VIDEO FOREGROUND LOCALIZATION FROM TRADITIONAL METHODS TO DEEP LEARNING

Thangarajah Akilan

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Akilan, Thangarajah, "VIDEO FOREGROUND LOCALIZATION FROM TRADITIONAL METHODS TO DEEP LEARNING" (2018). Electronic Theses and Dissertations. 7462.
https://scholar.uwindsor.ca/etd/7462

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
VIDEO FOREGROUND LOCALIZATION
FROM TRADITIONAL METHODS TO DEEP LEARNING

by
Thangarajah Akilan

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy at the
University of Windsor

Windsor, Ontario, Canada

© 2018 Thangarajah Akilan
VIDEO FOREGROUND LOCALIZATION
FROM TRADITIONAL METHODS TO DEEP LEARNING

by
Thangarajah Akilan

APPROVED BY:

C. Luo, External Examiner
University of Detroit Mercy

B. Boufama
School of Computer Science

H. Wu
Electrical and Computer Engineering

M. Khalid
Electrical and Computer Engineering

J. Wu, Advisor
Electrical and Computer Engineering

April 26, 2018
I Co-Authorship Declaration
I hereby declare that this dissertation incorporates material that is result of joint research, as follows: This dissertation also incorporates the outcome of a research under the supervision of professor Quinmin Jonathan Wu and collaboration with Jie Huo (Chapter 4) and Dr. Yimin Yang (Chapter 6). The research under Prof. QMJ Wu is covered in Chapter 4, 5, 6, 7, 8, and 9 of the dissertation. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by the author, and the contribution of the coauthors was primarily through the provision of proof reading and reviewing the research papers regarding the technical content.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my dissertation, and have obtained written permission from each of the co-authors to include the above materials in my dissertation.

I certify that, with the above qualification, this dissertation, and the research to which it refers, is the product of my own work.

II Previous Publication
This dissertation includes seven original papers that have been previously published/un-der review in peer reviewed journals and conferences, as follows:

<table>
<thead>
<tr>
<th>Thesis Chapter</th>
<th>Publication title/full citation</th>
<th>Publication status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 4</td>
<td>T. Akilan, QMJ. Wu, &quot;Video foreground detection in non-static background using multi-dimensional color space.” Procedia Computer Science 70 (2015): 55-61, ©2015 Elsevier.</td>
<td>Published</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>T. Akilan, QMJ. Wu, and J. Huo. ”A unified threshold updating strategy for multivariate Gaussian mixture based moving object detection.” High Performance Computing &amp; Simulation (HPCS), 2016 International Conference on. IEEE, ©2016 IEEE.</td>
<td>Published</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
<td>Status</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>

I certify that I have obtained a written permission from the copyright owners to include the above published materials in my dissertation. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

III General

I declare that, to the best of my knowledge, my dissertation does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my dissertation, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owners to include such materials in my dissertation. I declare that this is a true copy of my dissertation, including any final revisions, as approved by my dissertation committee and the Graduate Studies office, and that this dissertation has not been submitted for a higher degree to any other University or Institution.
Abstract

These days, detection of Visual Attention Regions (VAR), such as moving objects, has become an integral part of many Computer Vision applications, viz. pattern recognition, object detection and classification, video surveillance, autonomous driving, human-machine interaction (HMI), and so forth. The moving object identification using bounding boxes has matured to the level of localizing the objects along their rigid borders and the process is called foreground localization (FGL). Over the decades, many image segmentation methodologies have been well studied, devised, and extended to suit the video FGL. Despite that, still, the problem of video foreground (FG) segmentation remains an intriguing task yet appealing due to its ill-posed nature and myriad of applications. Maintaining spatial and temporal coherence, particularly at object boundaries, persists challenging, and computationally burdensome. It even gets harder when the background possesses dynamic nature, like swaying tree branches or shimmering water body, and illumination variations, shadows cast by the moving objects, or when the video sequences have jittery frames caused by vibrating or unstable camera mounts on a surveillance post or moving robot. At the same time, in the analysis of traffic flow or human activity, the performance of an intelligent system substantially depends on its robustness of localizing the VAR, i.e., the FG. To this end, the natural question arises as what is the best way to deal with these challenges?

