope (planar road), (b) upwards slope and (c) downward slope as presented in Table 3.1. Locations for slope angle estimation experiments are as presented in Fig. 3.9.

Furthermore, slope images were categorized as, (a) up slope, (b) down slope, (c) no slope and (d) failures cases to evaluate the slope detection consistency. True Positive Rate (TPR) is used as the evaluation criteria, where; FN/Miss (False Negatives) are the cases where slope detection failed or slope detection is wrong and TP(True Positives) are correctly detected up, down and no slope scenarios. The results are as presented in Table 3.2.

Table 3.1: Slope Estimation Accuracy

S/N	Distance (to camera)	Slope Measured	Relative Slope			Estimated Road Surface Pitch
			Meas	Est	Error	
Planar road						
1	0 m	1°	-	-	-	0.616° (4m-9m road)
2	6 m	1°	0°	0°	0°	
3	11 m	0°	0°	1°	0°	0.67° (9m-14m road)
4	16 m	1°	0°	1°	1°	
5	11 m	0°	1°	-	1°	1.1° (9m-20m road)
6	16 m	1°	0°	1°	1°	
	Average Error				0.6°	
Downward sloped road						
1	0 m	3°	-	-	-	1.638° (4m-9m road)
2	6 m	4°	-1°	-1°	0°	
3	10 m	5°	-2°	-2°	0°	1.893° (9m-14m road)
4	15 m	7°	-4°	-2°	2°	
5	10 m	5°	-2°	-3°	1°	4.908° (9m-20m road)
6	15 m	7°	-4°	-3°	1°	
7	20 m	7°	-4°	-4°	0°	
8	25 m	5°	-2°	-5°	3°	
	Average Error				1.0°	
Upward sloped road						
1	0 m	0°	-	-	-	0.472° (4m-9m road)
2	6 m	0°	0°	1°	1°	
3	10 m	2°	2°	2°	0°	6.805° (9m-20m road)
4	15 m	7°	7°	-	-	
5	10 m	2°	2°	3°	0°	
6	15 m	7°	7°	4°	3°	
7	20 m	5°	5°	5°	0°	
8	25 m	6°	6°	6°	0°	
9	30 m	5°	5°	6°	1°	
	Average Error (Scenario I)				0.7°	
1	0 m (vehicle)	2°	-	-	-	1.695° (4m-9m road)
2	6 m	3°	1°	1°	0°	
3	10 m	4°	2°	2°	0°	1.695° (9m-14m road)
4	15 m	4°	2°	2°	0°	
5	10 m	4°	2°	0°	2°	3.169° (9m-20m road)
6	15 m	4°	2°	3°	1°	
7	20 m	5°	3°	3°	0°	
8	25 m	4°	2°	4°	2°	
9	30 m	6°	4°	4°	0°	
	Average Error (Scenario II)				0.6°

Figure 3.11: Slope detection experimental results. From left to right are examples for; downward slope, upward slope and upward slope. Top row represent the calculated slope angle at varying depths from the ego-vehicle.

Table 3.2: Slope Detection Consistency

	Number of frames	
	Seq 1	Seq 2
Upward slope (TP)	50	356
Downward slope (TP)	154	202
No slope (TP)	769	419
Failures (FP+FN)	95	58
Total frames	1068	1035
TPR	0.91	0.94

3.3.2 Execution Time

Execution time for different steps are as summarized in the Table 3.3. All the post tests are conducted using a computer running Ubuntu 16.04 with 32 GB RAM and Nvidia GTX 1080 graphics card while the methods are implemented using C++ following ROS (Robotic Operating System) architecture. Images are of size 960×604 and total of around 9,000++ images are used to generate the results. In order to increase the execution time of the full algorithm, threading is applied for stereo matching module and Yolo detection module as these two modules are mutually exclusive processors.

Table 3.3: Execution Time

Module	FPS
Stereo matching	204.0 Hz
Road and slope detection	38.8 Hz
Yolo detection	65.3 Hz
Instance segmentation	352.9 Hz
Full algorithm	20.3 Hz

3.3.3 Instance Segmentation Performance

As explained earlier, instance segmentation is quite useful for SLAM to avoid features from dynamic objects for the localization task. Therefore, in this study, instance segmentation is performed to obtain fast results with adequate accuracy to remove dynamic obstacles. The performance of the proposed method is compared with Mask-RCNN [60], which outperformed all the methods in COCO [61] 2016 detection challenge. Mask-RCNN is a famous object detection network with a separate sub-network for the instance segmentation task, based on the object bounding boxes detected by the network.

Since it is costly to generate ground truth for the collected data, instance segmentation evaluation is performed on KITTI [125] benchmark. KITTI contains a separate dataset for instance segmentation task with 200 images each for train and test datasets. Furthermore, KITTI provides stereo images as well as evaluation codes. The evaluation metric used is the Average Precision at 50% overlap

(AP50%). AP50% is the area under the curve of the Precision-Recall curve, when there is at least 50% overlap with the ground truth object. Performance is evaluated for five object classes (person, car, bus, motorcycle and bicycle) for 200 training images. KITTI images are of the size 1242×375 . Results for this subsection is generated using a laptop running Ubuntu 16.04 with 8 GB RAM and Nvidia GTX 850M graphics card. The results are as shown in Table 3.4. *Proposed_Yolo* presents the results when Yolo object detector is used and *Proposed_RCNN* presents the results when detection information from Mask_RCNN is used.

Table 3.4: Instance Segmentation Performance

Method	AP50%				
	Person	Car	Bus	Motorcycle	Bicycle
Mask_RCNN	41.7%	63.8%	17.6%	31.7%	23.3%
Proposed_Yolo	7.9%	36.0%	17.6%	0.0%	0.0%
Proposed_RCNN	28.1%	48.2%	17.6%	31.7%	4.7%

When analysing quantitative results in Table 3.4 and qualitative results in Fig. 3.12, it is evident that *Mask_RCNN* outperforms the proposed method. Superiority of the bounding box results of *Mask_RCNN* can be seen when analyzing *Proposed_Yolo* and *Proposed_RCNN* rows in Table 3.4. It can be seen that *Mask_RCNN* adhere to object boundaries well, while instance masks of the proposed method are bigger than the object. In certain situations, the whole bounding box is masked in the proposed method. This happens when the object is far away and when two objects are nearby. The reason is that the disparity variations alone is not good enough in such situations to separate the object from the background.

However, deep network based two stage object detector such as Mask_RCNN is very time consuming and resource intensive. Since the objective of the proposed segmentation task is to provide masks for dynamic objects to improve SLAM performance, latency is very important. Therefore, the proposed method is designed to generate a bigger mask (less accurate instance segmentation) at a lower latency, to suit real time tasks.

In order to evaluate the efficiency of the proposed method, computer resource consumption was used in terms of CPU and GPU utilization. The proposed method

Figure 3.12: Instance segmentation results for KITTI data. (a) Proposed method with Yolo detections and (b) Mask_RCNN.

is based on YoloV3 object detection and disparity based segmentation which are based on C and C++ coding. Mask-RCNN on the other hand is based on PyTorch framework and Python. Therefore, rather than comparing the time consumption, measuring resource utilization is more appropriate. Furthermore, both the methods involve large number of algorithms and large number of code files, making it very challenging to accurately estimate total number of floating point operations involved in each method. The results comparing CPU and GPU utilization are as shown in the Fig. 3.13 below. In order to measure the CPU utilization, percentage of CPU cycles and RAM space used are recorded. GPU utilization is measured based on Streaming Multiprocessors (SM) and memory utilization, recorded using *nvidia-smi* package at each second. Utilization was measured when each method is performing on KITTI instance segmentation images. The Fig. 3.13 clearly indicates that the proposed method utilizes less resources in terms of both CPU and GPU resources.

Figure 3.13: : CPU and GPU utilization over time by Proposed method and Mask-RCNN method.

3.4 Concluding Remarks

In this chapter, an efficient framework is proposed for understanding the road surface, calculating the road angle, detecting obstacles with 3D information and segmentation of obstacle points to object instances, in real time. The proposed framework has been tested with data collected in public roads in Singapore. Data is collected using four different stereo rigs covering different weather and traffic conditions such as sunny and cloudy days, during day time and evening time, peak and off-peak hours, sloped roads. Data is collected when stereo rigs are assembled at zero degree pitch angle as well as ten degrees pitch angle. The proposed framework can operate in 22.5 frame per second on collected data.

Chapter 4

Negative Obstacle Region Detection

4.1 Introduction

Obstacle detection is an important area of research for automatic robot navigation and obstacle avoidance. However, the main focus has been on detecting positive obstacles such as vehicles, pedestrians and anything above the ground plane. Equally important task is detecting negative obstacles such as holes, ditches etc. that lies below the ground surface, which has not been studied extensively. In this study, such holes and potholes are detected as an energy minimization problem. Stereo information combined with saliency is used to initialize the energy function and colour information is used to optimize the result. The optimized result is then subjected to hysteresis thresholding to arrive at the final negative obstacle region. Experimental results prove that the introduced method can successfully detect negative obstacles and run in real time (in the range 14-20 Hz) when using GPU for stereo matching. The proposed method was tested under three camera configurations with corridor, concrete-road and tar-road scenes. Proposed method perform well compared to the recent works.

Obstacles can be broadly divided into two main categories as positive and negative. The most intuitive and common type is the positive obstacle, which is an obstacle that lies on top of road surface or the ground plane such as pedestrians, vehicles, plants, buildings etc. Many studies have been conducted in positive obstacle detection. Negative obstacles are the obstacles that are below the ground surface such as holes, ditches, potholes etc.

Autonomous guided vehicles are expected to enter the road in the future and autonomous robots are expected to be deployed to clean roads and pedestrian ways. In fact there are autonomous robots used in cleaning corridors and path ways. These robots needs to avoid positive obstacles as well as numerous potholes and ditches to do their intended jobs. Autonomous robot or vehicle being unable to avoid a negative obstacle can lead to damaging itself and surroundings, thus making

it a critical component for safety purposes. Furthermore, unattended and open man-hole type negative obstacles are present even when the naturally occurred potholes are absent. On the other hand, absence of natural potholes means there is good maintenance, which require some mechanism to detect such negative obstacles and take corrective measures. In this study, the concern is detecting such negative obstacles from binocular stereo images.

Most common obstacles in road scenarios are either vehicles or pedestrians. Pedestrian detection is lavishly studied in literature and some are reviewed in [48, 126]. Different methods used in vehicle detection using vision images are reviewed in [127, 47]. Literature on pothole detection is not extensive as that of human or vehicle detection. The reasons could be lack of data, especially in many developed countries having a well maintained road network and on the other hand there's no such publicly available dataset with labelled ground truth data for the research community to work on.

Detecting negative obstacles (ditches, potholes, and other depressions) is one of the most difficult problems in perception for autonomous navigation. Its basic geometry is shown in Fig. 4.1.

Figure 4.1: Geometry of negative obstacle detection

Given the height H_{veh} , the range to the obstacle R and the width of the obstacle w , the small angle approximation is used to represent the angles α and $\alpha + \theta_n$ as in (4.1) and (4.2).

$$\alpha \approx \frac{H_{veh}}{R + w} \quad (4.1)$$

$$\alpha + \theta_n \approx \frac{H_{veh} + h}{R + w} \quad (4.2)$$

By considering similar triangles, h_{neg}^{vis} is defined as per (4.3).

$$h_{neg}^{vis} = \frac{H_{veh}w}{R} \quad (4.3)$$

Combining (4.1), (4.2) and (4.3), a solution is found for θ_n as per the (4.4) by eliminating h_{neg}^{vis} .

$$\theta_n \approx \frac{H_{veh}w}{R(R + w)} \quad (4.4)$$

This explains that the angle θ_n decreases significantly as the range R increases (because θ_n is inversely proportional to R^2), which makes negative obstacles so difficult to detect as range increases.

Therefore it is not wise to depend only on finding height difference information (which can be obtained from disparity information), when negative obstacle region is to be detected at a distance. Hence, in this study negative obstacle detection is formulated as an energy minimization problem. The energy function is initiated based on the 3D data obtained from the stereo input of the scene combined with the salient nature of a negative obstacle compared to the normal ground regions. Later it is optimized to minimize the overall energy to obtain the refined negative obstacle region which is then subjected to hysteresis thresholding to get the final negative obstacle region. The proposed method pipeline is shown in Fig. 4.2.

4.2 The Proposed Method

The overall architecture of the proposed method visualized in Fig. 4.2 is explained in detail in this section. Firstly, the energy function is introduced and explained as to how it will generate the desired result. Then the measures taken to increase the efficiency using superpixels and region of interest extraction process is discussed. Finally, weights initialization based on information from stereo matching and saliency detection is presented.

Figure 4.2: Proposed negative obstacle detection pipeline

4.2.1 Introducing the Energy Function

The purpose of the energy function is to identify the negative obstacle region, once optimized. The detection problem is formulated as a foreground segmentation task, where the negative obstacle region is the foreground. When minimizing the total cost of the function it tries to separate the image into two regions (foreground and background) constrained with a smoothness term. The use of an energy minimization approach for this task was motivated from the segmentation works such as Graph Cut [128]. The energy function defined in detecting negative obstacles is as expressed in (4.5).

$$E_{tot} = \sum_{i=1}^n W_i^{fg}(1 - P_{p_i})^2 + \sum_{i=1}^n W_i^{bg} P_{p_i}^2 + \sum_{i,j} W_{i,j}^{sm}(P_{p_i} - P_{p_j})^2 \quad (4.5)$$

P_{p_i} is the probability of i^{th} pixel p_i belonging to the foreground, i.e. pixel belonging to a negative obstacle where n being the total number of pixels. W_i^{fg} , W_i^{bg} and $W_{i,j}^{sm}$ are weights indicating p_i^{th} pixel belonging to foreground, background and similarity to the p_j^{th} pixel. $\sum_{i=1}^n W_i^{fg}(1 - p_{x_i})^2$ term defines the energy for the foreground region, i.e. negative obstacle region. If a pixel belongs to a negative obstacle, the weight W_i^{fg} will be high. Hence, when the energy of (4.5) is

minimized, it will force the probability of that pixel belonging to a negative obstacle (p_i) to grow closer to 1 in order to make $(1 - P_i)$ smaller. Similarly, higher value for the second term $\sum_{i=1}^n W_i^{bg} P_{p_i}^2$ encourages background pixel probabilities to optimize to a lower value. The last term $\sum_{i,j} W_{i,j}^{sm} (P_{p_i} - P_{p_j})^2$ will encourage pixel p_i and p_j to have similar probabilities of belonging to a negative obstacle, if the two pixels have higher similarity weight ($W_{i,j}^{sm}$). This can even be considered as a smoothness term. Thus, the pixels belonging to a negative obstacle can be detected by minimizing (4.5). In the next subsection, superpixel segmentation is introduced as a method to increase the efficiency of this process.

4.2.2 Superpixel Segmentation

Computational complexity of the proposed energy function in (4.5) increases with the number of pixels in the image. If the total number of pixels can be reduced the process can be expedited. Therefore, superpixels are used instead of merely using pixels. So, the equation in (4.5) is rewritten as (4.6), where pixels p_i are replaced with superpixels s_i . k is the total number of superpixels in the region of interest (detailed in subsection 4.2.3).

$$E_{tot} = \sum_{i=1}^k W_i^{fg} (1 - P_{s_i})^2 + \sum_{i=1}^k W_i^{bg} P_{s_i}^2 + \sum_{i,j} W_{i,j}^{sm} (P_{s_i} - P_{s_j})^2 \quad (4.6)$$

Superpixel is a collection of pixels that has been clustered together based on some similarity measure [129]. All superpixel methods can be categorized as classical segmentation and over segmentation methods. Classical superpixel methods are derived based on object colour information [130, 131] while over segmentation methods introduce an additional spatial constraint so that the pixels in the superpixel does not exceed a certain distance to the center of the superpixel [129, 132]. Therefore, over segmentation methods can generate spatially constraint superpixels which can be controlled easily to over segment the image. Therefore, over segmentation based methods are preferred over classical segmentation methods as a pre-processing step. Superpixel based methods are also of two major types based on the superpixel shape regularity. Some superpixel methods generate regular shaped superpixels while the other types generate irregular shaped superpixels.

This shape regularization is done by imposing an additional constraint on the shape regularity. In [129], authors present a simple yet powerful algorithm to generate regular shaped spatially constrained superpixels called SLIC-Superpixels. Because of the similarity of the method superpixel segmentation can be obtained in real time. This work really enabled using superpixels as a pre-processing step because it produces superpixels very fast. Method in [132] gives faster and better superpixel segmentation in terms of boundary recall and segmentation error. Hence, this method is used in the proposed pipeline. Comparison on these methods is presented in section 4.3.

In order to further reduced the computational complexity, rather than comparing all the superpixels in the image for the smoothness term $\sum_{i,j} W_{i,j}^{sm} (P_{s_i} - P_{s_j})^2$ in (4.6), superpixels within a maximum distance to the given superpixel is used. The maximum distance is defined as three times the minimum distance of all the distances between superpixels. This not only help in reducing the computational time but also helpful in disregarding similar coloured superpixels that are farther away from each other (as only colour distance is used in calculating the weights for the similarity $W_{i,j}^{sm}$, which will be explained in the subsection 4.2.5) to prevent them adding any signifance to the superpixel in consideration (as a negative obstacle region should be connected pixel wise in an image).

4.2.3 Defining Region of Interest

Negative obstacles always lie on the ground plane. This observation is used to reduce the computational complexity of the process. Rather than using the whole image to detect negative obstacles, only the image areas corresponding to the ground or road plane can be used. Therefore, firstly the ground plane is detected from the image. For this purpose, stereo information of the scene is used. Stereo matching is based on Semi-Global Matching which was first introduced in [133]. Example of a disparity image can be visualized in Fig. 4.3.

Two variants of disparity called *u* and *v* which were introduced in [6] is helpful in estimating the ground plane. v-disparity image is like a histogram of disparity image along rectified image rows where the peaks corresponds to road disparity values and u-disparity image is like a histogram of disparity image along rectified

Figure 4.3: Region of interest extraction. (a) left rectified image. (b) right rectified image. (c) disparity image from stereo matching. (d) v-disparity (e) u-disparity. (d) and (e) are derived from (c). (f) large obstacles removed disparity - by removing peak values from (e). (g) v-disparity - based on (f). (h) disparity profile of ground plane. (i) detected ground plane and region of interest (RoI) - RoI in red box.

image columns where the peaks corresponds to disparity values of large obstacles.

Non-parametric based road detection method in [10] was used to detect the ground surface. This method is fast, efficient and even work in sloped ground plane. This method has a simple implementation and has the ability to provide accurate road approximations even for non-planar road surfaces. First, based on the disparity image u-disparity is calculated where peaks in the u-disparity map corresponds to the obstacles. Then large obstacles are removed from disparity image based on the peaks on the u-disparity image. Points on u-disparity image exceeding a threshold (u_{th}) are identified as peaks. This step is there to help with road profile extraction by removing large obstacles which will otherwise lead to erroneous results in road profile. Threshold u_{th} is calculated based on the minimum height (h_{min}) to categorize a large obstacle, following (4.7).

$$u_{th}(d) = \frac{h_{min} \times d}{b} \quad (4.7)$$

where b is the baseline of the stereo system and d is the disparity value of a point. The minimum height is used as 0.3 m while the baseline is calculated at the time of camera calibration. Camera calibration process and stereo matching (and disparity calculation) method is explained in detail in Appendix A. Thresholds are calculated for each disparity value and stored in a look up table to speed up the process during run-time.

Once the peaks in the u-disparity are detected, the corresponding obstacles can be easily removed from the original disparity image. For each peak point in u-disparity, peak disparity and the column is known. Then in the original disparity image, all the disparity values in the same column are compared to the corresponding peak disparity value. If the two disparity values are the same, the corresponding points are marked as large obstacles. In this way an obstacle map (*obs-map*) is generated for the scene. Then the large obstacles removed disparity image is used to construct the v-disparity image. Then the disparity values corresponding to the road is identified by traversing this v-disparity image row-by-row. It is done based on several observations, (1) maximum intensity value in at least first few rows belongs to the road (i.e. bottom part of the image will mostly constitute from the road), (2) in adjacent rows the disparity value of the road in

the higher row (smaller row number) is always smaller than or equal to disparity value of the road in the lower row (higher row number). Road profile extraction is started from the bottom of the image. Based on the first assumption the detection process is initiated by identifying the highest intensity of the v-disparity map for the first few rows. Then road profile is approximated by traversing row by row based on the second assumption. For this purpose, first an initial road profile is constructed by just sorting the disparity value corresponding to the peak intensity value for each row of the v-disparity image. Then if the disparity value of the peak intensity of the row in consideration is equal or lower than (within a threshold limit to avoid outliers) that is of the previous row, that disparity is added to the disparity road profile. If no match found the same disparity as per previous row is considered. This process is repeated until the road profile stops moving to the left for a threshold number of rows.