Thus, the goal of this thesis is to investigate plausible real-time performant implementations from traditional approaches to modern-day deep learning (DL) models for FGL that can be applicable to many video content-aware applications (VCAA). It focuses mainly on improving existing methodologies through harnessing multimodal spatial and temporal cues for a delineated FGL. The first part of the dissertation is dedicated for enhancing conventional sample-based and Gaussian mixture model (GMM)-based video FGL using probability mass function (PMF), temporal median filtering, and fusing CIEDE2000 color similarity, color distortion, and illumination measures, and picking an appropriate adaptive threshold to extract the FG pixels. The subjective and objective evaluations are done to show the improvements over a number of similar conventional methods.
The second part of the thesis focuses on exploiting and improving deep convolutional neural networks (DCNN) for the problem as mentioned earlier. Consequently, three models akin to encoder-decoder (EnDec) network are implemented with various innovative strategies to improve the quality of the FG segmentation. The strategies are not limited to double encoding - slow decoding feature learning, multi-view receptive field feature fusion, and incorporating spatiotemporal cues through long-short-term memory (LSTM) units both in the subsampling and upsampling subnetworks. Experimental studies are carried out thoroughly on all conditions from baselines to challenging video sequences to prove the effectiveness of the proposed DCNNs. The analysis demonstrates that the architectural efficiency over other methods while quantitative and qualitative experiments show the competitive performance of the proposed models compared to the state-of-the-art.
This thesis is dedicated to
my late mother, Vijayaluxmi.
I miss her every moment, but
I feel that she saw this process through to its completion,
offering eternal love and support from the nirvana.
Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor, Dr. Q.M. Jonathan Wu for giving me the opportunity to work under his supervision as well as for his guidance and continuous support for my Ph.D study and research. I also thank my committee members Dr. M. Khalid, Dr. H. Wu, and Dr. B. Boufama for guiding me and helping me with their invaluable comments. I heartily thank our department graduate secretary Ms. Andria Ballo, who has helped me in so many situations that I cannot possibly list in this limited space. Finally, I would like to convey my earnest gratitude to the Golden Key International Honour Society for bestowing me with the prestigious Golden Keys premier scholarship.

The person, who guided me to resume studies after nearly a decade of a debacle life, continuously provided support, is my cousin Dr. Esai Selvarasa. I would like to convey my sincere thanks to her. I could not have come this far without the help from my fellow lab members, and would like to thank them for supporting me. A special thank goes to the department technician Mr. Frank Cicchello, who helped me whenever I requested, even during his most busy hours.

I like to convey my regards, and sincere gratitude to my parents, my brother, sisters, cousins, and my in-laws. Under inexpressible circumstances and financial difficulties, they stand with me as strong pillars of support. I cannot thank them enough for their sacrifices and patience. I convey my gratitude to my father and late-mother for their continuous support and unfathomable trust. I would like to extend my sincere appreciation to Dr. Mohd Amaludin Yusoff for his motivational advice whenever I fell in the midst of confusion. His guidance helped me in all the time of research and life.

Finally, I thank the researchers, whose works were the stepping stones and guidelines for me to proceed working in my research. I convey my sincerest regards to Google, Wikipedia and the researchers around the world, helping us to free our minds, grow our knowledge and come out of the darkness of ignorance, false beliefs and judgments. They have helped me believe that this gradual progress will lead to self-awareness and help us make a better world.
# Table of contents

- Declaration of Co-Authorship / Previous Publication
- Abstract
- Dedication
- Acknowledgements
- List of tables
- List of figures
- List of abbreviation

## 1 Introduction
- 1.1 Overview
- 1.2 Motivation
- 1.3 Evaluation of foreground localization
- 1.4 Contributions to knowledge
- 1.5 Methodology
- 1.6 Research findings
- 1.7 Thesis structure