Then each row of the original disparity image is checked with the corresponding disparity value from the road profile. If the disparity $d(r, c)$ is less than the corresponding disparity from the road disparity profile d_r^{rp} plus a pre-defined small threshold value d_{th} , that pixel in the rectified image is assigned as a road pixel p_{road} otherwise an obstacle pixel p_{obs} . That is, pixel $p \in p_{road}$ if $d(r, c) < (d_r^{rp} + d_{th})$ or $p \in p_{obs}$ otherwise. Then for $p \in p_{road}$ for which $d(r, c) > (d_r^{rp})$, p is checked whether it was identified as a large obstacle in the u-disparity based method, if yes p is re-assigned to $p \in p_{obs}$. Small threshold d_{th} is required because there are some road pixels that have higher disparity value than the median road disparity value. This process enables in detecting the disparity values corresponding to the ground plane at each image row which can be used to detect all pixels corresponding to the ground plane. This process is visually shown in Fig. 4.3. Once the ground plane is identified, the region of interest is defined as a rectangle region inside the ground plane with an offset from the borders of the image to remove inaccurate disparity values.

4.2.4 Initial Negative Obstacle Region

‘Initial Negative Obstacle Region’ (N) is identified by combining the potential negative obstacle region from 3D information (N_{st}) and the salient road region (N_{sail}),

in the region of interest defined as in section 4.2.3. N_{st} is identified as the points that lie below the road surface. In each row, disparity belonging to road is known from the process explained in subsection 4.2.3. Thus, the height of the road for each image row can be calculated from stereo information. Then the height of each pixel in the row is compared with the road height to find the points below a certain threshold (h_{th}^{neg}) and mark them as valid points for N_{st} . During this comparison any pixel in the row that is lower in depth than the depth of the road (with respect to the left camera) is excluded from N_{st} .

N_{sail} is calculated based on the salient regions on the road. At first, the salient image is calculated following the fine grained saliency approach proposed in [134]. Then the salient regions that may lie on positive obstacles are removed based on the dilated *obs-map* (which is generated as described in section 4.2.3). After that Otsu Thresholding [135] is applied to retain the most salient regions. ‘Initial Negative Obstacle Region’ (N) is then calculated by taking the pixels present in both N_{st} and N_{sail} . After which it is subjected to morphological opening to remove noises. Fig. 4.4 explains the process visually.

Figure 4.4: Initial Negative Obstacle Region N profiling process. (a) Superpixel image of the RoI. (b) 3D information based negative obstacle region N_{st} . (c) Obstacle removed saliency detection. (d) Otsu Threshold applied on (c) N_{sail} . (e) N combining (b) and (d)

4.2.5 Energy Function Initiation and Optimization

Weights for the i^{th} superpixel belonging to foreground W_i^{fg} , is calculated as the normalized energy of the i^{th} superpixel being a part of a negative obstacle $\hat{E}_{fg,i}$, i.e. the initial probability of that superpixel belonging to a negative obstacle.

The purpose of calculating E_{fg} is to find how strong is the probability of that superpixel belongs to the negative obstacle region at the initiation. In order to calculate the energy of the i^{th} superpixel $E_{fg,i}$ belonging to a negative obstacle, two terms are defined from (a) the energy term for initial negative N region E_i^N and (b) the energy term for road information E_i^{rd} as expressed in (4.8).

$$E_{fg,i} = E_i^N E_i^{rd} \quad (4.8)$$

Initial negative obstacle region detection is explained in subsection 4.2.4. When calculating the energy term E_i^N , the mean value of all the j pixel values $p(j, i)$ inside the i^{th} superpixel, that belongs to initial negative obstacle region N is calculated as in (4.9). n_i is the total number pixels in the i^{th} superpixel.

$$E_i^N = \frac{1}{n_i} \sum_{j=1}^{n_i} (p_{j,i} \in N) \quad (4.9)$$

The equation defined in (4.10) is used to calculate the probability, a superpixel belonging to ground G . Road detection is done based on the road information obtained through the process described in subsection 4.2.2. This term is helpful to remove noise from salient regions that may not belong in the road (e.g. salient regions in vehicles or surroundings) by assign is small energy to them. $p_{j,i}$ is the j^{th} pixel in the i^{th} superpixel.

$$E_i^{rd} = \frac{1}{n_i} \sum_{j=1}^{n_i} (p_{j,i} \in G) \quad (4.10)$$

If the initial probability of a superpixel is either very high or very low, it can be ascertained that the superpixel should belong to the foreground or to the background respectively. In (4.8), $E_{fg,i}$ is defined with strict constraints, that the i^{th} superpixel belongs to the foreground, i.e. belongs to the negative obstacle region. Therefore, initially it can be assumed safely, that if the normalized energy

(from 0 to 1) of the i^{th} superpixel ($\hat{E}_{fg,i}$) is higher than half (i.e. higher than 0.5), the i^{th} superpixel belongs to the foreground. However, it is not the same in the reverse direction. That is, in order to safely assume that the i^{th} superpixel belongs to the background (non negative region), $\hat{E}_{fg,i}$ should be negligible (i.e. less than 0.001). All other superpixels should have an equal chance of belonging to the foreground or the background initially. Therefore, W^{fg} is initialized as shown in (4.11).

$$W_i^{fg} = \begin{cases} \hat{E}_{fg,i}; & \text{if } (\hat{E}_{fg,i} > 0.5 \text{ or } \hat{E}_{fg,i} \leq 0.001) \\ 0.5; & \text{otherwise} \end{cases} \quad (4.11)$$

Weight for each superpixel belonging to background W_i^{bg} , is simply calculated as taking the inverse form of the foreground weights which is expressed in mathematics terms in (4.12).

$$W^{bg} = (1 - \hat{E}_{fg}) \quad (4.12)$$

Weight for the similarity between i^{th} and j^{th} superpixels $W_{i,j}^{sm}$ are calculated using the equation specified in (4.13). Similarity between the superpixels are calculated using the colour similarity.

$$W_{i,j}^{sm} = \exp\left(\frac{-E_{clrdiff}^2(s_i, s_j)}{C_{clrdiff}}\right) \quad (4.13)$$

Similarity among superpixels were calculated using the distance between RGB colour components for a colour image and intensity distance for a grey image. Specifically, the colour difference $E_{clrdiff}(s_i, s_j)$ is calculated using the Euclidean distance. Constant $C_{clrdiff}$ is used to control the term. The colour similarity term will enforce similar colour superpixels to have similar probability of belonging to negative obstacle region.

All the probabilities for superpixels belonging to a negative obstacle s_i is initialized using normalized energy $\hat{E}_{fg}$ defined in (4.8). In the energy minimization phase, least squares optimization is used on (4.6), using the auto-differentiate function in Ceres library [136]. Fig. 4.5 shows how the initial guess is optimized.

Figure 4.5: Detection of negative obstacle region. (a) Image RoI. (b) Salient RoI. (c) Otsu thresholded (b) N_{sail} . (d) 3D information based negative region N_{st} . (e) Initial Negative Obstacle Region N . (f) Superpixel segmentation on RoI. (g) Initial probabilities of superpixels P_{s_i} belonging to negative obstacles. (h) Optimized results for P_{s_i} . (i) Hysteresis thresholded results. (j) Detected negative obstacle regions - cropped image

4.2.6 Negative Obstacle Region Detection

The optimized negative obstacle region is then subjected to hysteresis thresholding. Since all optimized probability of superpixel belonging to a negative obstacle P_{s_i} are varying from 0 to 1, high threshold (th_h) and low threshold for hysteresis is defined to be between the same. Lower threshold is defined as $th_l \times th_h$, where th_l is a parameter to select accordingly. Firstly, all the superpixels with P_{s_i} higher than th_h are marked as superpixels belonging to final negative obstacle region. Then if the neighbouring superpixels (to these marked superpixels) with P_{s_i} higher than th_l are also marked as superpixels belonging to final negative obstacle region. The results of this process can be visualized as in Fig. 4.5.

4.3 Results and Discussion

Main focus in this study is to apply the proposed method in robotics applications (mainly cleaning) to avoid negative obstacles when navigating. In order to validate the proposed idea, stereo images with negative obstacles are collected on corridors with off-the-shelf ZEDTM stereo camera and on concrete road with custom made stereo rig using PointGrey Flea[®]3 cameras. Furthermore, some images from CCSAD stereo dataset [137] that contain potholes in tar roads are selected. No suitable pothole images are found from KITTI [125] stereo benchmark. Two sequences from PointGrey stereo, two sequences from ZEDTM stereo and one sequence with CCSAD stereo is created in evaluating the proposed method.

PointGrey stereo camera rig is custom designed to see far away objects, hence the baseline is very large, In this experiments the exact baseline used is 0.7265 m and the size of the images captured is 640x422 for the customized PointGrey stereo rig. MATLAB in built ‘Stereo Camera Calibrator App’ is used in finding the internal and external parameters of the stereo rig. Range for this configuration is 10-30 m. When using ZEDTM, visible range or disparity is around 8-10 m with 0.12 m baseline. Intrinsic and extrinsic calibration parameters for ZEDTM stereo rig can be directly downloaded from their database and then refined with ZEDTM calibration toolbox. Data collection is done with images of size 1280x720, however during testing time the image is down sampled by half. CCSAD dataset provides the calibration information and stereo images are of the size 1096x822, which are

down sampled by half during the testing. Range for this configuration is 5-15 m. GPU based Semi-Global Matching implementation called libSGM¹ is used to generate the stereo matching. Computer with an Intel Core i5-8400 2.80GHz 6 processor and a Nvidia GTX 1080 Ti graphics card is used for data processing. Graphics card is used to make stereo matching real time.

Fig. 4.5 summarizes the main processes proposed in this paper for detecting negative obstacles. The figure shows how the negative obstacle region get initialized from the stereo input and how it gets optimized to arrive at the final result. It can be seen that from the stereo output there are outliers detected as possible negative obstacles and how these get corrected after the optimization step. The reason for this correction is due to the information from colour and the energy function forcing same colour superpixels to have same probability of belonging to a negative obstacle. Detecting negative obstacles is a very challenging task even when humans are driving vehicles. Humans often rely on salient dark regions on the road to suspect the presence of a negative obstacle in the far range. This suspicion allows to slow down the vehicle and to avoid any damage if it really is a negative obstacle. This motivated to include colour similarity and giving more attention to salient regions in the proposed optimization algorithm. However, if there is a strong density of false positives with in small range, this error can get aggregated and detected as a negative region as well. Therefore, it is important to have an accurate disparity image, where occasional errors can be rectified in the process. Even though there can be such errors these can be rectified by integrating the results across few frames of the same scene (time integration).

Each sequence is evaluated based on *Precision*, *Recall*, *False Positive Rate (FPR)* and *F-measure* where $Precision = TP / (TP + FP)$, $Recall = TP / (TP + FN)$, $FPR = FP / (FP + TN)$ and $F = 2(Precision \times Recall) / (Precision + Recall)$. *TP*, *FP* and *FN* stands for *True Positives*, *False Positives* and *False Negatives* respectively. *F-measure* is same as the *Sørensen-Dice coefficient*, which provides a good indication as to how accurate the overall detection to the ground truth combining both precision and recall. The evaluation is done at pixel level, i.e. *TP*, *FP* and *FN* are evaluated for each pixel detected as negative obstacle

¹libSGM: <https://github.com/fixstars/libSGM>

compared to ground truth pixel. Ground truth is generated by hand labeling the negative obstacle regions using labelme² tool.

4.3.1 Parameters tuning

The proposed method depends on a total of five parameters, which are (a) minimum height (h_{min}) to define $u_{th}(d)$ for large obstacle removal detailed in section 4.2.3, (b) height difference (h_{neg}^{min}) to define N_{st} detailed in section 4.2.4, (c) number of superpixels (k) for superpixel segmentation detailed in section 4.2.2, (d) high threshold (th_h) and (e) low threshold (th_l) for hysteresis detailed in section 4.2.6.

Since three different camera configurations are used with different image sizes and the range seen by each configuration is different to each other, parameters should be tuned for each configuration separately. The final values used for these parameters are summarized in Table 4.1 for each configuration. Out of these five parameters h_{min} is rather a setting than a parameter, as it is just a measure for the minimum height of a large obstacle. Therefore, it is globally set as $h_{min} = 0.3$ meters for all camera configurations. Since the number of superpixels (k) does not change the output result drastically, it is selected based on the varying *F-measure* or the *Sørensen-Dice coefficient* values.

h_{neg}^{min} is a parameter that controls the output of the 3D information based negative obstacle region (N_{st}) detection. Furthermore, optimization algorithm initiation is affected by N_{st} . Therefore, best h_{neg}^{min} is selected that minimizes the number of false positives (hence best *Precision*) for N_{st} region detection. However, as visible in Fig. 4.6, compared to PointGrey and ZED data CCSAD has a very low *Precision* compared to its' *Recall* and when $h_{neg} > 1$ *Recall* decreases sharply. Hence, for CCSAD data $h_{neg}^{min} = 1$ is selected. th_h and th_l can be considered as parameters that will directly affect the final detection result. Therefore, PR (Precision-Recall) curves and ROC (Receiver Operating Characteristics) curves are presented for these parameters. ROC curve does not cover the full range when there is a large bias between the foreground class and background class, similar to negative obstacle detection problem. Hence, AUC (Area Under Curve) is calculated only for the PR curve. The results are as shown in Fig. 4.6. Then the best values are selected

²labelme: <https://github.com/wkentaro/labelme>

Table 4.1: Selected Parameters for Testing

Parameter	PointGrey	ZED	CCSAD
h_{min}	0.3 m	0.3 m	0.3 m
k	300	400	500
th_h	0.4	0.3	0.4
th_l	0.1	0.1	0.4
h_{neg}^{min}	0.15 m	0.07 m	0.01 m
$dist(Plane)$	1.3	1.4	1.5
$dist(Plane_RoI)$	4.2	2.4	1.5
$th_h(Plane_RoI)$	0.4	0.5	0.6
$th_l(Plane_RoI)$	0.5	0.2	0.7
$th_h(Dark)$	0.4	0.5	0.7
$th_l(Dark)$	0.4	0.4	0.7
$k(SLIC)$	306	375	625

based on *F-measure* along the curves and summarized in Table 4.1.

Energy minimization is an iterative process, where minimization stopping point needs to be determined. Based on the experiments conducted, minimization is stopped when the change in the overall cost is less than 0.01. Total energy variation against number of iterations are shown in Fig. 4.7 for two stereo systems based on images from two sequences.

4.3.2 Comparison with others

The proposed method is based on stereo images. In [34, 35] authors use stereo based method by fitting a 3D plane in the disparity image to identify the most prominent plane in the scene as the ground plane. RANSAC is used when finding the plane parameters for best fitting plane. Then the points corresponding to the disparity values below this surface are considered negative obstacles. In [35], the result is improved by integrating multiple frames outputs and ego motion. However, this comparison is limited to a single frame scenario. Even though the plane fitting in [34, 35] is by using the whole disparity image, it leads to erroneous fitting when there are multiple objects present in the scene. Therefore, this process is enhanced by considering the disparities in the region of interest (calculated for the proposed method). Final result is subjected to morphological closing to improve the output. Results for these two methods are presented in Table 4.2 in

(a)

(b)

(c)

Figure 4.6: Parameter fine tuning and selection process. (a) Variation of Recall(R), Precision(P) and F-measure (F) for different h_{neg} values. (b) ROC (Receiver Operating Characteristics) and PR curves when changing th_h . (c) ROC and PR curve when changing th_l . h_{neg} is selected based on the best P value. th_h and th_l parameters are selected based on F maximizing point in the PR curve.

Figure 4.7: Energy minimization in iterations. Each graph shows how the total energy is minimized in successive iterations for each stereo set. Different colours represent different images. Total energy is calculated calculation as per (4.6).

rows *Plane* and *Plane_RoI* respectively. *Plane_RoI* based negative obstacle detection process is visualized in Fig. 4.8. Most important parameter in plane fitting method is the maximum distance from the plane to a given point ($dist(Plane)$ and $dist(Plane_RoI)$) to classify whether that point belongs in the plane or not.

For each camera configuration the best $dis(Plane)$ and $dis(Plane_RoI)$ is found by following the *F-measure* in the PR curve, similar to as in Fig. 4.6 and are summarized in Table 4.1. Additionally, the negative region detected in *Plane_RoI* can be used as the ‘Initial Negative Obstacle Region’ (N) for the proposed method. Therefore, negative obstacle detection result following this method is presented in the row *Proposed (Plane_RoI)* in Table 4.2. Best values for th_h and th_l are experimentally found following the same procedure detailed in section 4.3.1 and presented in Table 4.1 as $th_h(Plane_RoI)$ and $th_l(Plane_RoI)$ respectively.

Most vision based methods assume that the negative obstacle region is darker than that of its surroundings [36, 37, 38, 43], where as in this study it is assumed to be salient. To check the effect of this, energy calculation ($E_{fg,i}$) in (4.8) is modified as (4.14 below).

$$E_{fg,i} = E_i^{clr} E_i^{N_{st}} E_i^{rd} \quad (4.14)$$

where $E_i^{N_{st}}$ is the probability of i^{th} superpixel belongs to 3D information based negative obstacle region N_{st} as in (4.15) and E_i^{clr} is an indication as to how dark the superpixel is as in (4.16).

Table 4.2: Negative Obstacle Region Detection Results Comparison.
Precision(P)=TP/(TP+FP), Recall(R)=TP/(TP+FN), and
F-measure=2PR/(P+R)

Detection Method	PointGrey Images		
	Precision	Recall	F-measure
<i>Plane</i>	0.595	0.053	0.097
<i>Plane_RoI</i>	0.532	0.628	0.576
<i>Proposed(Plane_RoI)</i>	0.759	0.871	0.811
<i>Proposed(Dark)</i>	0.822	0.900	0.859
<i>Proposed(SLIC)</i>	0.856	0.888	0.872
<i>Proposed(CFTP)</i>	0.883	0.910	0.896
	ZED Images		
	Precision	Recall	F-measure
<i>Plane</i>	0.692	0.025	0.048
<i>Plane_RoI</i>	0.643	0.715	0.677
<i>Proposed(Plane_RoI)</i>	0.718	0.854	0.780
<i>Proposed(Dark)</i>	0.815	0.851	0.833
<i>Proposed(SLIC)</i>	0.763	0.826	0.793
<i>Proposed(CFTP)</i>	0.787	0.909	0.844
	CCSAD Images		
	Precision	Recall	F-measure
<i>Plane</i>	0.713	0.070	0.127
<i>Plane_RoI</i>	0.553	0.628	0.588
<i>Proposed(Plane_RoI)</i>	0.677	0.677	0.677
<i>Proposed(Dark)</i>	0.896	0.108	0.193
<i>Proposed(SLIC)</i>	0.570	0.579	0.574
<i>Proposed(CFTP)</i>	0.731	0.607	0.663

$$E_i^{N_{st}} = \frac{1}{n_i} \sum_{j=1}^{n_i} (m_{j,i} \in N_{st}) \quad (4.15)$$

$$E_i^{clr} = \frac{255}{clr_i + 1} \quad (4.16)$$

where clr_i is the mean gray-scale colour of i^{th} superpixel, 255 is the maximum colour value and 1 is to stop E_i^{clr} going to infinity when clr_i is zero.

The rest of the minimization process remains the same. Negative obstacle region detection based on this method is presented Table 4.2 in row *Proposed (Dark)*. Best values for th_h and th_l are experimentally found and presented in Table 4.1 as $th_h(Dark)$ and $th_l(Dark)$ respectively.

In addition the results of the proposed method for two different superpixel segmentation methods are presented. Results based on SLIC superpixels [129] are presented in the row *Proposed (SLIC)* and CFTP superpixels [132] in the row *Proposed (CFTP)* in Table 4.2. While the default number of superpixels (k) configuration is used for *Proposed (CFTP)*, for *Proposed (SLIC)* similar number is used as indicated in Table 4.1 as $k(SLIC)$ to obtain the maximum *F-measure*. th_h and th_l are kept as the default values.

When comparing the results in Table 4.2, the proposed method (*Proposed (CFTP)*) is able to achieve the best results for PointGrey and ZED data. Best results for CCSAD data is achieved when the proposed optimization is applied on top of *Plane_RoI* method. *Proposed (CFTP)* results (F-measure) is almost similar to the best as well. *Plane_RoI* method is when *Plane* method in [34, 35] is guided with region of interest proposed in the proposed method and it outperform *Plane* method in all data types. Applying optimization on top of *Plane_RoI* further improves the final detection results for all cases. When comparing results of *Proposed (Dark)* to *Proposed (CFTP)* and *Proposed (SLIC)*, salient feature is validated over the assumption that negative obstacles are darker, especially for CCSAD data as potholes present in that dataset are not so dark. Furthermore, CFTP method outperforms SLIC method. It was very challenging to collect many naturally occurred on-road negative obstacle stereo data (in Singapore), instead uncovered man made holes and some data from CCSAD data are used to prove the proposed idea. More

Figure 4.8: RoI based plane fitting process. (a)-(b) Example on seq4 (ZED). (c)-(d) Example on seq5 (CCSAD). (a) & (c) Points on the calculated plane are shown in blue while points outside the plane are in red. (b) & (d) Points outside the plane in the downward direction identified as negative obstacle region in blue.

results covering all sequences are shown in Fig. 4.9 and Fig. 4.10 shows failure cases.

The proposed method can run in about 14-20 Hz depending on the camera configuration as indicated in Table 4.3, making it suitable to be used in real time applications. It is observed that higher number of superpixels results in increased computational time. Furthermore, optimization method runs faster when $N(st)$ is considered to be from RoI based plane estimation method.

Since the proposed method is based on stereo matching, it suffers the inherent depth limitations. After certain range the performance drops drastically. In Fig. 4.11 this effect is visible, especially PointGrey line shows consistent detection up to 17m, after which it reduces drastically and eventually fails to detect anything at all. Furthermore, as can be seen from results in Table 4.2, PointGrey achieves the best results followed by ZED and CCSAD. This can be mainly due to the fact that the sizes of the negative obstacles captured in these sequences also follows the same order (biggest negative obstacles in PointGrey and smallest in CCSAD).

Figure 4.9: Negative obstacle region detection examples. (a)-(b) PointGrey-seq1. (c)-(d) PointGrey-seq2. (e)-(f) ZED-seq3. (g)-(h) ZED-seq4. (i)-(j) CCSAD-seq5.

Figure 4.10: Failed cases in negative obstacle detection. (a) False positives in seq 1, (b) Incomplete detection (false negatives) in seq 2, (c) Failing to detect (false negatives) in seq 3, (d) Incomplete detection (false negatives) in seq 4 and (e) Almost incomplete detection (false negatives) in seq 5

Table 4.3: Algorithm Execution Time per Frame

	PointGrey (<i>ms</i>)	ZED (<i>ms</i>)	CCSAD (<i>ms</i>)
Stereo matching	2.00 ± 0.18	2.00 ± 0.00	2.00 ± 0.00
Road extraction	3.62 ± 0.49	2.06 ± 0.23	2.27 ± 0.45
Superpixel (CFTP)	9.48 ± 0.73	11.49 ± 2.74	16.09 ± 2.02
Superpixel (SLIC)	28.38 ± 1.06	22.6286 ± 2.33	29.64 ± 1.15
Optimization	30.83 ± 2.17	34.49 ± 4.47	49.18 ± 5.11
Proposed (CFTP)	46.21 ± 1.88	49.17 ± 5.25	69.45 ± 6.70
Plane fitting	6.59 ± 1.71	5.51 ± 0.65	6.18 ± 0.39
Plane fitting (RoI)	1.93 ± 0.25	2.03 ± 0.17	3.00 ± 0.00
Optimization (RoI)	20.65 ± 2.25	22.49 ± 3.93	33.00 ± 3.83
Proposed (Plane RoI)	38.03 ± 2.17	39.85 ± 3.86	56.36 ± 5.01
Image size	640x422	640x360	548x411
Sequences	seq1-14 imgs seq2-15 imgs	seq3-16 imgs seq4-19 imgs	seq5-11 imgs

Figure 4.11: Negative obstacle detection accuracy variation with the distance to the negative obstacle from the camera. F-measure used in calculating the detection accuracy. Three potholes analysed for three camera configurations.

4.4 Concluding Remarks

In this chapter, a global optimization approach combining information from several sources, is proposed to detect negative obstacles. Information from stereo, colour and saliency nature are combined into a single equation to achieve a common objective. To the best of the knowledge, this is the first attempt to use global optimization on different information sources for the negative obstacle detection task. The proposed method is tested with concrete-road, tar-road and corridor based scenarios and have obtained promising results. The proposed method can achieve better results compared to the recent work. Moreover, the optimization method in the proposed work can be applied to improve the performance of negative obstacle detection by plane fitting method. Furthermore, using salient nature of negative obstacles can lead to better performance compared to assuming that negative obstacles are darker than the surroundings.

However, the method needs to be further analyzed with different environmental conditions. Collecting data with negative obstacles on different scenarios is a challenging task, which has resulted in very few researches conducted in this area. The most challenging reason is the unavailability of benchmark dataset for nega-

tive obstacles. To the best of our knowledge, any other work in negative obstacle detection has not compared their results with others as well due to these reasons. This challenge remains open as a future research work to facilitate more research in this area. Testing the performance of the proposed method with water filled negative obstacles will be an interesting test bed. Moreover, with a good benchmark proposed work can be compared with lidar based methods as well.

Chapter 5

Multiple Object Tracking with attention to Appearance, Structure, Motion and Size

5.1 Introduction

Tracking is a challenging problem in many video analysis tasks where an object defined by a bounding box, is to be identified and assigned a unique identity over all the frames it appears in an image sequence. Tracking is an important task in many applications such as surveillance, autonomous driving & advance driver assistance systems, behaviour analysis, motion prediction and particle transformation analysis.

Tracking research can broadly be divided in to multiple Object Tracking (MOT) and single object tracking. While MOT assumes object detection as prior knowledge, the latter tries to localize and track an unknown object that has only been described by the localization information at the first frame.

MOT tries to track multiple objects present in all the frames over the given image sequence. MOT generally has an object detector to detect the objects in each frame and then followed by detections association method to track them over time. Almost all the MOT methods follows this type of tracking-by-detection framework. Methods used in MOT can be separated to *Online methods* and *Offline methods* according to how they use the object detections in the image sequence. Offline methods make use of all the object detections available from the whole image sequence and handle the tracking as a global optimization problem. In contrast, online methods only rely on the information from object detection up to the current frame, which makes it suitable for real time applications.

In this chapter, an online tracker is proposed which can be used for multiple class object tracking. In the proposed method object detector is used to detect the object bounding boxes in an image frame and associate these detections using Hungarian Algorithm [118] based on pair-wise matching scores calculated between detections in the current frame against the tracks in the memory. The matching

cost calculation is based on four traditional features to improve both efficiency and accuracy. The first two features used are appearance and structure based, where both features are extracted using multiple histograms in a grid assembly, opposed to using a single histogram. Motion dissimilarity is based on the distance between predicted object location and observed detection location. The last dissimilarity component is calculated based on the Intersection over Union (IoU) between the predicted bounding box size and the current detection bounding box size. Furthermore, a method is proposed to address false negatives arising from the object detector, for a stable track. This overall tracking process is presented as a flow chart in Fig. 5.1. Proposed framework gained the expected efficiency, achieving 150+ fps for KITTI data and 27.0 fps and 17.1 fps for MOT 17 training and test data respectively, with competitive accuracy with the recent work.

5.2 The Proposed Method

Even though deep learning based MOT methods [88, 100, 101, 85] can yield accurate tracking results, these methods require a sizable amount of training data in comparison to traditional hand crafted feature based methods. Furthermore, deep learning methods in general are time consuming. Review on deep learning based multiple object tracking methods presented in [138], shows that the best frame rate achieved is 6.7 FPS for MOT15 and 2.8 FPS for MOT16 test data, among the top performing trackers. The proposed method in this work is based on hand crafted features where it is mainly based on object appearance, size and motion. Based on the object appearance, a colour histogram based matching and structure based matching measures are defined. In this work, it is shown that by following a grid based histogram matching can provide higher accuracy rather than using a single histogram to represent a detection in terms of appearance. Hungarian algorithm [118] is used in assigning objects to tracks. Furthermore, a method is proposed to compensate the false negatives arising due to detector failures for stable tracks.

The proposed tracking method mainly consists of three major components, (a) Matching Cost Computation, (b) Track Match and Update and (c) Tracking False Negatives for Stable Tracks, which are discussed in detail in this section. In the proposed tracking method, it is assumed that at every frame a set of detected object

Figure 5.1: Flow chart of the proposed tracking framework, where $bb_{i,det}$ is the bounding box of the i^{th} detection, $LBPH$ stands for Linear Binary Pattern Histogram and $MC(i, j)$ is the dissimilarity cost between i^{th} object detected in the current frame with the j^{th} object in the track list as defined in (5.1).

bounding boxes along with object class and detection confidence are available, since the tracking by detection framework is used.

5.2.1 Matching Cost Computation

The purpose of matching calculation is to find a quantitative value to measure how similar is given two objects (an object in the previous frame and in the current frame). Matching cost $MC(i, j)$ between i^{th} object detected in the current frame with the j^{th} object in the tracked objects list is defined as a weighted sum of four distance measures based on appearance distance $app(i, j)$, structural distance $str(i, j)$, location difference $loc(i, j)$ and size difference $iou(i, j)$ between i^{th} detection and j^{th} track as defined in (5.1).

$$MC(i, j) = W_{app} \cdot app(i, j) + W_{str} \cdot str(i, j) + W_{loc} \cdot loc(i, j) + W_{iou} \cdot iou(i, j) \quad (5.1)$$

W_{app} , W_{str} , W_{loc} and W_{iou} are the weights for the contributions from each of the four distance measures. All four distance measures are defined such that they vary between 0 and 1.

5.2.1.1 Appearance based Distance

Appearance of an object is an important clue when tracking as it can be used to recognize the object in different frames as well as to differentiate with other objects. However, with time the object appearance may be subjected to changes due illumination changes, view-point changes and deformations. Therefore, it is important to define a method to capture or compensate these changes in an appearance feature. To address this problem, in this work a grid based multiple histogram matching method is proposed to be used as an appearance feature. Appearance based distance is measured using the correlation between the colour histograms of the i^{th} detection and j^{th} track. Histogram matching is done against a maximum of three colour histograms extracted from the history of the track. In this work, HSV (hue, saturation, value) colour space is used because it is more robust for illumination changes in the frame compared to the RGB (red, green, blue) colour space. Hence, H and S variants are used for histogram matching. Furthermore, a grid based structure is used when calculating the histogram for each object bound-

ing box which results in a histogram by concatenating n (number of grid cells per bounding box) histograms, rather than using single histogram. The grid needs to be designed appropriately such that it is not too big nor too small. Implementation parameters are discussed in section 5.3.2. Since an object can be partially occluded in a given instance, an occlusion map is defined for this grid structure to be used when histogram matching to avoid any bias from inaccurate information. Fig. 5.2 visualize an example on the grid structure and the appearance history of a track.

Figure 5.2: Grid structure and appearance based history of a track. (a) current frame matched detection and (b), (c) and (d) are the visualizations of the three appearance history of the matched track kept in memory. Green boxes indicate the grid structure and shaded area by red lines indicate the occlusion. Yellow colour indicates the bounding box of the object in front of the track. Grid is 3×4 . 1^{st} , 4^{th} , 7^{th} and 10^{th} grid cells are occluded due to object in front (yellow bounding box) for (a) and (d). 2^{nd} , 5^{th} , 8^{th} and 11^{th} grid cells are considered non-occluded as yellow bounding box does not cover at least 50% of the grid cell area. No grid cell is occluded for (b) and (c). When appearance matching, 2^{nd} grid cell of (a) is compared with 2^{nd} grid cells of (b), (c) and (d) and average is taken. 1^{st} , 4^{th} , 7^{th} and 10^{th} grid cells are not considered because it is occluded for (a) and all the remaining grid cells are considered in matching.

Since each object is defined in terms of a bounding box, a grid based histogram matching helps to minimize the errors that may otherwise arise due to background. Such situation is presented in Fig. 5.3. When using single histogram to represent an object, background pixels can sometimes change the histogram giving rise to inaccurate matching, especially if the object is re-appearing after occlusion.

Appearance based distance between i^{th} detection and j^{th} track is defined as in (5.2).

$$app(i, j) = 1.0 - \frac{\sum_l \sum_m occl(m, i) \cdot occl(m, j)^l \cdot corr(H_{m,i}, H_{m,j}^l)}{m \cdot l} \quad (5.2)$$

where l is the number of concatenated histograms kept in the memory for each track from its history of detections which can be up to a maximum of three, and m is the number of cells in the grid. The procedure followed in updating histograms for a track is discussed in section 5.2.2. $occl(n, i)$ and $occl(n, j)^k$ are the occlusion indicators for the m^{th} grid cell of the i^{th} detection and n^{th} grid cell of the j^{th} track l^{th} history map, respectively. $corr(H_{m,i}, H_{m,j}^k)$ is the correlation between the histogram of the n^{th} grid cell of the i^{th} detection $H_{m,i}$ and m^{th} grid cell of the j^{th} track l^{th} history map $H_{m,j}^l$. Correlation indicates the similarity between two histograms and output a value between 0 and 1. Hence, the sum of histogram correlations is divided by total number of histograms ($m \times l$) and then take one minus of that as the appearance distance. Correlation between any two histograms $corr(H_a, H_b)$ is calculated based on (5.3).

$$corr(H_a, H_b) = \frac{\sum_I (H_a(I) - \bar{H}_a)(H_b(I) - \bar{H}_b)}{\sqrt{\sum_I (H_a(I) - \bar{H}_a)^2 \sum_I (H_b(I) - \bar{H}_b)^2}} \quad (5.3)$$

where,

$$\bar{H}_c = \frac{1}{N} \sum_J H_c(J) \quad (5.4)$$

in which N is the total number of bins in the histogram.

Occlusion parameter $occl$ for each grid cell is either 1 or 0 depending on the level of occlusion for that particular grid cell. Level of occlusion is defined based on the bounding box overlap. If any two or more bounding boxes overlap with each other, the bounding box belonging to the object that has the highest detection confidence is considered to be in front and the overlapping areas with other bounding boxes are considered to be occluded areas for those bounding boxes. Then in a given grid cell, if more than 50% of the pixels are occluded in such a way, the occlusion parameter for that grid cell is set to 0, otherwise set to 1 as presented in Fig. 5.2. This is expressed in (5.5) below, where n_m is the total number of pixels in the m^{th}

Figure 5.3: Grid based histogram matching against single histogram matching. (a) grid-based concatenated histogram based matching and (b) single histogram based matching. In (a) identity of the track ID 3 is occluded by the person walking in front (track ID 105) is kept successfully when matching with grid-based concatenated histogram. In (b) track ID 3 is falsely matched with track ID 72 after occlusion. Track ID is magnified for better visualization.

grid cell and n_m^{occl} is the number of pixels occluded in that cell.

$$occl(m) = \begin{cases} 1; & \text{if } n_m^{occl} > 0.5 \times n_m \\ 0; & \text{otherwise} \end{cases} \quad (5.5)$$

The appearance cost is defined as an average based on l frames in track history rather than considering the maximum response from one history frame. Reason for such measure is to avoid track drifting based on the most recent tracked detection (mostly when the track is partially occluded by another object and to avoid identity switch) as one of the concatenated histogram in track memory is always the concatenated histogram of the immediate track history as explained in section 5.2.2. Also refer to Fig. 5.2 track history containing the object in front of the track.

5.2.1.2 Structure based Distance

Structure is another clue that can be used in recognition task. Linear Binary Pattern (LBP) [139] is a good feature that can capture structural information from an image and is a robust feature for illumination variances. LBP outputs a binary code word for each pixel by comparing pixels values in a 3x3 neighbourhood with the center pixel value. Such example is presented in Fig. 5.4. LBP image is a collection of 8 bit binary code words out of all possible code words (in 3x3 neighbourhood $2^{9-1} = 256$ code words), that can be represented as a histogram. This is known as the LBP Histogram (LPBH). In [140] Ahonen et al. used LBPH in a grid structure for face recognition task and achieved the best results at the time. In grid structure LBPH captures the shape of the nose, eyes, mouth and other shapes in the face which can be then used for recognition. In object level LPBH may not be a good feature to differentiate one another as a given object class can have very similar shape. But when considering a grid based structure and concatenated LBPH matching, structural information can be used to find the best candidate. Unlike in face domain, at object level structure may not be a reliable feature for recognition but the object level structure can be a good feature to measure the correlation between two structures in adjacent frames as the structure of an object is not expected to change drastically in adjacent frames of a video.

Hence, a grid based LBPHs are used to define a distance measure in this tracking task. The same grid structure described in the section 5.2.1.1 is used here as well. Unlike during the appearance based matching, the concatenated LBPH of the detection with the most recent concatenated LBPH of the track is matched without using further history. The structure based distance measure between i^{th} detection and j^{th} track $str(i, j)$ is as defined in (5.6).

$$str(i, j) = 1.0 - \frac{\sum_m corr(LBPH_{m,i}, LBPH_{m,j})}{m} \quad (5.6)$$

where $corr(LBPH_{m,i}, LBPH_{m,j})$ is the correlation between the LBPH of the m^{th} grid cell of the i^{th} detection $LBPH_{m,i}$ and m^{th} grid cell of the j^{th} track $LBPH_{m,j}$.

Figure 5.4: Linear Binary Pattern (LBP). (a) Gray scale image. (b) Zoom-in view of the area of the red box of (a). (c) LBP of (b). (d) LBP of full image (a).

5.2.1.3 Motion based Distance

When tracking an object in adjacent frames, its motion helps to predict the position of the object in the next frame. The difference between the predicted position $pred(x_{bb}, y_{bb})$ and the detected position $det(x_{bb}, y_{bb})$ is defined as the motion based distance measure as in (5.7). L2 norm is used. x_{bb} and y_{bb} are the center positions of the object bounding box expressed in 2D image coordinates. n_{dst} is the value used to normalize the distance measure.

$$loc(i, j) = \min \left(1, \frac{\| det(x_{bb}, y_{bb}) - pred(x_{bb}, y_{bb}) \|_2}{n_{dst}} \right) \quad (5.7)$$

Predicted position $pred(x_{bb}, y_{bb})$ is calculated based on the object bounding box information from its track history. The predictions for the bounding box center (x_{bb}, y_{bb}) , width w_{bb} and height h_{bb} is generated using a weighted average of it's previous five frames information. Five frames were used because it is large enough to get good average for predictions and small enough not to drift because of old information. Weights were used to give a preference to the information from the most recent track assignments in the history. Predicted bounding box bb_j^p of the j_{th} object in existing tracks is calculated based on (5.8) below, where t is number of frames the j^{th} track is matched with a detection (i.e. bb_j^t bounding box contains the most recent matched detection bounding information) and x_{bb}, y_{bb}, w_{bb} and h_{bb} are bounding box parameters. q is the consecutive number of frames that track is undetected.

$$bb_j^p = \begin{cases} x_{bb_j^t} + \Delta_x(1 + q); \\ y_{bb_j^t} + \Delta_y(1 + q); \\ w_{bb_j^t} + \Delta_w(1 + q); \\ h_{bb_j^t} + \Delta_h(1 + q); \end{cases} \quad (5.8)$$

where $\Delta_z : z \in [x, y, w, h]$ is defined as in (5.9).

$$\Delta_z = \frac{\sum_{a=2}^5 [(a-1)(z_{t+a-5} - z_{t+a-6})]}{\sum_{a=2}^5 (a-1)} \quad (5.9)$$

Predicted position $pred(x_{bb}, y_{bb})$ is defined as in (5.10) and width of the most recent bounding box $w_{bb_j^t}$ is used as n_{dst} .

$$pred(x_{bb}, y_{bb}) = \begin{cases} (x, y)_{bb_j^t}; & \text{if } t < 5 \\ (x, y)_{bb_j^p}; & \text{otherwise} \end{cases} \quad (5.10)$$

5.2.1.4 Size based Distance

Size based dissimilarity is calculated based on the object localization information (object bounding box) of each detection compared to the predicted localization information from object track history. Specifically, intersection over union (IoU) between the detection bounding box $bb_{i,det}$ of the i_{th} object in current detection and the predicted bounding box $bb_{j,pred}$ of the j_{th} object in existing tracks is considered.

IoU is inversely proportional to the dissimilarity between two bounding boxes and the IoU will directly provide the similarity value normalized to 0 – 1. Therefore, size based dissimilarity is defined below as in (5.11).

$$iou(i, j) = 1 - \frac{bb_{i,det} \cap bb_{j,pred}}{bb_{i,det} \cup bb_{j,pred}} \quad (5.11)$$

IoU is measured considering both the location and size of the bounding boxes in comparison, thus is a good measure of the location similarity of bounding boxes. Predicted bounding box $bb_{j,pred}$ for the j^{th} track is as below in (5.12), where bb_j^t and bb_j^p is as in (5.8).

$$bb_{j,pred} = \begin{cases} bb_j^t; & \text{if } t < 5 \\ bb_j^p; & \text{otherwise} \end{cases} \quad (5.12)$$

5.2.2 Track Match and Update

Once all the four dissimilarity measurements are calculated, overall dissimilarity values $MC(i, j)$ are calculated as in (5.1) for each i^{th} detection in the current frame against each j^{th} track in the memory. Therefore, this dissimilarity matrix is used in the Hungarian algorithm [118] to assign each of the object detection in the current frame to the best matching track. However, this association is done only if the dissimilarity between the matched detection and track is below a certain threshold th_{MC} , otherwise the detected object is initiated as a new track after tracking for false negatives detailed in the section 5.2.3.