## 2 Background
- 2.1 Overview
- 2.2 Computer vision
- 2.3 Foreground
- 2.4 Background
- 2.5 Foreground localization
- 2.6 Background subtraction
- 2.7 Gaussian mixture models
- 2.7.1 Univariate Gaussian distribution
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.2</td>
<td>Threshold updating</td>
<td>51</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Foreground refinement</td>
<td>52</td>
</tr>
<tr>
<td>5.4</td>
<td>Experimental results</td>
<td>53</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Nature of the experiments</td>
<td>53</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Visual results</td>
<td>54</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Quantitative analysis</td>
<td>54</td>
</tr>
<tr>
<td>5.5</td>
<td>Conclusion</td>
<td>56</td>
</tr>
<tr>
<td>6</td>
<td>Fusion-based Foreground Enhancement for Background Subtraction</td>
<td>57</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary</td>
<td>57</td>
</tr>
<tr>
<td>6.2</td>
<td>Introduction</td>
<td>58</td>
</tr>
<tr>
<td>6.3</td>
<td>The proposed algorithm</td>
<td>59</td>
</tr>
<tr>
<td>6.3.1</td>
<td>The applied multivariate Gaussian mixture model</td>
<td>59</td>
</tr>
<tr>
<td>6.3.2</td>
<td>The proposed framework</td>
<td>60</td>
</tr>
<tr>
<td>6.3.2.1</td>
<td>Background estimation</td>
<td>61</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Color similarity measure</td>
<td>63</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Color distortion measure</td>
<td>64</td>
</tr>
<tr>
<td>6.3.5</td>
<td>FG feature enhancement and detection</td>
<td>65</td>
</tr>
<tr>
<td>6.3.6</td>
<td>Foreground validation</td>
<td>68</td>
</tr>
<tr>
<td>6.4</td>
<td>Experimental results</td>
<td>70</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Qualitative analysis</td>
<td>71</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Quantitative analysis</td>
<td>74</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Contribution of the features</td>
<td>76</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Fully utilized MVGMD</td>
<td>77</td>
</tr>
<tr>
<td>6.4.5</td>
<td>Limitations of the proposed model</td>
<td>78</td>
</tr>
<tr>
<td>6.5</td>
<td>Conclusion</td>
<td>80</td>
</tr>
<tr>
<td>7</td>
<td>sEnDec: An Improved Image to Image CNN for Foreground Localization</td>
<td>81</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary</td>
<td>81</td>
</tr>
<tr>
<td>7.2</td>
<td>Introduction</td>
<td>82</td>
</tr>
<tr>
<td>7.3</td>
<td>The proposed CNN architecture: sEnDec</td>
<td>84</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Deep slow encoding</td>
<td>86</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Deep slow decoding</td>
<td>86</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Training strategy</td>
<td>87</td>
</tr>
<tr>
<td>7.3.4</td>
<td>Binarization</td>
<td>89</td>
</tr>
</tbody>
</table>
List of tables

1.1 Confusion matrix .................................................................................. 4
4.1 Performance comparison of the proposed algorithms for WaterSurface dataset .............................................................................. 43
4.2 Performance comparison of the proposed algorithms for WavingTree dataset ................................................................................ 43
5.1 Description of the datasets used for validation ........................................... 53
5.2 Performance comparison. The best figures are in red ink. ......................... 55
6.1 Description of the video sequences used for validation ......................... 70
6.2 Average F-measure Comparison on Railway data sequence ................ 75
6.3 Average F-Measure Comparison On Watersurface Data Sequence .......... 75
6.4 Average F-Measure Comparison On Pedestrian, Highway, And Canoe Data Sequences ................................................................. 75
6.5 Average PTPF (in Sec.) Comparison On Pedestrian, Highway, And Canoe Data Sequences ................................................................. 76
6.6 Impact of fully utilized MVGMD in terms of FoM ................................. 78
7.1 The dataset pairs involved in model initialization ..................................... 88
7.2 The environment .................................................................................... 90
7.3 Dataset summary ................................................................................... 91
7.4 Sanity check results of the proposed sEnDec compared to the basic model ................................................................. 93
7.5 F-measure-based performance comparison: G-th, O-th, and OG-th stand for global, Otsu, and Gaussian smoothed Otsu threshold methods applied, respectively. The values in red and blue are the best and the second best figures, respectively ................................................................. 97
7.6 F-measure performance comparison: G-th, O-th, KI-th, and BGM stand for binarization methods with global, Otsu and Kittler-Illingworth thresholdings, and variational Bayesian Gaussian distribution, respectively. The values in red and blue are the best and the second best figures, respectively ................................................................. 103
8.1 Layer detail of the MV-FCN .................................................................. 110
8.2 Dataset summary ................................................. 111
8.3 Transfer learning detail for fine-tuning the proposed MV-FCN .......... 113
8.4 Results: impact of complimentary feature flows .............................. 116
8.5 Performance comparison of random vs transfer learning-based model initialization in terms of f-measure: S and P stand for type of training strategy, scratch and fine tuning pre-trained model. Global and Otsu threshold methods are referred by G-th and O-th respectively. .................... 118
8.6 F-measure performance comparison: S- training from scratch, P- pre-trained model fine-tuning, Global and Otsu stand for the two used thresholding methods. Values in red are the best figures while the ones in blue are the second best. ........................................... 124

9.1 Layer detail of the proposed 3D CNN-LSTM .................................. 134
9.2 Dataset summary .......................................................... 137
9.3 The dataset pairs used for model fine-tuning ................................. 139
9.4 Performance Comparison in terms of FoM: Global-th and Otsu-th stand for the two thresholding methods applied. Values in red are the best FoM while the ones in blue are the second best ........................................... 144

10.1 Publication summary since 2015 ............................................... 152
List of figures

1.1 Examples for video surveillance camera placements. .................................. 2
1.2 Applications of video cameras: (a) Almotive’s customized Toyota Prius

has cameras in front, on its sides, and in back. (b) Onboard cameras

are key to developing truly independent robots. ........................................... 2
1.3 A probable application environment of foreground localization. .............. 3

2.1 A background subtraction paradigm. ..................................................... 11
2.2 Plots of a bivariate normal distribution. .............................................. 13
2.3 A Venn diagram showing how deep learning (DL) is a kind of representa-

tion learning, which is in turn a part of machine learning (ML). ............... 15
2.4 Deep learning to convolutional neural network. .................................... 16
2.5 An example of CNN architecture: A handwritten digit classifier .......... 16
2.6 Convolution feature map generation in LeNet5 [77]. ............................ 18
2.7 An example of top fully connected layer. ............................................. 18
2.8 Block diagram describing transfer learning technique. ........................ 22

3.1 A scene and its useful region of interest. .............................................. 24
3.2 Categories of foreground localization algorithms. .................................. 25
3.3 CNN feature flows: (a) ResNet flow, and (b) the residual feature mapping

of our models. ......................................................................................... 33
3.4 U-net architecture. Blue boxes correspond to a multi-channel feature map.

The number of channels is denoted on top of the boxes. The x-y-size is

provided at the lower left edge of the box. White boxes represent copied

feature maps and the arrows refer to the different type of operations [119]. 34
3.5 The network is trained with two small patches extracted from the in-

put and background images in gray-scale. The network is inspired by

LeNet-5 network [17]. ........................................................................... 36

4.1 An example for PMF computation: (a) Collected samples, (b). Splitting

color channels, (c). Extracted red channel of a scene over a period \(N\)

and (d) PMF of a pixel at \((u,v)\) observed over the same period \(N\). ........ 40
4.2 3D Geometrical representation of (a). Point $P$, (b). Point $P'$, and (c) The distance between $P$ and $P'$ ................................................. 40

4.3 Example of 3D-color space .......................................................... 41

4.4 Sample results for the WaterSurface dataset .................................. 43

4.5 Sample results for the WavingTree dataset .................................... 43

4.6 Average performance comparison across all the color-spaces and datasets. 44

5.1 Process-flow of the proposed algorithm ........................................ 48

5.2 Unclassified FG Region ............................................................... 51

5.3 Visual comparisons of the results for frames no. 250 - 1st row and no. 559 - 2nd row form datasets (a) and (b), respectively .......................... 54

5.4 Visual comparisons of the results for frames 435 - 1st row, 626 - 2nd row, and 827 3rd row from datasets (c) - (e) respectively .......................... 54

5.5 Performance comparison ............................................................. 55

6.1 A scene and its useful region of interest .......................................... 58

6.2 First level of foreground features .................................................. 61

6.3 Fused foreground features and its corresponding intensity histogram ............................................. 61

6.4 Step diagram of color similarity calculation through CIEDE2000. ....... 64

6.5 Optimal threshold adjustment in the histogram of enhanced FG features .................................................. 66

6.6 Detected FG with optimal threshold .............................................. 68

6.7 Validated foreground ................................................................. 68

6.8 Process block diagram of the proposed algorithm ............................ 68

6.9 Comparisons of foreground extraction for Railway data sequence: (a). original scene, (b). ground truth, and (c) to (i) show the detected FG of different methods GMM, EGMM, BOD, S-SVM, DECOLOR, Ours, and Ours$^{K^{th}}$ respectively ........................................... 72

6.10 Comparisons of FG separation for WaterSurface data sequence: (a). original scene, (b). ground truth, and (c) to (i) show the detected FG of different methods GMM, EGMM, FCH, RPCA, DECOLOR, Ours, and Ours$^{K^{th}}$ respectively ........................................... 72