In the detection and track association stage, object information in the list of tracks is updated with the current detection's localization co-ordinates, LBPHs and detection confidence. Appearance based histograms and occlusion information are updated in a different manner. For each track three most relevant concatenated appearance histogram and occlusion information are stored. Firstly, from the already existing three most relevant concatenated histograms the most occluded one is removed. If the level of occlusion is similar, the oldest concatenated histogram is removed. Then the current matched detection's concatenated appearance histogram and occlusion information is stored for the track.

5.2.3 Tracking False Negatives for Stable Tracks

One of the applications of a tracking algorithm following tracking-by-detection framework is to identify and maintain the track of an object that goes undetected in some frame/frames as a result of a false negative of the detection algorithm, of a stable track. Fig. 5.5 visualize an example on such missed detection successfully addressed by the proposed method.

Figure 5.5: Maintaining the track in case of a false negative in detection. (a) object with ID 0 is undetected in adjacent frames ($t+1$ and $t+2$) and (b) track ID maintained based on the track history. Track ID is magnified for better visualization. Orange boxes indicate the exit regions for track ID 0.

Here, a stable track is defined as a track that has been tracked in at least five previous frames, as it is the number of frames required to predict the localization information for a track based on (5.8) and (5.9). Tracking for false negatives is initiated after the track matching and updating step described in the section 5.2.2 but before initiating new tracks for unmatched detections and invalidating tracks as shown in the flow chart in Fig. 5.1.

After the Hungarian assignment there are three outputs, (a) i^{th} detection matched with j^{th} track, (b) unmatched detection and (c) unmatched track. For each of these unmatched tracks size based dissimilarity measure is calculated against all the unmatched detections, according to (5.11) where the predicted bounding box $bb_{j,pred}$ for the unmatched track is bb_j^p as per the (5.8) and (5.9). If none of the above, size based dissimilarity measures is not less than a threshold

th_{iou} (0.5 is used as the threshold in this study during the experiments), the appearance based dissimilarity is calculated according to (5.2) with a new detection defined using the predicted bounding box $bb_{j,pred}$ for that track. If the appearance based dissimilarity is less than a threshold th_{app} (0.6 is used as the threshold in this study during the experiments) the predicted bounding box $bb_{j,pred}$ for that track is considered as its current frame match. However, if the predicted bounding box is out of the frame the track is immediately marked as an invalid track. Additionally if the predicted bounding box is fully within the exit regions as indicated in Fig. 5.5, it is not considered as a match. Width of the exit regions are dynamically defined for each track as the width of the track's last matched bounding box width.

In case of at least one detection has a dissimilarity measure less than th_{iou} these tracks and detections are considered in Hungarian assignment and matched with each other. Association is done only if the dissimilarity between the matched detection and track is below the same threshold th_{iou} , otherwise the detected object is initiated as a new track.

After updating the track list with the current matched detection data and tracking for false negatives, the track list is updated to invalidate certain tracks which have not being tracked (i.e. matching object not found) for a predefined consecutive number of frames.

5.3 Results and Discussion

The proposed tracking framework is implemented using C++ and all the test cases presented in this section is done in a desktop with Intel Xeon(R) CPU E5-1630 v3 with 3.70 GHz 8 processors. Even though there are 8 processors, code is not optimized for parallel processing.

MOT16 [141] is a popular benchmark for human tracking while KITTI [125] vision benchmark suite is a famous benchmark for autonomous driving based applications. The proposed tracking methodology is tested in both these data-sets. MOT17¹ includes all sequences of MOT16 with a new, more accurate ground truth with three sets of detections for each sequence. Hence, MOT17 benchmark is used for human tracking evaluation. MOT17 tracking data-set consists of 21 training

¹MOT17: <https://motchallenge.net/data/MOT17/>

sequences and 21 test sequences, where in each of these sets, 2 sequences consist of images with 640×480 resolution while the remaining 19 sequences contain images of 1920×1080 resolution. MOT17 training set has a total of 15,948 frames with an average of 21.1 detections per frame, while the test set contain a total of 17,757 frames with an average of 31.8 detections per frame. KITTI tracking data-set consists of 21 training sequences and 29 test sequences, where 31 sequences contains images of size 1242×375 while other sequences contain images of similar resolution (i.e. $12xx \times 37x$ resolutions). KITTI training set has a total of 8,008 frames with an average of 3.8 detections per frame and the test set contain a total of 11,095 frames with an average of 3.5 detections per frame. Since the proposed tracking method does not require any training on tracking data, both training and test data-sets have been used as test beds in experiments. The evaluation codes by MOT17 and KITTI benchmarks were used, which are based on CLEARMOT [142] and Mostly Tracked (MT: % of ground truth trajectories which are covered by tracker output for more than 80% in length) – Mostly Lost (ML: % of ground truth trajectories which are covered by tracker output for less than 20% in length) [143] metrics. CLEARMOT [142] includes total Identity Switches (IDS: The total of number of times that a tracked trajectory changes its matched ground truth identity), Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP) and average run-time per frame excluding detection time (Time). Here, $MOTA = 1 - \frac{\sum_t (fn_t + fp_t + IDS_t)}{\sum_t g_t}$ and $MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}$, where t is the frame t , fn is false negatives, fp is false positives, g is ground truth detections, c_t is correct matches found at frame t and d_t^i is the distance between predicted detection and ground truth detection for each correct match which is taken as the intersection of union between the two bounding boxes.

5.3.1 Detections

In the proposed method, tracking-by-detection framework is followed. For the MOT17 data, the benchmark has made detections available for each sequence. In fact it has made available detections from three human detectors namely, DPM [144], F-RCNN [59] and SDP [145]. In total there are 14 unique sequences and with three detection results for each make it 42 sequences. KITTI benchmark

does not provide any detection results. Therefore, bounding box results from RRC detector [109] were used.

Bonding boxes from DPM [144] human detector has higher number of false positives in general compared to other detectors. Therefore, as a pre-processing step a filtering criteria is proposed for DPM based detections by calculating a threshold for detection confidence for each sequence. At each frame, overlapping detection were searched. If the overlap is greater than 50% of the detection, the detection confidence of the detection in consideration is compared against the most overlapped detection's confidence and invalidate the bonding box with the lower confidence. This is repeated for the whole sequence and filter some detections. Then based on these invalidated detection list, a threshold for detection confidence is defined as the average of all these invalidated detection confidences. This is done for each sequence separately because of the large differences in sequences in terms of view-point, dynamic & static camera, illuminations differences and etc. Then all the detections which has a lower confidence than the threshold is marked as invalid and not used in tracking.

5.3.2 Ablation Study and Parameters Tuning

Ablation study is conducted to understand the contribution from each component defined in (5.1) and the contribution from tracking false negative part described in section 5.2.3.

All these experiments were conducted in MOT17 and KITTI training data. First, the contribution from each component in (5.1) is evaluated where the results are presented in Table 5.1 for tracking accuracy when the tracks and detections matching is done only using each component in rows *app*, *str*, *loc* and *iou*. When comparing MOTA values it is evident that each component contribute almost equally for tracking accuracy. Therefore, the weights in (5.1) are defined to be equal, i.e. $w_{app} = w_{str} = w_{loc} = w_{iou} = 0.25$. Hence, $MC(i, j)$ in (5.1) is constrained between 0 and 1.

Performance based on combining different components are analyzed. Since the main feature is based on appearance, combinations that include *app* component are analyzed. Results in the *MC* row is when all the four components are combined

Table 5.1: Ablation Study in MOT17 and KITTI Tracking Training Sets

	MOTA	MOTP	IDS	MT	ML
MOT17					
<i>app</i>	48.3%	83.4%	4,512	22.8%	32.5%
<i>str</i>	48.2%	83.4%	4,550	22.6%	32.5%
<i>loc</i>	48.3%	83.4%	4,295	22.5%	32.5%
<i>iou</i>	48.3%	83.4%	4,439	22.6%	32.5%
<i>app + str</i>	48.5%	83.4%	3,737	22.6%	32.5%
<i>app + loc</i>	48.7%	83.4%	3,061	22.5%	32.5%
<i>app + iou</i>	48.7%	83.4%	3,015	22.6%	32.5%
<i>app + str + loc</i>	48.6%	83.4%	3,194	22.6%	32.4%
<i>app + iou + loc</i>	48.6%	83.4%	3,235	22.5%	32.4%
<i>app + iou + str</i>	48.7%	83.4%	3,150	22.6%	32.5%
<i>MC</i>	48.7%	83.4%	2,905	22.6%	32.5%
MASS	49.5%	83.1%	2,808	23.8%	31.3%
KITTI					
<i>app</i>	88.3%	89.9%	994	87.8%	1.6%
<i>str</i>	91.7%	89.9%	167	87.8%	1.6%
<i>loc</i>	91.0%	89.9%	326	87.8%	1.6%
<i>iou</i>	91.2%	89.9%	299	87.8%	1.6%
<i>app + str</i>	91.4%	89.9%	234	87.8%	1.6%
<i>app + loc</i>	91.6%	89.9%	181	87.8%	1.6%
<i>app + iou</i>	91.7%	89.9%	180	87.8%	1.6%
<i>app + str + loc</i>	91.9%	89.9%	124	87.8%	1.6%
<i>app + iou + loc</i>	91.5%	89.8%	212	87.8%	1.6%
<i>app + iou + str</i>	91.9%	89.9%	118	87.8%	1.6%
<i>MC</i>	91.9%	89.9%	113	87.8%	1.6%
MASS	92.0%	89.8%	113	88.3%	1.6%

according to (5.1) (i.e. $app + str + loc + iou$). Even though the MOTA is similar to the nearest decimal, IDS is lowest when all the four components are combined. The MASS row shows the results when ‘*tracking false negatives for stable tracks*’ is included on top of MC . The MOTA is increased approximately by 0.8 % when ‘*false negative tracking for stable tracks*’ is incorporated for the MOT17 data. Unlike the human detectors in MOT17, false negatives from the the car detector is very low, hence the improvement for KITTI data is not that considerable by incorporating false negative tracking. When considering the results of each individual components for KITTI data, it can be seen that appearance feature app based tracking performance is low compared to other individual components due to higher identity switches. When tracking humans, appearance is an important factor since most of the time the clothes have different colors and combination of colours is even more differentiating (i.e. two persons may be wearing black and white colour clothes, but one may be using black for the top and the other for the bottom, which can be used to differentiate them). Since the proposed appearance matching is based on grid based colour information, different colour combinations are captured well. But in the case of cars, they have the same colour everywhere which makes the proposed appearance based tracking method has a higher probability for error compared to human tracking.

In the proposed tracking framework, after the Hungarian assignment, detection is assigned to the matched track if the matching cost $MC(i, j)$ is below a certain threshold th_{dis} . The graph in the Fig. 5.6 shows how MOTA varies with changing th_{dis} for MOT17 and KITTI tracking training data. Based on this, $th_{dis} = 0.75$ is selected for the experiments. Furthermore, number of grid cells m in the appearance feature matching introduced in section 5.2.1.1 has an impact on the accuracy of the proposed tracking framework. It is visualized in the same graph and $m = 12$ was selected as the configuration in this work. Grid cell division is defined based on the object class. Human object has a greater height than the width where as for car object it is the opposite. Therefore, when dividing $m = 12$ grid cells, (3×4) is assigned for human class and (4×3) is assigned for car class. The HSV colour space is used for colour histogram where 15 bins for hue (H) and 16 bins for saturation (S) is used. The average processing time per frame is 35 ms for MOT17 tracking

training data and 5 ms for KITTI tracking training data.

Figure 5.6: Comparison of MOTA changes for different dissimilarity threshold th_{MC} and number of grid cells m . (a) MOT17 Training Data and (b) KITTI Training Data. Blue colour represent MOTA and red colour represent ID Switches. MOTA and IDS values based on dissimilarity threshold th_{MC} are in dotted lines while solid lines are based on number of grid cells m

5.3.3 Comparison with Others

In this section the proposed tracking method is compared with other state-of-the-art methods listed in the two benchmarks. Comparison is made between the online methods that are published. Summary of the performance of the selected trackers are as shown in Table 5.2. The proposed method is compared with the best tracking frameworks in the KITTI and MOT17 benchmarks that are published and are vision based online multiple object tracking methods. Tracking performance with the test data sets are compared in the benchmarks. Test data is more challenging than the train data in both benchmarks, which is why there is a performance drop compared to results on training data stated in Table 5.1.

The proposed method is ranked top in the KITTI benchmark for online tracking methods that are published and third place among all online methods, in terms of MOTA. MOTBeyondPixels [91], reports that it uses inputs from two object detectors and is the next best to the proposed method. In KITTI benchmark the lowest amount that can be entered for time is 0.01 s. In 3D-CNN/PMBM [116] authors record a running time of 73 fps for the test set while the proposed method can achieve 150+ fps. However, it should be noted that 3D-CNN/PMBM [116] is

Table 5.2: Results in MOT17 and KITTI Tracking Test Sets as displayed in the benchmark online (accessed in April 2019)

	MOTA	MOTP	MT	ML	IDS	Time
MOT 17						
MOTDT17 [111] 3GHz, GTX1060 (Caffe)	50.9	76.6	17.5	35.7	2,474	0.054s
MTDF17 [114] 3.5GHz, GTX1060	49.6	75.5	18.9	33.1	5,567	0.83s
HAM_SADF17 [101] 3.7GHz, TITAN X	48.3	77.2	17.1	41.7	1,871	0.2s
DMAN [85] 2.4GHz (Matlab+Tensorflow)	48.2	75.7	19.3	38.3	2,194	3.33s
AM_ADM17 [102] 3.4GHz	48.1	76.7	13.4	39.7	2,214	0.175s
PHD_GSDL17 [115] 3.5GHz (Matlab)	48.0	77.2	17.1	35.6	3,998	0.149s
MASS (proposed) 3.7GHz (C++)	46.9	76.1	16.9	36.3	4,478	0.058s
KITTI						
MOTBeyondPixels [91] 2.5GHz (C/C++)	84.24	85.73	73.23	2.77	468	0.3s
IMMDP [99] 3.5GHz (Matlab + C/C++)	83.04	82.74	60.62	11.38	172	0.19s
3D-CNN/PMBM [116] 3GHz (Matlab)	80.39	81.26	62.77	6.15	121	0.01s
extraCK [98] 2.5GHz (Python)	79.99	82.46	62.15	5.54	343	0.03s
MASS (proposed) 3.7 GHz (C/C++)	85.04	85.53	74.31	2.77	301	0.01s

based on Matlab.

In MOT17, out of all the published online tracking methods, the proposed tracker is ranked in seventh place. However, it recorded the second best running speed and is nearly three times the speed of the third best method (PHD_GSDL17 [115]) in terms of the speed. Out of the best six methods, HAM_SADF17 [101] and DMAN [85] pair-wise cost calculations are based on deeply leaned neural networks while MOTDT17 [111] appearance cost is based on deeply learned person re-identification cost. MTDF17 [114] uses detection information from two object detectors in its detection stage. Compared to AM_ADM17 [102] and PHD_GSDL17 [115] the proposed method has lower MOTA but higher running speeds. The proposed method is written using C++ as it was to be implemented in an embedded system for a real time application. Among these methods, three (MOTDT17, HAM_SADF17, DMAN) are based on deeply learnt appearance features that mainly rely on GPU based parallel processing, while the proposed method is on CPU. Therefore, comparing resource utilization in this scenario will be less meaningful as two different types of hardware are used. The other three methods (MTDF, AM_ADM17, PHD_GSDL17) are classical methods such as the proposed method, making use of CPU resources. However, authors of these studies have not made their codes publicly available to perform such resource utilization analysis. The best comparison can be made using the average time recorded in the benchmark for running each tracking methods. As recorded, these three methods use CPUs with similar clock speeds with the CPU used during the experiments in this study (3.4 GHz – 3.7 GHz). However, coding language has only been recorded as MATLAB for PHD_GSDL17 method. The proposed method achieves, by far the best average processing time, being nearly three times better than the best timing (0.058s vs. 0.149s) among these three classical methods.

The proposed method achieves a good compromise between the speed and the accuracy, making it suitable to be used in real time robotics application with no requirement for specialized hardware for pedestrian tracking. Tracking results on few random frames is visualized in Fig. 5.7, where you can see the track ID is consistent across the frames.

The running time of the proposed method depend on the complexity of the

Figure 5.7: Tracking results on few frames. (a) human tracking results from MOT17 training data and (b) car tracking results from KITTI training data. At each sequence four tracks are highlighted using yellow, green, blue and green colours. These identities are uniquely tracked for more than 60 frames.

scene rather than the frame size. MOT17 training data-set has an average of 21.1 detections per frame, while the test set has an average of 31.8 detections per frame. Running time for MOT17 training set is 35 ms while for test set it is 55 ms per frame. This reflects the average detections per frame in each set. Compared to MOT17, KITTI [125] training and test data has similar average detections per frame (3.5 for test set and 3.8 for training set) and result in similar running speeds (5 ms per frame). When comparing MOT17 and KITTI, running speeds are higher in KITTI data because the average number of detections per frame is lower than that of MOT17. The proposed tracking algorithm depends on number of detections in the current frame plus the number of tracks in the track list, which justify the observed behaviour.

Tracking performance of the methods following tracking-by-detection framework is affected by the detection method used. In Table 5.3, tracking performance of the proposed method and the best two methods in Table 5.2 is presented considering the third sequence of the MOT17 benchmark. All the three methods in Table 5.3 perform best when SDP [145] detection results are used, followed by when F-

Table 5.3: Comparison of Results on Different Detection Methods on MOT17-3rd sequence

	MOTA	MOTP	IDS
MOT17-03 DPM [144]			
MOTDT17 [111]	55.5 %	71.3 %	364
MTDF17 [114]	56.8 %	72.9 %	992
MASS (proposed)	45.1 %	74.7 %	602
MOT17-03 F-RCNN [59]			
MOTDT17 [111]	58.7 %	77.3 %	220
MTDF17 [114]	59.9 %	76.2 %	683
MASS (proposed)	58.0 %	77.6 %	347
MOT17-03 SDP [145]			
MOTDT17 [111]	73.2 %	78.5 %	361
MTDF17 [114]	72.5 %	78.3 %	728
MASS (proposed)	73.6 %	77.1 %	636

RCNN [59] and DPM [144] detection results are used. When SDP [145] is used as the detector the proposed method achieve the best MOTA. The proposed method performs competitively when F-RCNN [59] is used as the detector but show poor results when DPM [144] detection results are used in comparison with other two methods. As explained in section 5.3.1 DPM detections contain large number of false positives and method does not handle these as well as other methods.

5.4 Concluding Remarks

In this chapter, an efficient frame work for multiple object tracking is proposed, where it can achieve 150+ fps for KITTI data and 27.0 fps and 17.1 fps for MOT17 training and test data respectively. Thus, the proposed method is highly suitable for real time applications and the results show that the proposed method achieve state-of-the-art performance for both human tracking and car tracking. The proposed tracker is based on combining grid based colour histogram matching, grid based LBPHs matching, predicted object motion matching and predicted size based matching. Furthermore, in this work, false negative tracking is introduced to compensate errors from the object detector. Better detection lead to better tracking results, especially when following tracking-by-detection architecture where some work [109, 110] combine detection results from multiple detectors to achieve this.

The proposed method show very competitive tracking results when there are less number of false positives in detections. Furthermore, when comparing the results in Table 5.2, IDS are higher in the proposed method. Therefore, including a false positive removal mechanism as well as post processing method to reduce IDS can be future directions to further improve the tracking accuracy.

Chapter 6

Learning Features for Multiple Object Tracking as an Extension to Detection

6.1 Introduction

This chapter introduces a 3D convolutional extension for an existing detection network, to learn good appearance features suitable for tracking. In the previous chapter, appearance features are extracted as grid based histograms using a traditional approach. In this study, the attempt is to improve the appearance feature extraction process using deep learning at a minimum effect to the computational complexity.

Objective of Multiple Object Tracking (MOT) is to assign a unique track identity to each object present in a sequence of frames. In contrast, the objective of single object tracking is to correctly track the size and appearances variances of an object specified in the first frame of the sequence, in the rest of the frames. MOT problem has been addressed by using multiple instances of single object trackers or following tracking-by-detection framework. The former approach is to initiate a single object tracker at the first frame for each object and track these objects across the rest of the frames. The objects are detected in the first frame using an object detector. However, new objects can appear in other frames as well. This is handled by running an object detector in all the frames or few frames apart (based on the computational power and the tracking speed required for the application at hand). Tracking by detection framework is based on the assumption that at each frame localization information (bounding box) for all the objects in that frame is provided (by an object detector). In both the methods, an object detector is required to initiate the tracking process.

Running a good object detector [60] is time consuming compared to tracking an object, hence robotics applications that suffer from low processing power (such as unmanned aerial vehicles) favours the former method. However, the required computational power may increase when the number of objects to track increases

beyond a certain limit, due to the overhead from multiple instances of single object trackers. Some object detectors [62, 63] can run in real time with high-mid range hardware (which can be used in ground robots) and generate satisfactory results. Such applications follow tracking-by-detection framework.

This study is focused on solving the MOT challenge using tracking-by-detection framework. In most of the methods, once an object detector outputs all the detected objects, tracker learns some feature representations that will help it to calculate a similarity score to match similar objects with the objects detected in subsequent frames. Such feature representations contain appearance based features, motion based features, size or location based features and different variants of above. Most methods combine several feature representations to improve the tracking result. As such, the tracker waits for the output from the object detector to initiate its feature learning task. Therefore, by re-using feature information from detection task for the tracking feature learning task will be helpful in reducing the computational complexity of the overall framework. Almost all of the object detectors used in current research are based on deep learning architecture. Such detectors rely on deeply learnt appearance features of the image to classify and localize different objects belonging to different classes. These deep features can directly be used to aid the tracking task as appearance features. An object detector learns features to differentiate inter-class objects while a tracker should be able to differentiate both intra-class and inter-class objects. Because of this reason many object trackers use a separate deep network to calculate a matching score between objects in adjacent frames; which is an additional computational cost for the overall system. Therefore, this study proposes to learn features suitable for the tracking task as an extension to the object detector used. This adds a minimal complexity to overall framework and allows to track multiple objects at satisfactory speeds. The proposed framework achieves speeds up to 13.9 frames per seconds for the whole pipeline (i.e. detector plus tracker), for MOT16 [141] training data.

In the next section a detailed description of the proposed method is presented followed by a discussion on the implementation details and comparison with state-of-the-art methods, in the third section.

6.2 The Proposed Method

In this work, appearance features suitable for tracking are learnt as an extension to a detection network. 3D convolution architecture is used to capture the relations between adjacent frames. Network is trained using a variation of triplet loss, defined in a fractional manner. During inference, the features suitable for tracking and detection can both be obtained at once. The architecture of the proposed network is as shown in the Fig. 6.1. In addition to the base network, a tracking head is introduced to generate features that can differentiate objects. Furthermore, multiple frames are concatenated to exploit the temporal relations using 3D convolutional layers. Thus, the need to have a separate network to extract appearance features is eliminated. Furthermore, the proposed improvement can be easily integrated into any other detection network as well.

Figure 6.1: Network architecture of the proposed method. Detection head and Base network weights (denoted by blue colour) are shared and consists of 2D convolutions. 3D convolution parts indicated in the orange colour are the newly introduced layers for the network.

2D convolution tries to find the relations in a local kernel. However, in tracking, relations between frames needs to be analyzed. 3D convolution among adjacent frames helps in understanding these relations in a local window. 3D convolution kernel has three dimensions in which third dimension is the time. 2D convolution kernel is convolved with images along the width and height of the image. In addition, in a 3D kernel third dimension is present which is in the time direction, i.e. adjacent frames. Exploiting such relations is beneficial in the tracking task rather than relying on features from 2D convolution in a single frame. However, 3D convolution consumes a lot of memory, specially when the network is undergoing training. Furthermore, depending on the temporal stride and the number

of adjacent frames used in computation, memory consumption can get increased significantly. This makes it impossible to use a deep network architecture that only uses 3D convolutional layers because of the memory limitations in currently available GPUs (Graphical Processing Units) [119]. Therefore, in this work 2D convolution based base network is used to extract features which is deep enough to extract meaningful features from images. Such features from several adjacent frames are concatenated together and are then used in two layer 3D convolution structure to extract features that are useful to track.

6.2.1 Network Architecture

The proposed extension can be applied to any deep learning based detection network. As shown in Fig. 6.1, part of the network in blue colour are from the detection module. In the proposed network architecture, 3D convolutional layers are inserted into the network after the final feature layer of the detection module and before the detection head. Tracking is a task that spans multiple frames unlike the detection task. Therefore, integrating results from multiple frames into the network could improve the tracking accuracy. Hence, 3D convolutional layers are introduced in the proposed network architecture. However, initial feature extraction from the images is done by executing the base network with 2D convolution layers, separately for each frame. It could be beneficial to improve the base network with 3D convolutional layers but the challenge is the size of the network, which would be too big for the hardware that is available today, especially during the training phase [119]. Hence, base of the detection network is used to extract features from each frame in the input sequence and these features are concatenated and used as the input to the 3D network.

The base network with 2D convolution, shown in Fig. 6.1, could be any detection network. In this work, the base network used is the CenterNet DLA-34 based detection network proposed in [67], to prove the proposed idea. In [67], object detection is implemented as heatmap based single stage detection. Object detection results for [67] shows that it outperforms all single stage object detectors. As shown in [67], DLA-34 variation gives the best speed/accuracy trade-off. The node level architecture of the CenterNet DLA-34 base network is as detailed in Fig. 6.2.

Image inserted to the network is of size 512×512 colour image, so the feature vector is $(3, 512, 512)$. Feature vector output from the network is $(64, 128, 128)$. Width and height of the image is reduced four times in both dimensions.

Figure 6.2: Network architecture of CenterNet DLA-34 (Base network). I is an image of size 512×512 . In *Node Structure*, I_{in} is the input image size to the Node and I_{out} is the output image size from the node. C_{in} is the input number of channels and C_{out} is the output number of channels for the node.

In the proposed work, five frame blocks are used as an input to the network. Details of the proposed 3D network layers are as presented in Fig. 6.3. The 3D convolution layers are defined to retain the size of the input such that input shape is $(N_b, 64, 5, 128, 128)$ where N_b is the batch size, 64 is the input channels, 5 is the depth and 128×128 is the height and the width. In a similar notation output shape

is kept at $(N_b, 64, 5, 128, 128)$ using kernel size $(3 \times 3 \times 3)$ with $(1 \times 1 \times 1)$ padding and stride of $(1 \times 1 \times 1)$ in the three directions (depth, height and width). The output features are separated to five frames after the convolutions. Features after $1 \times 3D$ module are used as the input to the detection head and the features from $2 \times 3D$ module are used for tracking purposes.

Figure 6.3: Details of the 3D Network

6.2.2 Network Training

After inserting the 3D network blocks into the network, the network could be trained through the detection head in the first step as an object detector. The same loss functions as of the detection network is used for this purpose. Normally the detection head may consists of the class score and bounding box regression blocks. Since the input to the detection head is five frames (the resulting features after the $1 \times 3D$ convolution block), it can be split into five frames (as indicated in Fig. 6.1). Then the network can be trained similar to the end-to-end object detection network. Any large scale object detection database can be used for the training, loss propagating via the detection head. After the network is trained for the detection path, the tracking head, which consists of the $2 \times 3D$ convolutional block, needs to be trained. In order to train the tracking head, and extract robust feature representations for tracking object over time, database that contain ground truth information regarding objects and their identities is required. If the detection path is trained with a larger database other than the benchmark data used for tracking, its weights should be fine-tuned with detection ground truth from tracking database. The loss function used for tracking path is according to (6.1) where fm and fn are the feature vectors extracted from the $2 \times 3D$ convolution unit

for objects in two adjacent frames. The purpose of the loss function is to make the network learn the weights required to generate similar features vectors for the same object and larger distinct features for different objects in adjacent frames. When the similarity and dissimilarity are joined in a fractional manner as in (6.2), small dissimilarity will give rise to a large loss, forcing the network to learn large dissimilarity between different objects.

$$loss = \max \left\{ tr(fm, fn), 0 \right\}; \quad (6.1)$$

where;

$$tr(fm, fn) = \frac{sim(fm, fn) + \delta}{dis(fm, fn) + \delta} \quad (6.2)$$

In (6.2), δ is to denote a small constant. $sim(fm, fn)$ is the similarity between feature vectors fm and fn in adjacent frames, $dis(fm, fn)$ is the dis-similarity between feature vectors fm and fn in adjacent frames as defined in (6.3) and (6.4) respectively. Given an object, during the learning phase, it has it's own instance and set of other objects in an adjacent frame. Hence, similarity will be from comparing two objects with the same identity. However, for dissimilarity there are multiple values arising from multiple objects. Since the objective is to separate the feature space as far as possible for two different objects, the smallest dissimilarity is selected as the dissimilarity cost.

$$sim(fm, fn) = \sum_{i=\pm 1} \sum_{fm_d^t} D(fm_d^t, fn_d^{t-i})_{id_{fm_d^t}=id_{fn_d^{t-i}}} \quad (6.3)$$

$$dis(fm, fn) = \sum_{i=\pm 1} \sum_{fm_d^t} \min_{fn_d^{t-i}} D(fm_d^t, fn_d^{t-i})_{id_{fm_d^t} \neq id_{fn_d^{t-i}}} \quad (6.4)$$

The proposed loss function (6.1) is performed at the middle frame (objects at t frame compared to objects at $t - 1$ and $t + 1$ frames according to the notation at Fig. 6.1). In both (6.3) and (6.4), fm_d^t is the objects present in the t^{th} frame and fn_d^{t-i} are the objects in $(t - 1)^{th}$ and $(t + 1)^{th}$ frames. In (6.3), feature distance D between the same object (having the same identity id) in two adjacent frames is calculated while in (6.4), feature distance between two different objects in two adjacent frames is calculated. Specifically, frame t is compared with $(t + 1)$ and

$(t - 1)$ when calculating similarity and dis-similarity measures. Sum of absolute error is considered as the distance measure D as shown in (6.5), where n_f is the length of the feature vector.

$$D(fm, fn) = \sum_{n_f} |fm_{n_f} - fn_{n_f}| \quad (6.5)$$

During training the network, fm and fn feature vectors need to be obtained from the network. Features output at the end of $2 \times 3D$ convolution path has a shape of $(N_b, 5, 64, 128, 128)$. Firstly, for each of the N_b batches, the features are separated to five features of shape $(64, 128, 128)$. Then according to ground truth bounding boxes in the five frames scaled to 128×128 are used to extract features for each object. However, these bounding boxes have different sizes and hence the resulting feature vectors will be of different lengths, which makes them unusable to be used in calculating feature distances based on (6.5). Therefore, to overcome this challenge, two strategies are proposed. First alternative is to reshape the extracted features to a fixed size of $(C \times C)$ using bi-linear interpolation. Second alternative is to use an ROI Align layer proposed in Mask R-CNN [60] to extract features inside the bounding box to a fixed $(C \times C)$ length feature vectors. Performance based on these two methods is discussed in the following section.

6.2.3 Network Inference

During the inference time, features from both the detection path and tracking path (i.e. convolution features after $1 \times 3D$ convolution block and $2 \times 3D$ convolution block according to Fig. 6.1) are concatenated and used in feature matching. Features are extracted according to the bounding boxes detected by the detection head. Features are extracted following either the reshaping mechanism using bi-linear interpolation or ROI Align mechanism. Blocks of five image frames are input to the network. At each inference, there are four possible strategies to process the input frames, as shown in Fig. 6.4.

6.2.4 Tracking

After extracting the object's bounding boxes, class scores, and their corresponding feature representations from the network, the post-processing step can be applied

Figure 6.4: Images inserting strategy during inference. Sequence starting frame is 1 and sequence ending frame is t_n . 2 *frames* indicates the number of frames processed for final tracking result for the five frame batch inserted to the network, where the processed frames are indicated by orange colour. The rest of the naming follows the same convention.

to match them. Here, a matching cost MC is defined according to (6.6), by combining bounding box overlap and feature similarity between i^{th} track and j^{th} object detected in each frame.

$$MC(i, j) = \begin{cases} D(i, j) + W_{iou}(1 - iou(i, j)); & \text{if } i_{cls} = j_{cls} \\ C_p; & \text{otherwise} \end{cases} \quad (6.6)$$

Here, C_p is a large penalty to indicate that these two objects has a very large dis-similarity to each other, when the classes of the objects belonging to (i_{cls} and j_{cls}) are different. $D(i, j)$ is the feature distance between i^{th} track and j^{th} object in t^{th} frame, according to the normalized (according to the feature length) distance of (6.5) and $iou(i, j)$ is the intersection over union between the most recent bounding box of i^{th} track (bb_i) and the bounding box of j^{th} object (bb_j) as defined in (6.7).

$$iou(i, j) = \frac{bb_i \cap bb_j}{bb_i \cup bb_j} \quad (6.7)$$

Then a cost matrix is constructed based on the matching costs between all the objects and tracks at each frame. This cost matrix is used in Hungarian algorithm [118] to find the optimum assignment of objects to tracks. However, the assignment is done only if the matching cost MC is below a pre-defined threshold th_{MC} . The objects that does not get matched are initiated as new tracks.

6.3 Results and Discussion

The proposed method is coded using PyTorch and Python. All the results are generated using a PC with 3.7 GHz 8 core CPU and 1080 Nvidia GPU. The proposed method can be implemented using any detection network. In this work the detection network proposed in [67] is used as it has higher accuracy than a typical shot shot detector such as Yolo [62] for the similar inference time and higher frame rate than that of regional proposal based detection network such as Faster R-CNN [59] or Mask R-CNN [60] for similar accuracy levels. Authors in [67] have used several backbone networks, out of which Deep Layer Aggregation (DLA) [146] based variant is selected.

In order to evaluate tracking performance, MOT16 [141] benchmark is selected. MOT16 [141] is a popular benchmark for human tracking and it includes seven training sequences and seven test sequences, where in each of these sets, two sequences consist of images with 640×480 resolution while the remaining five sequences contain images of 1920×1080 resolution. MOT16 training set has a total of 5,316 frames with an average of 20.8 detections per frame and 517 tracks, while the test set contain a total of 5,919 frames with an average of 30.8 detections per frame and 759 tracks. Since the ground truth for test set is not made publicly available training set is divided to train and validation sets. There are seven sequences in train set where the first 70% of the frames from each sequence are used to construct the test set while the rest is used as the validation set. Since, the validation set consists of 1,597 frames. During training, the network is initiated with the weights provided by authors in their github repository ¹. Then the network is first trained with coco detection [61] benchmark dataset for the detection path. Then it is fine-tuned using the data from MOT16 [141]. Same has been used to train the tracking path as well.

6.3.1 Comparison to Baseline

Firstly, the experiments are conducted to compare the improvement of the proposed method over using appearance features for the detection task from the base network (baseline). Therefore, the weights of the *dla_34* network proposed in [67] is fine-

¹<https://github.com/xingyizhou/CenterNet>

tuned with MOT16 [141] training data and best weight is selected. Since features are learnt during the detection task, features after the final convolution layer are extracted as the appearance features to be used during tracking. In [147, 108], authors directly use features from detection network as their appearance features. Then Hungarian assignment [118] is used frame by frame to get the tracking result. Since the objective is to check whether there's an improvement in the proposed method in term of the features learnt, matching cost MC is set to be feature based distance $D(i, j)$ in (6.6), without using the information from bounding boxes. The same criteria is applied to the proposed network architecture as well. For the proposed method two results are shown, each for the different criteria in feature extraction methods, reshape (re) and RoI Align (ra). Proposed method is termed as $MOTeD$ and the results are as presented in Table 6.1. The results are shown for validation dataset as defined during experiments. During these experiments, 2 *frames* processing strategy (as in Fig. 6.4) is used for higher accuracy unless otherwise stated.

Table 6.1: MOT Performance Compared to Baseline Method

	MOTA	MOTP	IDS	MT	ML	FPS
$Base_{re}$	56.2%	82%	244	91	58	8.7 Hz
$Base_{ra}$	56.3%	82%	202	91	58	15.4 Hz
$MOTeD_{re}^f$	57.6%	81%	238	95	53	3.8 Hz
$MOTeD_{ra}^f$	57.8%	81.1%	172	94	51	9.6 Hz

In order to evaluate the accuracy, CLEARMOT [142] and Mostly Tracked – Mostly Lost [143] (MT: % of ground truth trajectories tracked for more than 80% and ML: % of ground truth trajectories tracked for less than 20% of the ground truth track length) matrices are used. CLEARMOT [142] includes total Identity Switches (IDS: The total of number of times that a tracked trajectory changes its matched ground truth identity), Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP) and frames processed per second (FPS). FPS is for the both detection and tracking processes. The evaluation codes provided by the MOT16 benchmark is used. Results in Table 6.1 indicates that with the proposed method, all the accuracy measures have improved. Specifically,

the IDS has improved significantly as in the proposed method, network is trained to make the feature distance between the objects of similar identities in adjacent frames to become smaller while the same to increase when the identities are different. It can be seen that using RoI Align for feature extraction is better than using bi-linear interpolation based reshaping.

In this study, feature distance is calculated using the absolute error. In Table 6.2, sum of absolute error (SAE), sum of squared error (SSE) and cosine distance ($Cosine$) measures are compared for the baseline when RoI Align is used (on the validation data). Based on this, SAE is selected as the distance measure for both training and inference.

Table 6.2: Different Distance Measures for Baseline

	MOTA	MOTP	IDS	MT	ML	FPS
SAE	56.3%	82%	202	91	58	15.4 Hz
SSE	56.1%	82%	247	91	58	16.7 Hz
$Cosine$	56.2%	82%	244	91	58	11.6 Hz

MOT16 tracking benchmark contains seven sequences which are captured from different cameras with different image resolutions. Therefore, rather than using a single threshold to select valid objects from the detection network, using different thresholds for each sequence could yield in better detection results. This is evident from the results in Table 6.3, under the rows named th_{det} . Then the improvement due to including bounding box information when calculating the matching cost MC according to (6.6) is shown in rows under the name iou . The weight W_{iou} as per (6.6), is found experimentally to be 0.03 for reshape (re) and 0.02 for RoI Align (ra). Hungarian assignment finds the global minimum assignment cost, where sometimes matching cost (MC) between a track and assigned detection is too high. This can be handled using a threshold to discard an assignment based on MC value, which is presented in rows named th_{MC} in Table 6.3. Best value for th_{MC} is found to be 0.08 for RoI Align (ra) and 0.05 for reshape (re). FP and FN stands for false positives and false negatives respectively. It can be observed that all the above improvements lead to better tracking results in terms of MOTA, IDS, MT and ML.

Table 6.3: MOT Accuracy for Improvements on MOTeD

Upgrade	MOTA	MOTP	IDS	MT	ML	FP+FN
$th_{det,re}$	59.7%	81.2%	242	88	57	12,487
$th_{det,ra}$	60.1%	81.3%	163	88	56	12,454
iou_{re}	60%	81.3%	173	88	57	12,485
iou_{ra}	60.1%	81.3%	161	88	56	12,454
$th_{MC,re}$	60%	81.3%	170	88	57	12,485
$th_{MC,ra}$	60.2%	81.3%	124	88	56	12,454

In addition, a different strategy for tracking is tested. In that strategy, as the first step, Hungarian assignment is used only between objects detected and active tracks. Then, a second Hungarian assignment is used to connect the remaining objects to inactive tracks. The motivation to follow such a process is to give priority to tracks that got matched with detections in most recent frame (active tracks) over the recent unassigned tracks (inactive tracks). However, the tracking performance under these circumstances are marginally worse (about 0.2% drop in MOTA).

6.3.2 Analysis on Running Speed

Furthermore, based on the number of frames processed at each inference (as in Fig. 6.4), different results can be observed as shown in Table 6.4. When increasing the number of frames processed for tracking at each inference, frame rate is increased as expected with a slight drop in accuracy. In order to demonstrate the drop in the accuracy clearly, results in Table 6.4 are generated for the combined data from both training and validation sets. In all these instances, each image set (i.e. 5 images together) is processed in the network for about 111 ms. In the 5 *frames* instance both detection and tracking combined, can run at 13.9 frames per second, i.e. about 72 ms per frame. Out of this time, average time for each frame in the network is 32 ms, extracting features requires about 8 ms and tracking (i.e. matching cost MC calculation and Hungarian assignment) takes about 32 ms. This 32 ms required to perform tracking in each frame can be reduced significantly if the code is transferred to C++ from Python.

Table 6.4: MOT Accuracy and Processing Speed based on number of frames processed for tracking at each inference

	MOTA	MOTP	IDS	MT	ML	FPS
<i>2 frames</i>	62.3%	83%	494	186	96	8.3 Hz
<i>3 frames</i>	62.3%	83%	506	186	96	10.6 Hz
<i>4 frames</i>	62.1%	82.9%	756	189	95	12.4 Hz
<i>5 frames</i>	61.9%	82.9%	824	183	94	13.9 Hz

6.3.3 Comparison to Others

MOT16 [141] benchmark maintains two benchmarks, one for methods that use the public detections provided by the benchmark and another for methods that use their own detections. The methods in the later are of higher accuracy since they make use of better detectors. Since the method can simultaneously perform tracking and detection with a single network, results are compared with the best published methods is the later benchmark. Five frames are required to generate results in the proposed method. However, tracking is carried out frame by frame without considering the detections from all the frames. Therefore, method is compared with the online methods. Results are presented in Table 6.5 as displayed in the MOT16 website ² online. Some results are visualized in Fig. 6.5. The method used for the test dataset is the *2 frames* RoI Align method with the parameters obtained for the best method in Table 6.3.