6.11 Comparisons of FG separation on Pedestrians data sequence: Columns (a) to (h) show original significant scenes with frame IDs embedded, ground truth, and detected FG of different methods CRF, EGMM, GMM, wavGMM, Ours, and Ours$^{K^{th}}$ respectively ........................................... 73
6.12 Comparisons of FG separation on Highway data sequence: Columns (a) to (h) show original significant scenes with frame IDs embedded, ground truth, and extracted FG of different methods CRF, EGMM, GMM, waVGMM, Ours, and Ours$^{K^1th}$ respectively. ........................................ 74

6.13 Comparisons of FG separation Canoe data sequence: Columns (a) to (h) show original significant scenes with frame IDs embedded, ground truth, and detected FG of different methods CRF, EGMM, GMM, waVGMM, Ours, and Ours$^{K^1th}$ respectively. ........................................ 74

6.14 F-Measure comparison on various input feeds vs methods. .............................................................. 76

6.15 Processing time comparison on various input feeds vs methods. .......................................................... 77

6.16 Processing time of the proposed algorithm vs input feeds. ................................................................. 77

6.17 FoM error bar vs level of fused FG features. .......................................................................................... 78

6.18 BusStation frame no. 1096: The proposed model fails to distinguish moving shadow and sudden region level changes. ................................................................. 79

6.19 Bungalows frame no. 650: The proposed model fails to distinguish the moving shadow. ......................... 79

7.1 Block diagram of a basic image-to-image CNN. ...................................................................................... 82

7.2 Schematic drawing of the proposed DCNN: sEnDec. .............................................................................. 84

7.3 Illustration of convolution and transpose (de) convolution operations: A 2D conv with $K = 3, S = 2, $ and $P = 1$, and its corresponding convT with $K' = K, S' = 1$, and $P' = K - P - 1$. ........................................ 85

7.4 The exploited residual feature flow and resolution recovery mechanism. .............................................. 87

7.5 Input data configuration: G.T - Ground truth, I - Image, Med$(\cdot)$ - Median filtering in temporal domain over collected $k$ samples prior to training, $b_0$ - The precomputed BG model, and $f_t$ - Input scene at time $t$. .... 88

7.6 Random vs ordered splits: G.T - ground truth, I - raw image, Sq.# - sequence ID. ................................ 92

7.7 Bar chart comparison- sanity test results of the sEnDec and basic model: F-measure vs video sequences. ................................................................. 93

7.8 Sample results. Row 1-8: Highway, Office, Pedestrians, PETS2006, Canoe, Boats, OverPass, and Fall video sequences. Col. 1-5: input frames, ground truths, sEnDec’s score-maps, binary FG masks generated with G-th and O-th, respectively. ........................................ 94
7.9 Sample results. Row 1-8: Traffic, Boulevard, CopyMachine, PeopleInShade, BusStation, Sofa, Tramstop, and SnowFall video sequences. Col. 1-5: input frames, ground truths, sEnDec's score-maps, binary FG masks generated with G-th and O-th, respectively. 95

7.10 Sequence category-wise performance analysis. 96

7.11 Overall performance analysis of various models. 98

7.12 F-measure vs model: Performance analysis across all the video sequences. 99

7.13 F-measure vs binarization method. 100

7.14 Processing time vs binarization method. 100

7.15 FGL speed of sEnDec including binary mask creation. 101


8.1 The ResNet-like and Inception-like modules. 108

8.2 Layer schematic of the MV-FCN: Convk, Si, CTransk, Concat, and BN stand for convolution using kernel size of k and stride of i, transpose convolution with filter size of k, activation maps concatenation, and batch normalization operations, respectively. 109

8.3 Ordered exclusive split of training and test sets: G.T- ground truth, I-RGB raw input image, S.q#- sequence ID. 112

8.4 Creating FG mask: Applying an appropriate threshold to the score-map generated from the last classification layer of MV-FCN for a frame taken from the Office dataset. 114

8.5 Network configurations for investigating impact of complimentary feature flows. 117

8.6 Visualizing impact of complimentary feature flows: (a) Input frame, (b) - (e) are the salient map of PFF, combined feature flow of PFF & CFF2, PFF & CFF1, and PFF & CFF1 & CFF2, respectively. 118

8.7 Performance vs model initialization in terms of f-measure: S and P stand for type of training strategy, scratch and fine tuning pre-trained model. Global and Otsu threshold methods are referred by G-th and O-th respectively. 119

8.8 MV-FCN salient map when: (b) trained from scratch, (c) fine-tuned with intra-class transfer learning. 119
8.9 Office dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: training and validation FoM and loss respectively in the top and bottom.