Table 6.5: MOT Performance on MOT16 Test Data as displayed in the online benchmark (accessed in February 2020)

	MOTA	MOTP	IDS	MT	ML	FP+FN	FPS
POI [103]	66.1%	79.5%	805	34%	20.8%	60,975	9.9 Hz
CNNMTT [108]	65.2%	78.4%	946	32.4%	21.3%	62,474	11.2 Hz
RAR [100]	63%	78.8%	482	39.9%	22.1%	66,911	1.6 Hz
MOTeD	61.9%	78.3%	1,030	31.8%	25.2%	68,428	8.8 Hz
DeepS [107]	61.4%	79.1%	781	32.8%	18.2%	69,520	17.4 Hz

In the benchmark, POI [103] has obtained the best tracking accuracy in terms of MOTA criteria out of all the published online methods. Authors have used Faster R-CNN [122] as the detector which was trained on pedestrian data collected

²MOT16: <https://motchallenge.net/results/MOT16/?det=Private>

from multiple pedestrian datasets. In order to obtain the appearance feature, they train another network (input size 96x96) on multiple person re-identification data using triplet loss. In contrast, MOTeD is trained on MOT16 data and single network is used to extract appearance features as well as detections. Two stage regional proposal network based detector such as Faster R-CNN [122] achieves the state-of-the-art detection results while the heat map based methods such as the detection network used in the proposed method achieves satisfactory results with much faster processing speeds. FPS mentioned in Table 6.5 is only for the tracking part as per the benchmark requirement. In the proposed method, mentioned FPS includes time to extract features from the network as well, where tracking only can achieve 12.1 Hz and combined with detection the whole pipeline can achieve 5.1 Hz. RAR [100] method achieved the best results in terms of IDS. They used a separate recurrent auto regressive network for each track to stay updated with appearance and motion features, which makes the method slower than the others. DeepS [107] achieved very high running speed where it uses a CNN network to extract appearance features for each object, which is trained on larger person re-identification dataset. CNNMTT [108] used convolution features from the detection network itself as appearance features and additionally use ideas from social behavioural models obtain tracking results. In contrast, the proposed method simply combine appearance features and intersection over union of bounding boxes to calculate matching cost. Furthermore, the proposed idea can be integrated into detection networks used in POI, CNNMTT and DeepS to generate appearance features suitable for tracking directly from the same network and may lead to performance improvement.

Furthermore, the proposed method (*MOTeD*) in the current chapter is compared with the traditional method proposed in Chapter 5 (*MASS*). Since *MASS* method is independent of human detector, the detection results of *MOTeD* is used. This enables pure comparison between the two tracking methods. However, it should be noted that the purpose of the *MOTeD* method is to perform tracking as an extension to the deep learning based detector at a minimum incremental computational complexity and time for the tracking task, unlike the additive complexity in *MASS* method. Tracking performance comparison is as presented in

Figure 6.5: MOT results on few sequences in MOT16 data. Each row of images corresponds to different sequence and each image is 100 frames apart from each other. Bounding boxes of the same track in different frames are presented in a unique colour with a unique track identity. Some of the tracks are highlighted for better visualization that have maintained an unique identity across 200 frames.

Table 6.6. MOT16 training data is used in this comparison.

Table 6.6: MOTeD vs. MASS Performance on MOT16 Training Data

	Seq 2	Seq 4	Seq 5	Seq 9	Seq 10	Seq 11	Seq 13
IDS							
MASS	111	114	74	33	126	36	126
MOTeD	93	73	73	21	96	26	112
MOTA							
MASS	38.5%	76.8%	56.2%	53.9%	55%	67.2%	45%
MOTeD	39.2%	77%	56.4%	54.3%	56.7%	67.4%	46.5%

According to Table 6.6, for all the seven sequences in MOT16 training data *MOTeD* performs better in terms of both *IDS* and *MOTA*. The improvement of *MOTeD* is clearly evident in terms of *IDS* as *MOTeD* tracking head is trained to reduce the identity switches. Since deep learning based detection networks outperform the traditional detection approaches, it is used in almost all detection applications. However, after the detection step traditional methods such as *MASS* requires additional step to extract multiple features prior to perform matching

cost calculation and Hungarian assignment. Therefore, it is most efficient to use features from the network itself trained for the tracking task at a minimum computational cost. Furthermore, the quantitative analysis shows that the improvement in accuracy as well.

6.4 Concluding Remarks

This study proposes a 3D convolutional extension for an existing detection network, to learn good appearance features suitable for tracking. Extracting tracking features as an extension has helped to keep the incremental computational complexity at a minimum. Extension network is trained using a variation of triplet loss in a fractional manner opposed to the additive manner, used in person re-identification networks. The proposed method could be applied to any type of detection network. The proposed network is tested in MOT16 multiple object tracking benchmark for pedestrian tracking and has achieved comparable tracking results with the state-of-the-art. Proposed method can achieve 13.9 FPS and 8.8 FPS for the whole pipeline of detection and tracking, for MOT16 training and test data respectively. Moreover, the results obtained suggests that the features learnt using the extension, produce accurate tracking results compared to using the features learnt only from the detection network. Results obtained are based on the network trained only on MOT16 training data. Exploring the improvements that can be achieved using a bigger dataset remains a future work. Furthermore, final layer convolution features from the base network are used to train the 3D convolution branch for tracking. Combining features from higher levels of convolution layers is believed to increase the feature learning ability (while addressing the vanishing gradient problem). Such improvement could have the potential to improve the proposed method.

Chapter 7

Conclusion and Recommendations

This chapter concludes the report by summarizing the work that has been done during this research study and suggests few recommendations for future research.

7.1 Conclusion

The aim of this study is to provide real-time solutions for sub-tasks in vision based autonomous navigation system. Both positive & negative obstacle detection and multiple object tracking tasks are studied. Main contributions are as summarized below.

1. An efficient framework is proposed for understanding road slope, detecting class agnostic 3D obstacles points and instance level object segmentation in real time. The proposed framework has been tested with data collected in public roads in Singapore. During data collection, data is collected using four different stereo rigs covering sunny days as well as cloudy days. During data collection stereo rigs are assembled at zero degree pitch angle as well as ten degrees pitch angle. The proposed framework can operate in 22.5 frame per second on collected data.
2. A novel method of detecting negative obstacles using colour information, saliency detection and stereo output is presented. The proposed method is tested with concrete-road, tar-road and corridor based scenarios and have obtained promising results. The proposed method can achieve better results compared to the recent work. Moreover, the optimization method in the proposed work can be applied to improve the performance of the compared methods. Furthermore, using the salient nature of negative obstacles can lead to better performance compared to using the assumption that negative obstacles are darker than the surroundings.
3. Efficient framework for multiple object tracking is proposed using hand crafted

features, where it can achieve 150+ fps for KITTI data and 27.0 fps and 17.1 fps for MOT17 training and test data respectively. Thus, the proposed method is highly suitable for real time applications and the results show that the proposed method achieve state-of-the-art performance for both human tracking and car tracking. The proposed tracker is based on combining grid based colour histogram matching, grid based LBPHs matching, predicted object motion matching and predicted size based matching. Furthermore, in this work, a false negative tracking is proposed to compensate the errors from the object detector. As with all tracking-by-detection methods, tracking performance depends on the accuracy of the detection results, where the proposed method show very competitive results when there are less number of false positives in detections.

4. Introduced a 3D convolution based architecture for an existing detection network to learn appearance features for multiple object tracking tasks while carrying out the detection task as well. The proposed method could be applied to any type of detection network and could be used in both detection and MOT tasks. The proposed network is tested in MOT16 multiple object tracking benchmark for pedestrian tracking and have achieved comparable tracking results with the state-of-the-art. Proposed method can achieve 13.9 FPS for the whole pipeline of detection and tracking combined. While the proposed network is only trained on MOT16 data, exploring the improvements that can be achieved using a bigger dataset remains a future work. Furthermore, final layer convolution features from the base network are used to train the 3D convolution branch for tracking. Combining features from higher level convolution features is believed to increase the tracking feature learning ability, which has the potential to improve the proposed method.

7.2 Recommendations for future research

Based on the study, following recommendations can be made for future research.

1. Negative obstacle detection method needs to be further analyzed with different environmental conditions, such as water-filled potholes, multiple potholes

in one image and etc. Collecting data with negative obstacles on different scenarios is a challenging task, which has resulted in very few research studies conducted in this area. The most challenging reason is the unavailability of benchmark dataset for negative obstacles. To the best of our knowledge, any other work in negative obstacle detection has not compared their results with others as well due to these reasons. This challenge remains open as a future research work to facilitate more research in this area. Good dataset will help to propose learning based solutions for the negative obstacle detection task as well.

2. The state-of-the-art performance in multiple object tracking is still around 0.75 (out of 1), even for offline methods. Therefore, this area presents a lot of room for future research. One possible method to investigate is to incorporate ego-motion and estimating object's 3D world location for improving tracking. Another approach would be to use better deep learning models either with architectural change or different formulation for training loss computation for appearance based feature extraction. Furthermore, single network architecture to incorporate appearance features and location based information for online detection and tracking applications would be an interesting area of research. Exploring different detection networks modified with tracking extensions, could provide more insights on how to further improve research in this direction.

List of Publications

Journal Papers

1. **Karunasekera, Hasith**, Han Wang, and Handuo Zhang. “Energy Minimization Approach for Negative Obstacle Region Detection.” *IEEE Transactions on Vehicular Technology* 68.12 (2019): 11668-11678.
DOI: 10.1109/TVT.2019.2948684.
2. **Karunasekera, Hasith**, Han Wang, and Handuo Zhang. “Multiple Object Tracking With Attention to Appearance, Structure, Motion and Size.” *IEEE Access* 7 (2019): 104423-104434.
DOI: 10.1109/ACCESS.2019.2932301
3. Zhang, Handuo, **Hasith Karunasekera**, and Han Wang. “A Consistent and Long-term Mapping Approach for Navigation.” *International Journal of Robotics and Automation* 5 (2018): 23-31.
4. **Karunasekera, Hasith**, Mahdi Abolfazli Esfahani, Handuo Zhang, Hui Zhou, and Han Wang. “Learning Features for Multiple Object Tracking as an Extension to Detection.” *Journal of Image and Vision Computing*. (Under Review)

Conference Papers

1. **Karunasekera, Hasith**, Handuo Zhang, and Han Wang. “Real Time Multiple Object Tracking using Deep Features and Localization Information.” 2019 IEEE 15th International Conference on Control and Automation (ICCA). IEEE, 2019.
DOI: 10.1109/ICCA.2019.8899498
2. **Karunasekera, Hasith**, Handuo Zhang, Tao Xi, and Han Wang. “Stereo vision based negative obstacle detection.” 2017 13th IEEE International Conference on Control & Automation (ICCA). IEEE, 2017.
DOI: 10.1109/ICCA.2017.8003168

3. Zhang, Handuo, **Hasith Karunasekera**, Hui Zhou, and Han Wang. “GMC: Grid based motion clustering in dynamic environment.” Proceedings of SAI intelligent systems conference. Springer, Cham, 2019.
DOI: 10.1007/978-3-030-29513-4_93
4. Zhou, Hui, Handuo Zhang, **Hasith Karunasekera**, and Han Wang. “Real-time Robust Multi-lane Detection and Tracking in Challenging Urban Scenarios.” 2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM). IEEE, 2019.
DOI: 10.1109/ICARM.2019.8834317
5. Wang, Han, Rui Jiang, Handuo Zhang, **Hasith Karunasekera**, and Hui Zhou. “Stereo Visual Localization, Sensing and Perception for Fast-moving Unmanned Ground Vehicles.” In International Conference On Future Information & Communication Engineering, vol. 10, no. 1, pp. 343-347. 2018.
6. Zhang, Handuo, **Hasith Karunasekera**, and Han Wang. “A hybrid feature parametrization for improving stereo-slam consistency.” 2017 13th IEEE International Conference on Control & Automation (ICCA). IEEE, 2017.
DOI: 10.1109/ICCA.2017.8003201

References

- [1] K. Huh, J. Park, J. Hwang, and D. Hong, “A stereo vision-based obstacle detection system in vehicles,” *Optics and Lasers in Engineering*, vol. 46, no. 2, pp. 168 – 178, 2008.
- [2] R. D. Morton and E. Olson, “Positive and negative obstacle detection using the HLD classifier,” in *IEEE International Conference on Intelligent Robots and Systems*, pp. 1579–1584, 2011.
- [3] S. Sivaraman and M. M. Trivedi, “A review of recent developments in vision-based vehicle detection,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 310–315, 2013.
- [4] Z. Sun, G. Bebis, and R. Miller, “On-road vehicle detection: A review,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 5, pp. 694–711, 2006.
- [5] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, “Survey of pedestrian detection for advanced driver assistance systems,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1239–1258, 2010.
- [6] R. Labayrade, D. Aubert, and J.-P. Tarel, “Real time obstacle detection in stereovision on non flat road geometry through ”v-disparity” representation,” *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, no. January, pp. 646–651, 2002.
- [7] R. Labayrade, D. Aubert, and J. . Tarel, “Real time obstacle detection in stereovision on non flat road geometry through ”v-disparity” representation,” in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, pp. 646–651 vol.2, June 2002.
- [8] A. Iloie, I. Giosan, and S. Nedevschi, “UV disparity based obstacle detection and pedestrian classification in urban traffic scenarios,” *Proceedings - 2014*

IEEE 10th International Conference on Intelligent Computer Communication and Processing, ICCP 2014, pp. 119–125, 2014.

- [9] M. Wu, S. K. Lam, and T. Srikanthan, “Nonparametric Technique Based High-Speed Road Surface Detection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 874–884, 2015.
- [10] M. Wu, S. Member, C. Zhou, T. Srikanthan, and S. Member, “Robust and Low Complexity Obstacle Detection and Tracking Far and High Region,” in *International Conference on Intelligent Transportation Systems*, pp. 1249–1254, 2016.
- [11] F. Oniga and S. Nedevschi, “Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection,” *IEEE Transactions on Vehicular Technology*, vol. 59, no. 3, pp. 1172–1182, 2010.
- [12] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, “MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving,” *arXiv:1612.07695*, 2016.
- [13] V. Badrinarayanan, A. Handa, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling,” *arXiv preprint arXiv:1505.07293*, 2015.
- [14] C. Rabe, U. Franke, and S. Gehrig, “Fast detection of moving objects in complex scenarios,” in *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 398–403, IEEE, 2007.
- [15] P. Lenz, J. Ziegler, A. Geiger, and M. Roser, “Sparse scene flow segmentation for moving object detection in urban environments,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 926–932, IEEE, 2011.
- [16] A. Bak, S. Bouchafa, and D. Aubert, “Dynamic objects detection through visual odometry and stereo-vision: A study of inaccuracy and improvement sources,” *Machine Vision and Applications*, vol. 25, no. 3, pp. 681–697, 2014.
- [17] V. I. Ponomaryov and D. I. Rosas-Miranda, “Automatic detection and classification of obstacles with applications in autonomous mobile robots,” in *SPIE*

- Photonics Europe*, vol. 9897, pp. 98970P–98970P–11, International Society for Optics and Photonics, 2016.
- [18] T. Kim and S.-K. Ryu, “Review and analysis of pothole detection methods,” *Journal of Emerging Trends in Computing and Information Sciences*, vol. 5, no. 8, pp. 603–608, 2014.
 - [19] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, “The pothole patrol: using a mobile sensor network for road surface monitoring,” in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pp. 29–39, ACM, 2008.
 - [20] R. Madli, S. Hebbar, P. Pattar, and V. Golla, “Automatic detection and notification of potholes and humps on roads to aid drivers,” *IEEE Sensors Journal*, vol. 15, pp. 4313–4318, Aug 2015.
 - [21] A. Allouch, A. Kouba, T. Abbes, and A. Ammar, “Roadsense: Smartphone application to estimate road conditions using accelerometer and gyroscope,” *IEEE Sensors Journal*, vol. 17, pp. 4231–4238, July 2017.
 - [22] L. Matthies and a. Rankin, “Negative obstacle detection by thermal signature,” *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1, no. 818, 2003.
 - [23] A. Rankin, A. Huertas, L. Matthies, M. Bajracharya, C. Assad, S. Brennan, P. Bellutta, O. G. Drive, and G. W. Sherwin, “Unmanned ground vehicle perception using thermal infrared cameras,” *Spie*, vol. 8045, no. 818, pp. 1–26, 2011.
 - [24] Y. Bhatia, R. Rai, V. Gupta, N. Aggarwal, A. Akula, *et al.*, “Convolutional neural networks based potholes detection using thermal imaging,” *Journal of King Saud University-Computer and Information Sciences*, 2019.
 - [25] C. Fernández, M. Gavilán, D. F. Llorca, I. Parra, R. Quintero, A. G. Lorente, L. Vlacic, and M. A. Sotelo, “Free space and speed humps detection using

- lidar and vision for urban autonomous navigation,” *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 698–703, 2012.
- [26] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and R. Warner, “Toward reliable off road autonomous vehicles operating in challenging environments,” *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 449–483, 2006.
 - [27] N. Vandapel, R. R. Donamukkala, and M. Hebert, “Unmanned Ground Vehicle Navigation Using Aerial Ladar Data,” *The International Journal of Robotics Research*, vol. 25, no. 1, pp. 31–51, 2006.
 - [28] N. Heckman, J. F. Lalonde, N. Vandapel, and M. Hebert, “Potential negative obstacle detection by occlusion labeling,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 2168–2173, 2007.
 - [29] J. Larson and M. Trivedi, “Lidar based off-road negative obstacle detection and analysis,” in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 192–197, IEEE, 2011.
 - [30] X. Yu and E. Salari, “Pavement pothole detection and severity measurement using laser imaging,” in *Electro/Information Technology (EIT), 2011 IEEE International Conference on*, pp. 1–5, IEEE, 2011.
 - [31] M. Bajracharya, J. Ma, M. Malchano, A. Perkins, A. A. Rizzi, and L. Matthies, “High fidelity day/night stereo mapping with vegetation and negative obstacle detection for vision-in-the-loop walking,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3663–3670, Nov 2013.
 - [32] Z. Zhang, X. Ai, C. Chan, and N. Dahnoun, “An efficient algorithm for pothole detection using stereo vision,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 564–568, IEEE, 2014.

- [33] A. Mikhailiuk and N. Dahnoun, “Real-time pothole detection on tms320c6678 dsp,” in *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*, pp. 123–128, Oct 2016.
- [34] A. Dhiman, H. Chien, and R. Klette, “Road surface distress detection in disparity space,” in *2017 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pp. 1–6, Dec 2017.
- [35] A. Dhiman, H. Chien, and R. Klette, “A multi-frame stereo vision-based road profiling technique for distress analysis,” in *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, pp. 7–14, Oct 2018.
- [36] C. Koch and I. Brilakis, “Pothole detection in asphalt pavement images,” *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 507–515, 2011.
- [37] S.-K. Ryu, T. Kim, and Y.-R. Kim, “Image-based pothole detection system for its service and road management system,” *Mathematical Problems in Engineering*, vol. 2015, 2015.
- [38] I. Schiopu, J. P. Saarinen, L. Kettunen, and I. Tabus, “Pothole detection and tracking in car video sequence,” in *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 701–706, June 2016.
- [39] A. Akagic, E. Buza, and S. Omanovic, “Pothole detection: An efficient vision based method using rgb color space image segmentation,” in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1104–1109, May 2017.
- [40] A. Tedeschi and F. Benedetto, “A real-time automatic pavement crack and pothole recognition system for mobile android-based devices,” *Advanced Engineering Informatics*, vol. 32, pp. 11–25, 2017.
- [41] N.-D. Hoang, “An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction,” *Advances in Civil Engineering*, vol. 2018, 2018.

- [42] N.-D. Hoang, “Image processing based automatic recognition of asphalt pavement patch using a metaheuristic optimized machine learning approach,” *Advanced Engineering Informatics*, vol. 40, pp. 110–120, 2019.
- [43] M. H. Yousaf, K. Azhar, F. Murtaza, and F. Hussain, “Visual analysis of asphalt pavement for detection and localization of potholes,” *Advanced Engineering Informatics*, vol. 38, pp. 527–537, 2018.
- [44] C. Dima, N. Vandapel, and M. Hebert, “Classifier fusion for outdoor obstacle detection,” *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, pp. 665–671, 2004.
- [45] S. Li, C. Yuan, D. Liu, and H. Cai, “Integrated processing of image and gpr data for automated pothole detection,” *Journal of computing in civil engineering*, vol. 30, no. 6, p. 04016015, 2016.
- [46] H. Wu, L. Yao, Z. Xu, Y. Li, X. Ao, Q. Chen, Z. Li, and B. Meng, “Road pothole extraction and safety evaluation by integration of point cloud and images derived from mobile mapping sensors,” *Advanced Engineering Informatics*, vol. 42, p. 100936, 2019.
- [47] S. Sivaraman and M. M. Trivedi, “A review of recent developments in vision-based vehicle detection,” in *Intelligent Vehicles Symposium*, pp. 310–315, Citeseer, 2013.
- [48] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.
- [49] J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. Burkitt, “Performance of optical flow techniques,” in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pp. 236–242, IEEE, 1992.
- [50] B. Kitt, B. Ranft, and H. Lategahn, “Detection and tracking of independently moving objects in urban environments,” in *Intelligent Transportation*

Systems (ITSC), 2010 13th International IEEE Conference on, pp. 1396–1401, IEEE, 2010.

- [51] Q. Chen and V. Koltun, “Full flow: Optical flow estimation by global optimization over regular grids,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4706–4714, 2016.
- [52] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [53] B. Alexe, T. Deselaers, and V. Ferrari, “Measuring the objectness of image windows,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.
- [54] C. L. Zitnick and P. Dollár, “Edge boxes: Locating object proposals from edges,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 391–405, 2014.
- [55] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, “Bing: Binarized normed gradients for objectness estimation at 300fps,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [56] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for object recognition,” *International Journal of Computer Vision*, vol. 104, pp. 154–171, Sep 2013.
- [57] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [58] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.
- [59] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, June 2017.

- [60] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask r-cnn,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [61] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, 2014.
- [62] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [63] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 21–37, Springer International Publishing, 2016.
- [64] H. Law and J. Deng, “Cornersnet: Detecting objects as paired keypoints,” in *The European Conference on Computer Vision (ECCV)*, September 2018.
- [65] H. Law, Y. Teng, O. Russakovsky, and J. Deng, “Cornersnet-lite: Efficient keypoint based object detection,” *arXiv preprint arXiv:1904.08900*, 2019.
- [66] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “Centernet: Keypoint triplets for object detection,” in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.
- [67] X. Zhou, D. Wang, and P. Krähenbühl, “Objects as points,” *CoRR*, vol. abs/1904.07850, 2019.
- [68] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.
- [69] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *arXiv preprint arXiv:1511.00561*, 2015.

- [70] A. Adam, E. Rivlin, and I. Shimshoni, “Robust fragments-based tracking using the integral histogram,” in *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 798–805, IEEE, 2006.
- [71] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2411–2418, 2013.
- [72] Z. Kalal, K. Mikolajczyk, J. Matas, *et al.*, “Tracking-learning-detection,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, p. 1409, 2012.
- [73] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, “Visual object tracking using adaptive correlation filters,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2544–2550, IEEE, 2010.
- [74] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.
- [75] D. Held, S. Thrun, and S. Savarese, “Learning to track at 100 fps with deep regression networks,” in *European Conference Computer Vision (ECCV)*, 2016.
- [76] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature verification using a”siamese” time delay neural network,” in *Advances in neural information processing systems*, pp. 737–744, 1994.
- [77] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Convolutional features for correlation filter based visual tracking,” in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.
- [78] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, “Fully-convolutional siamese networks for object tracking,” in *Computer Vision – ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 850–865, Springer International Publishing, 2016.

- [79] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, “High performance visual tracking with siamese region proposal network,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [80] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, “Multiple object tracking: A literature review,” *arXiv preprint arXiv:1409.7618*, 2014.
- [81] W. Hu, X. Li, W. Luo, X. Zhang, S. Maybank, and Z. Zhang, “Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2420–2440, 2012.
- [82] L. Zhang and L. Van Der Maaten, “Preserving structure in model-free tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 756–769, 2013.
- [83] L. Zhang and L. van der Maaten, “Structure preserving object tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1838–1845, 2013.
- [84] B. Wu and R. Nevatia, “Cluster boosted tree classifier for multi-view, multi-pose object detection,” in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, IEEE, 2007.
- [85] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, “Online multi-object tracking with dual matching attention networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 366–382, 2018.
- [86] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn, “Fusion of head and full-body detectors for multi-object tracking,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1509–150909, IEEE, 2018.
- [87] S. Tang, M. Andriluka, B. Andres, and B. Schiele, “Multiple people tracking by lifted multicut and person reidentification,” in *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pp. 3539–3548, 2017.

- [88] A. Sadeghian, A. Alahi, and S. Savarese, “Tracking the untrackable: Learning to track multiple cues with long-term dependencies,” in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 300–311, 2017.
- [89] W. Choi, “Near-online multi-target tracking with aggregated local flow descriptor,” in *Proceedings of the IEEE international conference on computer vision*, pp. 3029–3037, 2015.
- [90] B. Lee, E. Erdenee, S. Jin, M. Y. Nam, Y. G. Jung, and P. K. Rhee, “Multi-class multi-object tracking using changing point detection,” in *European Conference on Computer Vision*, pp. 68–83, Springer, 2016.
- [91] S. Sharma, J. A. Ansari, J. Krishna Murthy, and K. Madhava Krishna, “Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [92] J. Hong Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon, “Online multi-object tracking via structural constraint event aggregation,” in *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pp. 1392–1400, 2016.
- [93] A. A. Butt and R. T. Collins, “Multi-target tracking by lagrangian relaxation to min-cost network flow,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1846–1853, 2013.
- [94] X. Wang, E. Türetken, F. Fleuret, and P. Fua, “Tracking interacting objects using intertwined flows,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2312–2326, 2016.
- [95] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele, “Motion segmentation & multiple object tracking by correlation co-clustering,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.

- [96] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, “Multiple hypothesis tracking revisited,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4696–4704, 2015.
- [97] X. Wan, J. Wang, Z. Kong, Q. Zhao, and S. Deng, “Multi-object tracking using online metric learning with long short-term memory,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 788–792, Oct 2018.
- [98] G. Gündüz and T. Acarman, “A lightweight online multiple object vehicle tracking method,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 427–432, IEEE, 2018.
- [99] Y. Xiang, A. Alahi, and S. Savarese, “Learning to track: Online multi-object tracking by decision making,” in *Proceedings of the IEEE international conference on computer vision*, pp. 4705–4713, 2015.
- [100] K. Fang, Y. Xiang, X. Li, and S. Savarese, “Recurrent autoregressive networks for online multi-object tracking,” in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 466–475, March 2018.
- [101] Y.-c. Yoon, A. Boragule, Y.-m. Song, K. Yoon, and M. Jeon, “Online multi-object tracking with historical appearance matching and scene adaptive detection filtering,” in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, IEEE, 2018.
- [102] S.-H. Lee, M.-Y. Kim, and S.-H. Bae, “Learning discriminative appearance models for online multi-object tracking with appearance discriminability measures,” *IEEE Access*, vol. 6, pp. 67316–67328, 2018.
- [103] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan, “Poi: Multiple object tracking with high performance detection and appearance feature,” in *Computer Vision – ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 36–42, Springer International Publishing, 2016.

- [104] L. Zhao, X. Li, Y. Zhuang, and J. Wang, “Deeply-learned part-aligned representations for person re-identification,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [105] A. Hermans, L. Beyer, and B. Leibe, “In defense of the triplet loss for person re-identification,” *arXiv preprint arXiv:1703.07737*, 2017.
- [106] Z. Zhou, J. Xing, M. Zhang, and W. Hu, “Online multi-target tracking with tensor-based high-order graph matching,” in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 1809–1814, Aug 2018.
- [107] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, Sep. 2017.
- [108] N. Mahmoudi, S. M. Ahadi, and M. Rahmati, “Multi-target tracking using cnn-based features: Cnnmtt,” *Multimedia Tools and Applications*, vol. 78, pp. 7077–7096, Mar 2019.
- [109] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, “Accurate single stage detector using recurrent rolling convolution,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [110] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, “Subcategory-aware convolutional neural networks for object proposals and detection,” in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pp. 924–933, IEEE, 2017.
- [111] L. Chen, H. Ai, Z. Zhuang, and C. Shang, “Real-time multiple people tracking with deeply learned candidate selection and person re-identification,” in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, July 2018.
- [112] J. H. Yoon, M.-H. Yang, J. Lim, and K.-J. Yoon, “Bayesian multi-object tracking using motion context from multiple objects,” in *Applications of*

- Computer Vision (WACV), 2015 IEEE Winter Conference on*, pp. 33–40, IEEE, 2015.
- [113] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, “Eco: efficient convolution operators for tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6638–6646, 2017.
 - [114] Z. Fu, F. Angelini, J. Chambers, and S. M. Naqvi, “Multi-level cooperative fusion of gm-phd filters for online multiple human tracking,” *IEEE Transactions on Multimedia*, vol. 21, pp. 2277–2291, Sep. 2019.
 - [115] Z. Fu, P. Feng, F. Angelini, J. Chambers, and S. M. Naqvi, “Particle phd filter based multiple human tracking using online group-structured dictionary learning,” *IEEE Access*, vol. 6, pp. 14764–14778, 2018.
 - [116] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granström, “Mono-camera 3d multi-object tracking using deep learning detections and PMBM filtering,” in *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, pp. 433–440, 2018.
 - [117] A. Osep, W. Mehner, M. Mathias, and B. Leibe, “Combined image-and world-space tracking in traffic scenes,” in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 1988–1995, IEEE, 2017.
 - [118] H. W. Kuhn, “The hungarian method for the assignment problem,” *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.
 - [119] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran, “Detect-and-track: Efficient pose estimation in videos,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
 - [120] E. Bochinski, V. Eiselein, and T. Sikora, “High-speed tracking-by-detection without using image information,” in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Aug 2017.

- [121] E. Bochinski, T. Senst, and T. Sikora, “Extending iou based multi-object tracking by visual information,” in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Nov 2018.
- [122] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 91–99, Curran Associates, Inc., 2015.
- [123] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, “Mot16: A benchmark for multi-object tracking,” *arXiv preprint arXiv:1603.00831*, 2016.
- [124] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [125] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [126] B. Li, Q. Yao, and K. Wang, “A review on vision-based pedestrian detection in intelligent transportation systems,” in *Proceedings of 2012 9th IEEE international conference on networking, sensing and control*, pp. 393–398, IEEE, 2012.
- [127] A. Mukhtar, L. Xia, and T. B. Tang, “Vehicle detection techniques for collision avoidance systems: A review,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2318–2338, 2015.
- [128] Y. Y. Boykov and M. . Jolly, “Interactive graph cuts for optimal boundary region segmentation of objects in n-d images,” in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, pp. 105–112 vol.1, 2001.
- [129] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, “SLIC superpixels compared to state-of-the-art superpixel methods,” *IEEE*

- Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2281, 2012.
- [130] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
 - [131] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.
 - [132] J. Yao, M. Boben, S. Fidler, and R. Urtasun, “Real-time coarse-to-fine topologically preserving segmentation,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 2947–2955, 2015.
 - [133] H. Hirschmüller, “Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information,” *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Volume: 2)*, pp. 807 – 814, 2005.
 - [134] S. Montabone and A. Soto, “Human detection using a mobile platform and novel features derived from a visual saliency mechanism,” *Image and Vision Computing*, vol. 28, no. 3, pp. 391–402, 2010.
 - [135] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
 - [136] S. Agarwal, K. Mierle, and Others, “Ceres solver.”
 - [137] R. Guzmán, J.-B. Hayet, and R. Klette, “Towards ubiquitous autonomous driving: The ccsad dataset,” in *Computer Analysis of Images and Patterns* (G. Azzopardi and N. Petkov, eds.), (Cham), pp. 582–593, Springer International Publishing, 2015.
 - [138] Y. Xu, X. Zhou, S. Chen, and F. Li, “Deep learning for multiple object tracking: a survey,” *IET Computer Vision*, vol. 13, pp. 355–368(13), June 2019.

- [139] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.
- [140] T. Ahonen, A. Hadid, and M. Pietikainen, “Face description with local binary patterns: Application to face recognition,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 28, pp. 2037–2041, dec 2006.
- [141] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, “MOT16: A benchmark for multi-object tracking,” *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831.
- [142] K. Bernardin and R. Stiefelhagen, “Evaluating multiple object tracking performance: the clear mot metrics,” *Journal on Image and Video Processing*, vol. 2008, p. 1, 2008.
- [143] Y. Li, C. Huang, and R. Nevatia, “Learning to associate: Hybridboosted multi-target tracker for crowded scene,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2953–2960, IEEE, 2009.
- [144] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.
- [145] F. Yang, W. Choi, and Y. Lin, “Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2129–2137, 2016.
- [146] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, “Deep layer aggregation,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [147] H. Karunasekera, H. Zhang, and H. Wang, “Real time multiple object tracking using deep features and localization information,” in *2019 IEEE 15th*

- International Conference on Control and Automation (ICCA)*, pp. 332–337, July 2019.
- [148] D. Marr and T. Poggio, “A computational theory of human stereo vision,” *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 204, no. 1156, pp. 301–328, 1979.
 - [149] C. Loop and Z. Zhang, “Computing rectifying homographies for stereo vision,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, pp. 125–131, IEEE, 1999.
 - [150] A. Fusiello, E. Trucco, and A. Verri, “A compact algorithm for rectification of stereo pairs,” *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.
 - [151] D. Scharstein, R. Szeliski, and R. Zabih, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” in *Stereo and Multi-Baseline Vision, 2001.(SMBV 2001). Proceedings. IEEE Workshop on*, pp. 131–140, IEEE, 2001.
 - [152] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, “Review of stereo vision algorithms and their suitability for resource-limited systems,” *Journal of Real-Time Image Processing*, vol. 11, no. 1, pp. 5–25, 2016.
 - [153] A. Smith and A. Abdel-Dayem, “A developmental survey of local stereo matching algorithms,” *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 10, no. 8, pp. 1398–1404, 2016.
 - [154] R. Zabih and J. Woodfill, “Non-parametric local transforms for computing visual correspondence,” in *European conference on computer vision*, pp. 151–158, Springer, 1994.
 - [155] Y. S. Heo, K. M. Lee, and S. U. Lee, “Robust stereo matching using adaptive normalized cross-correlation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 807–822, 2011.

- [156] H. Hirschmuller and D. Scharstein, “Evaluation of cost functions for stereo matching,” in *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pp. 1–8, IEEE, 2007.
- [157] L. Hong and G. Chen, “Segment-based stereo matching using graph cuts,” in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2004.
- [158] J. Sun, N.-N. Zheng, and H.-Y. Shum, “Stereo matching using belief propagation,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 7, pp. 787–800, 2003.
- [159] S. Forstmann, Y. Kanou, J. Ohya, S. Thuerling, and A. Schmitt, “Real-time stereo by using dynamic programming,” in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW’04. Conference on*, pp. 29–29, IEEE, 2004.
- [160] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [161] J. Zbontar and Y. LeCun, “Computing the stereo matching cost with a convolutional neural network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1592–1599, 2015.
- [162] W. Luo, A. G. Schwing, and R. Urtasun, “Efficient Deep Learning for Stereo Matching,” *Conference on Computer Vision and Pattern Recognition*, pp. 5695–5703, 2016.
- [163] A. Seki and M. Pollefeys, “Sgm-nets: Semi-global matching with neural networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [164] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, “High-resolution stereo datasets with subpixel-

- accurate ground truth,” in *German Conference on Pattern Recognition*, pp. 31–42, Springer, 2014.
- [165] D. Scharstein and C. Pal, “Learning conditional random fields for stereo,” in *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pp. 1–8, IEEE, 2007.
 - [166] D. Scharstein and R. Szeliski, “High-accuracy stereo depth maps using structured light,” in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2003.
 - [167] T. A. Clarke and J. G. Fryer, “The development of camera calibration methods and models,” *The Photogrammetric Record*, vol. 16, no. 91, pp. 51–66, 1998.
 - [168] G. Bradski, “The OpenCV Library,” *Dr. Dobbs’s Journal of Software Tools*, 2000.
 - [169] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.
 - [170] J. Heikkila and O. Silven, “A four-step camera calibration procedure with implicit image correction,” in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 1106–1112, IEEE, 1997.
 - [171] Z. Zhang, “Flexible camera calibration by viewing a plane from unknown orientations,” in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 666–673, Ieee, 1999.
 - [172] P. F. Sturm and S. J. Maybank, “On plane-based camera calibration: A general algorithm, singularities, applications,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, pp. 432–437, IEEE, 1999.

- [173] H. Hirschmuller and D. Scharstein, “Evaluation of stereo matching costs on images with radiometric differences,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1582–1599, Sep. 2009.

Appendix A: Stereo vision methods and derivations

Introduction

Stereo vision is the study on calculating disparity (depth) information from two images on the same scene. Humans can sense depth because they have the view of the same scene from right eye and the left eye. In a similar manner two cameras looking at the same environment can be used to estimate the 3D information of the scene.

Stereo matching has been studied over three decades, which was first introduced by [148] in 1979. Even though this has been studied for a long time, still research is active in this area. Stereo vision has two main areas, which are (a) stereo matching and (b) 3D reconstruction. Furthermore, stereo vision has been used in Chapter 4 and Chapter 5 and hence (c) camera setup and implementation is discussed in this chapter.

Stereo Matching

Stereo matching is the process of finding the corresponding points between the left and right images and calculating the disparity for each pixel on the image based on that. Generating matching points between the two frames is known as the correspondence problem. Once matching correspondences are generated disparity value is the co-ordinates (pixel positions in the two image views) difference between the two pixels corresponding to the same point in the world. Fig. A.1 visualize this situation and mathematical representation for disparity d calculation is given in (1) where subscript l and r are to indicate left and right image views.

$$d = u_l - u_r \quad (1)$$

Two main challengers in correspondence problem are (a) how to match and (b) what to match. The first can be simply done by conducting a 2D search in one image to a reference point in the other. However, this is a very time consuming process which can be efficiently solve using rectification. With image

Figure A.1: Disparity from stereo vision

rectification 2D search can be minimized to a 1D search using the principles of Epipolar Geometry.

Figure A.2: Epipolar Geometry

In the Fig. A.2, O_l and O_r are the two camera centers and $O_l O_r$ is the 3D line connecting the two camera centers which intersects the left and right image planes at E_l and E_r , which are called epipolar points. $O_l P O_r$ plane is termed as the epipolar plane. All the points in the world line $O_l P$ is seen by the right camera image plane $p_r E_r$ line. Therefore when you need to search for a match for point P in the world which is also a point in $O_l P$ line, you only need to search in the corresponding epipolar line $p_r E_r$ in the right image. This is effectively reducing your 2D search into a 1D search. Given a point $p(u, v)$ in an image corresponding

epipolar line l' in the other image can easily be found using (2) mentioned below, where F is the Fundamental Matrix between two images. Fundamental matrix can be easily found using intrinsic and extrinsic parameters, which is common knowledge in stereo vision. Interested readers are referred to [149, 150].

$$l' = F.p(u, v) \quad (2)$$

With help of epipolar geometry, 2D search has been reduced to a 1D search. However scanning an arbitrary line is also inefficient in image processing. In order to address this, as proposed in [149], epipolar lines are made horizontal and parallel to each other. This is when epipolar points are placed at infinity. This process is known as stereo image rectification. Once you have rectified image pair, match to a point in one image can be found just scanning a horizontal line in the other. Refer to the Fig. A.3 below to visualize the results.

Figure A.3: Image rectification

Once the images have been rectified the next challenge is to define a good metric to match the points between two images, which is second problem known as the matching problem. Many research has been conducted in addressing this stereo matching problem and still an area of active research. Literature in this area can be mainly categorized as (a) traditional and (b) deep learning based approaches.

Traditional stereo matching

[151] provides an excellent analysis and categorization of the significant achievements and work done in early Stereo vision while [152, 153] provide a review about more recent work in the area. Following the same categorization presented in [151],

there are two main types of stereo matching algorithms in traditional stereo matching approaches, which are (a) local stereo methods and (b) global stereo methods. Main difference in these two methods are based on how each algorithm performs optimization, whether the optimization is done locally or globally. Moreover, stereo matching process is divided into four parts, namely (1) matching cost computation, (2) cost aggregation, (3) disparity calculation and optimization and (4) disparity refinement. Fig. A.4 shows the main steps in two stereo matching methods.

Figure A.4: Stereo matching algorithms categorization

Normally in stereo matching disparity values are calculated for pre determined range. Let's assume this range is from $d = 0$ to $d = d_{max}$. The most appropriate disparity value is then selected for each pixel using the algorithm which minimizes

the dis-similarity calculated by the matching cost function followed by cost aggregation. Typical matching cost per pixel can be defined as per (3). The subscripts l and r are to indicate left and right respectively, for each pixel p while e_l is a vector of matching cost for pixels in the left image and F is matching cost function defined to calculate the dis-similarity between pixels in left and right images. Number of matching cost functions F have been introduced, among which Sum of Absolute Difference (SAD), Normalized Cross Correlation (NCC), Rank Transform (RT) [154], Census Transform (CT) [154], Mutual Information (MI) [133] and Adaptive NCC (A-NCC) [155] are famous methods. Authors in [156] evaluates several cost functions in their paper.

$$e_l(p, d) = F(p_l(u, v), p_r(u - d, v)) \quad (3)$$

Cost aggregation is mostly used in local windows based optimization methods while in most global methods it is indirectly addressed in the optimization step itself. Cost aggregation will smooth your per pixel cost based on cost values in the neighbourhood, as a weighted average as expressed in (4) below. Typically this weight is calculated using the colour/intensity information from the original image.

$$E_l(p, d) = \frac{\sum_{q \in W(p)} w(p, q) e_l(q, d)}{\sum_{q \in W(p)} w(p, q)} \quad (4)$$

Here $W(p)$ is the window considered around the pixel p and q denotes the other pixels in the window except for p . $w(p, q)$ is the weight for each q pixel in the window $W(p)$ and $e_l(q, d)$ is the value calculated in (3). $E_l(p, d)$ is a 3D cost volume where, height and width corresponds to the image height and width and length will be the disparity range considered (from $d = 0$ to $d = d_{max}$). This is also known as the Disparity Space Image (DSI).

Next step is the disparity value calculation using optimization. In local methods, Winner Takes All (WTA) approach is used to obtain the winning disparity value as in (5).

$$d_l(p) = \operatorname{argmin}_{d(p)} E_l(p, d) \quad (5)$$

Among many global optimization based methods Graph-cut (GC) based optimization [157], Belief Propagation (BP) based optimization [158] and Dynamic Programming (DP) based optimization [159, 133] are popular. All these global optimization methods follow an equation similar to below (6).

$$d_l(p) = \underset{d(p)}{\operatorname{argmin}} \left(E_l(p, d) + \lambda \sum_{q \in W(p)} E_s(d_p, d_q) \right) \quad (6)$$

where $E_s(d_p, d_q)$ is the smoothness term considering the smooth transition in the neighbourhood $W(p)$.

The final step in stereo matching algorithm is to refine the disparity map using different post processing steps. This typically involves occlusion handling, obtaining sub-pixel accuracy and outlier rejection.

Deep learning based stereo matching

In the recent past, deep learning based stereo matching methods have emerged with increased accuracy because of the specialized hardware (Graphical Processing Units - GPUs) and large datasets with ground truth data. Convolution Neural Netwrk (CNN) is an extension on the neural network architecture where the first few layers of a CNN are convolution layers which are a network where each neuron is connected to spatially connected subset of neurons in the previous layer which effectively reduces the amount of data present to the fully connected layers. CNN has been widely applied in image processing related tasks because its architecture suits image analysis very well. A detailed explanation is provided by authors in [160] about CNN architecture. With the recent popularity of CNNs, stereo matching process has been all been studied in relation to it. In [161], authors use a CNN architecture to extract features and calculate their matching cost. Feature extraction is done for both left and right images separately but with identical CNNs. Then these extracted features are concatenated and matched using a fully connected network. This will output the 3D DSI which will then be processed outside the network to obtain the final disparity after implementing a global optimization and post-processing steps. Authors in [162] compute the matching cost based on CNN network, where with introduction of inner layer which computes the prod-

uct of feature vectors to get results in seconds compared to minutes in previous architectures. In another work [163] authors use a CNN architecture to predict the penalties for the SGM algorithm to generate better disparity results. At each image, small patch with patch position is inputted to the network to predict the best values for the two penalties of SGM for different directions. Deep learning based stereo matching can generate better disparity maps compared to traditional methods. However, these method still cannot match the speeds of the traditional methods. Thus using deep network based stereo matching in real time systems is still not viable.

Since stereo matching has been studied for a long time, benchmark datasets are available with ground truth data. Middlebury Stereo ¹ has five datasets [164, 156, 165, 166, 151] and KITTI has two datasets [125, 52]. These benchmark datasets provide evaluations of the current state-of-the-art stereo matching algorithm performance on their datasets.

3D Reconstruction

3D reconstruction is the process of finding the real world (X, Y, Z) point information for each pixel in the image plane. Detailed explanation on obtaining 3D information for the general stereo case is presented in [7]. The Fig. A.5 below is the representation in [7].

Camera model

It is important to understand the camera model in calculating the 3D points from the disparity image. Camera parameters are of two sorts, (a) intrinsic camera parameters and (b) extrinsic camera parameters. Intrinsic corresponds to the focal lengths and principal axis points, while extrinsic parameters are the camera rotation and translation with respect to a known world point.

The functionality of a camera is that it projects a 3D world point into a 2D image plane. This transformation is approximated using the projection matrix, which is also known as the camera matrix. Camera matrix is a combination from intrinsic matrix and extrinsic matrix. Extrinsic matrix is a combination of rotation

¹Middlebury: <http://vision.middlebury.edu/stereo/>

Figure A.5: Stereo system and co-ordinate system

and translation matrices, used for conversion from the world model to camera model. Camera matrix is derived based on the pin-hole camera model, which is presented below.

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \mathbf{K}_{3 \times 3} \begin{bmatrix} R_{3 \times 3} & T_{3 \times 1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (7)$$

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (8)$$

$$u = \frac{\tilde{x}}{\tilde{w}} \quad v = \frac{\tilde{y}}{\tilde{w}} \quad (9)$$

$$f_x = \frac{F}{p_x} \quad f_y = \frac{F}{p_y} \quad (10)$$

In above (7), (8), (9) and (10) , (u, v) are the image co-ordinates in pixel units while (X, Y, Z) is corresponding world point in world measurement units. F and (p_x, p_y) are focal length and pixel dimensions (height and width) expressed in world

units and (f_x, f_y) are focal length in pixel units. These equations show how a 3D world co-ordinate is transformed to a 2D image plane. In (7) above, $\mathbf{K}$ denotes the intrinsic camera matrix while R and T are rotation and translation matrices which together forms the extrinsic camera matrix. Intrinsic camera matrix is independent of the scene and hence can be reused as long as the camera focal length remains unchanged. On the other hand extrinsic matrix translate 3D world co-ordinate to camera co-ordinate system. Point in the world co-ordinate system (X, Y, Z) and same point in camera co-ordinate system as (x, y, z) , can be expressed as in (11).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} + T \quad (11)$$

Hence image points (u, v) , can be derived simply in the below manner.

$$u = f_x \times \acute{x} + c_x \quad v = f_y \times \acute{y} + c_y \quad (12)$$

where;

$$\acute{x} = x/z \quad \acute{y} = y/z \quad (13)$$

Camera distortion is common phenomenon which occurs to imperfect manufacturing and properties associated with lenses. Distortion parameters are also of two sorts, which are (a) radial distortion and (b) tangential distortion. Radial distortion occurs because the light rays bend more near the edge of the lens compared to the lens optical center. Tangential distortion occurs when the lens and the image plane are not parallel to each other. The above mentioned camera model is for a pin-hole camera and needs to be adjusted for the distortion parameters. So the model is extended as below. T_D represent tangential distortion while R_D represent radial distortion.

$$u = f_x \left(x' \frac{1 + D_{r1}r^2 + D_{r2}r^4 + D_{r3}r^6}{1 + D_{r4}r^2 + D_{r5}r^4 + D_{r6}r^6} + 2D_{t1}x'y' + D_{t2}(r^2 + 2x'^2) \right) + c_x \quad (14)$$

$$v = f_y \left(y' \frac{1 + D_{r1}r^2 + D_{r2}r^4 + D_{r3}r^6}{1 + D_{r4}r^2 + D_{r5}r^4 + D_{r6}r^6} + D_{t1}(r^2 + 2y'^2) + 2D_{t2}x'y' \right) + c_y \quad (15)$$

where;

$$r^2 = x'^2 + y'^2 \quad (16)$$

$$R_D = (D_{r1}, D_{r2}, D_{r3}, D_{r4}, D_{r5}, D_{r6}) \quad (17)$$

$$T_D = (D_{t1}, D_{t2}) \quad (18)$$

Aim of camera calibration is to find the parameters to model the above mentioned camera model. [167] presents history on camera calibration and methods used. Calibration is a well studied and comes as off-the shelf implementations in both Matlab and OpenCV. Even though there are six parameters used to model radial distortion, both OpenCV[168] and Matlab² uses up to a maximum of three parameters. These implementations are inspired from the work done in [169, 170, 171, 172]. As mentioned in [169, 171] distortion is dominated by radial distortion components. Calibrating a stereo camera pair requires an additional step, which is to calculate the rotation and translation of the second camera (usually the right camera) with respect to the first camera (usually the left camera). These calibration parameters (intrinsic and extrinsic parameters, radial and tangential distortion parameters) are required to perform rectified images when solving the correspondence problem efficiently and estimating the 3D world coordinates for disparity values.

Estimating 3D Coordinates

Using the equations described in (7), (8), (9), the mapping between the world coordinates (X, Y, Z) and the image co-ordinates (u, v) can be found. According to the above Fig. A.5, (7) can be defined as below (19).

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\varepsilon_i \frac{b}{2} \\ 0 & \cos\theta & -\sin\theta & h \\ 0 & \sin\theta & \cos\theta & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (19)$$

²Matlab: <https://www.mathworks.com/>

where θ is the angle between the optical axis of the cameras $O_r R_{cr}$, $O_l R_{cl}$ and the horizontal axis. b is the baseline, which is the distance between the camera optical centres and h is the height of the cameras mounted from the origin. ε_i is a constant where $\varepsilon_i = -1$ for left image and $\varepsilon_i = 1$ for the right image.

The roll angle θ can be made negligible when mounting the cameras in the vehicle by making the camera rig parallel to the road surface. This simple trick will reduce the complexity of (19) as $\theta \approx 0$ will make $\sin\theta \approx 0$ and $\cos\theta \approx 1$. Furthermore, if the origin is assumed to be at the optical center of the left image, (19) can be re-written as below (20) for the left image and (21) for the right image.

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (20)$$

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (21)$$

If the left image is considered, i.e.(20) the relationships between the physical world and image world can be derived using (22) and (23).

$$u = \frac{\tilde{x}}{\tilde{w}} = c_x + \frac{f_x X}{Z} \quad (22)$$

$$v = \frac{\tilde{y}}{\tilde{w}} = c_y + \frac{f_y Y}{Z} \quad (23)$$

Now there are two equations to find the three world co-ordinates (X, Y, Z) . Given an image point, the image co-ordinate (u, v) and with calibration c_x, c_y, f_x and f_y are known. Consider $PO_r O'_l$ triangle in the Fig. A.6 which represent a rectified stereo camera system with parallel epipolar lines, to arrive at (24).

$$Z = f \frac{b}{u_l - u_r} = f \frac{b}{d} \quad (24)$$

where f is the focal length of the camera system. General assumption in such stereo systems is that $f \approx f_x \approx f_y$, where f can be found by taking the average of the f_x and f_y of the two cameras. The baseline b can be found from the calibration process. This process of determining depth from disparity d is known as **triangulation**. Note that depth Z is inversely proportional to disparity d .

Figure A.6: Stereo system with parallel epipolar lines

Therefore, with this configuration the depth Z for all the pixels (u, v) in the left image for which disparity value d is known can be calculated using (24). Then by using (22) and (23), X and Y co-ordinates can be found, which will give the 3D world co-ordinate. However, a dense disparity map will only provide a sparse 3D map as a pixel-level disparity value will not provide a continuous depth field but only step-wise depth values. By incorporating (24) in (22) and (23), (25) and (26) can be obtained.

$$X = \frac{b(u - c_x)}{d} \quad (25)$$

$$Y = \frac{b(v - c_y)}{d} \quad (26)$$

However, the camera rig can be tilted downwards to focus more on the road. In such cases, the roll angle θ cannot be neglected. In such situations, relationship

between 3D coordinates with image coordinated can be obtained as below;

When the origin is assumed to be at the optical center of the left image, (19) can be re-written as below (27) for the left image and (28) for the right image.

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (27)$$

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -b \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (28)$$

Based on (27), (29) and (30) can be obtained.

$$u = \frac{\tilde{x}}{\tilde{w}} = c_x + \frac{Xf}{Y\sin\theta + Z\cos\theta} \quad (29)$$

$$v = \frac{\tilde{y}}{\tilde{w}} = c_y + \frac{f(Y\cos\theta - Z\sin\theta)}{Y\sin\theta + Z\cos\theta} \quad (30)$$

Based on (28), below (31) can be obtained.

$$u' = \frac{\tilde{x}}{\tilde{w}} = c_x + \frac{Xf - bf}{Y\sin\theta + Z\cos\theta} \quad (31)$$

Therefore, disparity d can be obtained as in (32)

$$d = u - u' = \frac{bf}{Y\sin\theta + Z\cos\theta} \quad (32)$$

Then by combining (29) and (32), below (33) can be obtained.

$$X = \frac{b(u - c_x)}{d} \quad (33)$$

Similarly by combining (30) and (32), below (34) can be obtained.

$$Y = \frac{f b \sin\theta}{d} + \frac{b(v - c_y) \cos\theta}{d} \quad (34)$$

Finally by combining (34) and (32), below (35) can be obtained.

$$Z = \frac{f b \cos\theta}{d} - \frac{b (v - c_y) \sin\theta}{d} \quad (35)$$

Therefore depending on the situation when pitch angle θ is negligible or significant, 3D coordinates of an image point can be easily calculated using (24), (25), (26) or (33), (34), (35). During implementation these can be stored in look up tables and infer in single clock cycle.

Camera Setup and Implementation

Camera Setup

The camera setup used to capture some negative obstacles is a custom made stereo rig. The dual stereo camera rigs, one to look objects that are far away and the other one to look at near-by wider area is as per the Fig. A.7. PointGrey® Flea3 cameras with FUJIFILM® lens were used in these camera rigs. The camera configuration is as according to the Fig. A.7.

Figure A.7: Stereo system setup

PointGrey® Flea3 camera can capture images upto speeds of 150 Hz and output monochrome images. Since the task is time critical, this PointGrey® specification suits the requirement best. High frame capture rate is required for two reasons,

(1) to avoid any blurring due to fast moving camera and vibrations that may occur and (2) to minimize the distance travelled between two adjacent frames. In the experiments a frame capture rate of 50 Hz was used. Camera has a resolution of 1280×1024 pixels which provide satisfactory details of the scene. For the far range camera system two cameras having lenses of 9mm focal length and for wide angle near looking camera pair, 4mm focal length lenses are used. Table A.1 provides a comparison between the specifications of the camera lenses. The images used in Chapter 4 is captured are from far range camera system.

Table A.1: Camera lens specifications

Lens Type	Viewing Range	Focal Length	Horizontal FOV	Verticle FOV
Narrow angle	15 m-50 m	9 mm, 1930 pixels	30°	26°
Wide angle	1 m-20 m	4 mm, 860 pixels	83°	64°

Other than the above two configurations, three off the shelf stereo camera sets have been used with colour images, where one was ZEDTM with 12 cm baseline and others with 50 cm baselines. Camera calibration of this stereo rig was a challenging task, especially due to large baseline for the long range camera system. Checkerboard based calibration approach was used. Calibration was difficult due to capturing the calibration grid (checkerboard) on both the cameras at the same time in all the regions is not possible. For an example, it is not possible to cover the right most area of the left camera as in that instance the checkerboard is not fully covered by the right camera. Another challenge was the decrease of the common field of view because of the large baseline. Common field of view is the common view of the scene shared by the left and right camera. To minimize the effect from these challenges a very large 10x8 calibration board with 125.6×125.6 mm² area for each square was used. This large checkerboard allowed to capture the calibration board even at higher depths. However, doing the calibration over and over was a tedious process due to the size of the board. Refer to Fig. A.8 for visualization of the calibration board.

In order to perform calibration several algorithms were used and stereo camera calibrator in Matlab³ was selected. Fig. A.8 shows results obtained from Matlab

³Matlab: <https://www.mathworks.com/>

Figure A.8: Calibration results

calibration package. To identify calibration result, reprojection error below 0.2 pixels was used as a general rule of thumb. In the following section stereo matching and disparity calculation is discussed.

Implementation of Stereo Matching and Disparity Calculation

With the calibration results the images were rectified which is then used to perform stereo matching to obtain the disparity for the scene. This process is visualized in the Fig. A.9 below.

Stereo matching is based on Semi-Global Matching which was first introduced in [133], where instead of the Mutual Information based per-pixel matching cost calculation Census transformation [154] based per-pixel matching cost calculation is used because of the higher accuracy [173] is used. GPU based Semi-Global Matching implementation called libSGM⁴ is used in experiments. Stereo matching process in [133] is briefly discussed below.

Stereo matching is a four step process, (1) matching cost computation, (2) cost aggregation, (3) disparity calculation and optimization and (4) disparity refinement. In [133] matching cost calculation is done using Mutual Information which is defined based on the entropy (on intensity information) of the two images and joined entropy. Entropy is calculated from the probability distributions of the in-

⁴libSGM: <https://github.com/fixstars/libSGM>

Figure A.9: Disparity map generation process. Top to bottom: Original left and right images, Rectified left and right images, Disparity map.

tensity values on the two images. Instead of using the whole image, they have used information of a neighbourhood window. However [173] suggests that matching cost calculation based on Census Transformation [154] provides a higher accuracy. Census Transformation [154] is generated based on the intensity values of pixels inside a window. If you consider 3×3 window Census Transform is a eight bit codeword, where the intensity value of the center pixel is compared with the eight pixels in the 3×3 neighbourhood in arriving at the eight bit code word. This can be arrived from clockwise direction or anti-clockwise direction, but should adhere to one direction throughout the implementation. If the center pixel intensity is higher than that of the neighbouring pixel in consideration, bit is set to 1 and 0 otherwise. Same convention is followed through out. In [173] different matching cost methods have been analyzed for best performance in both local and global optimization methods. Based on the results authors state that the Census Transformation based matching cost has yielded better results than other methods (this analysis also includes Mutual Information), except when the image is corrupted with strong noise. Therefore using Census Transformation over Mutual Information can be validated. Hamming distance is used for the per pixel cost calculation

for a 16 bit codeword generated considering 16 selected pixels from 9×9 window. Hamming distance is a measurement of the minimum number of bit changes required to match two codewords.

Cost aggregation and optimization is done based on multi-directional Dynamic Programming approach which is the main contribution in [133]. First an energy function is defined, which is as per (36).

$$E(d) = \sum_p C(p, d_p) + \sum_{q \in N_p} P_1 T[|d_p - d_q| = 1] + \sum_{q \in N_p} P_2 T[|d_p - d_q| > 1] \quad (36)$$

where d stands for disparity and q are pixels in the p neighbourhood N_p . P_1 and P_2 are penalty terms and $C(p, d_p)$ is the data term coming from the matching cost. Other two terms are smoothness terms where in the neighbourhood if there is one pixel difference in disparity small penalty P_1 is added and for higher disparity difference larger penalty P_2 is added. This method supports adaptation to slanted surfaces with smaller penalty and preserves discontinuities with a constant penalty for larger differences. Optimizing $E(d)$ is a time consuming process. Hence, authors propose multi-directional 1D Dynamic Programming approach in doing it efficiently. This can be explained based on (37).

$$L_r(p, d) = C(p, d) + \min\{L_r(p-r, d), L_r(p-r, d-1) + P_1, L_r(p-r, d+1) + P_1, \min_i L_r(p-r, i) + P_2\} - \min_k L_r(p-r, k) \quad (37)$$

where L_r is the 1D path travelled in the r direction and $C(p, d)$ pixel matching cost. So the pixel matching cost is added with the lowest cost of the previous pixel $p-r$ of the path with penalties for smoothness enforcement and then minimum path cost (for any disparity) deducted from that. The third term is to refrain L_r continuing to increase along the path. This can be seen as a 1D implementation of (36). To arrive at the aggregated cost L_r from all r are summed as in (38).

$$S(p, d) = \sum_r L_r(p, d) \quad (38)$$

Authors suggest that at least eight directions should be considered for good results. However, good results can be seen even at four directions, which will reduce the running time considerably. Disparity value is then selected using a Winner Takes All approach from $S(p, d)$ that corresponds to the minimum cost ($\min_d S(p, d)$). Hence the name Semi Global Matching (SGM) method. Once the disparity map is generated for the base image, disparity map is generated for the matched image using the same path costs calculated and following a WTA approach. If the disparity values for the same point differ more than one pixel its disparity value is marked as invalid. Hence a dense disparity map is generated with possible missing values due to invalidation.

In libSGM, this method is implemented using CUDA for GPU support to make the algorithm real time performance. A disparity search range up to a maximum of 128 pixels is used in the implementations. In order to further improve the calculation time, half sized images were used.