8.10 Overpass dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: Training and validation FoM and loss respectively in the top and bottom.

8.11 Traffic dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: Training and validation FoM and loss respectively in the top and bottom.

8.12 PeopleInShade dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: Training and validation FoM and loss respectively in the top and bottom.

8.13 TwoPositionPTZCam dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: Training and validation FoM and loss respectively in the top and bottom.

8.14 Turnpike.0.5fps dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: Training and validation FoM and loss respectively in the top and bottom.

8.15 Performance comparison: F-measure vs method.

8.16 Best performance comparison: F-measure vs dataset.

8.17 Inferencing speed of the proposed MV-FCN.

9.1 Traffic flow and its foreground (Input frame, Ground truth, 3D CNN-LSTM score-map, and Predicted FG mask).

9.2 An overview of the proposed CNN-LSTM image-to-image network: \( E(\cdot) \) - binary cross-entropy error.

9.3 A layer-wise schematic of the proposed 3D CNN-LSTM. It exploits 3D conv embedded with 2D LSTM both in the encoding and decoding phases, and residual layer concatenation and 3D transpose conv layers only in the decoding subnetwork.

9.4 Simplified 3D CNN-LSTM model with single stage EnDec.

9.5 A standard LSTM module with three gates.
9.6  Sequence generation for the 3D CNN-LSTM model. ..................... 138
9.7  Qualitative analysis on Highway video sequence. ...................... 141
9.8  Qualitative analysis on Office video sequence. ......................... 141
9.9  Qualitative analysis on Canoe video sequence. ......................... 141
9.10 Qualitative analysis on Boats video sequence. ......................... 142
9.11 Qualitative analysis on Overpass video sequence. ..................... 142
9.12 Qualitative analysis on Boulevard video sequence. .................... 142
9.13 Qualitative analysis on CopyMachine video sequence. ............... 143
9.14 Performance vs dataset. .................................................. 145
9.15 Overall performance comparison. ....................................... 146
# List of abbreviation

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D</td>
<td>One-Dimension or One-Dimensional</td>
</tr>
<tr>
<td>2D</td>
<td>Two-Dimension or Two-Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three-Dimension or Three-Dimensional</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>BN</td>
<td>Batch Normalization</td>
</tr>
<tr>
<td>BG</td>
<td>Background</td>
</tr>
<tr>
<td>BGS</td>
<td>Background Subtraction</td>
</tr>
<tr>
<td>BOD</td>
<td>Bayesian Object detection in Dynamic scenes</td>
</tr>
<tr>
<td>BRDL</td>
<td>Background subtraction via Robust Dictionary Learning</td>
</tr>
<tr>
<td>CEC</td>
<td>Constant Error Carousel</td>
</tr>
<tr>
<td>CFF</td>
<td>Complementary Feature Flows</td>
</tr>
<tr>
<td>CNN or ConvNet</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CRF</td>
<td>Conditional Random Field</td>
</tr>
<tr>
<td>CTC</td>
<td>Connectionist Temporal Classification</td>
</tr>
<tr>
<td>CTrans</td>
<td>Transpose Convolutional</td>
</tr>
<tr>
<td>CV</td>
<td>Computer Vision</td>
</tr>
<tr>
<td>DBFCN</td>
<td>Deep Background Fully Convolutional Network</td>
</tr>
<tr>
<td>DCNN</td>
<td>Deep Convolutional Neural Network</td>
</tr>
<tr>
<td>DConv</td>
<td>Deconvolution</td>
</tr>
<tr>
<td>CD</td>
<td>Change Detection</td>
</tr>
<tr>
<td>DESD</td>
<td>Double Encoding Slow Decoding</td>
</tr>
<tr>
<td>DL</td>
<td>Deep Learning</td>
</tr>
<tr>
<td>DPMM</td>
<td>Dirichlet Process Mixture Models</td>
</tr>
<tr>
<td>EnDec</td>
<td>Encoder-Decoder</td>
</tr>
<tr>
<td>FCH</td>
<td>Fuzzy Color Histogram</td>
</tr>
<tr>
<td>FCN</td>
<td>Fully Convolutional Network</td>
</tr>
<tr>
<td>FG</td>
<td>Foreground</td>
</tr>
<tr>
<td>FGL</td>
<td>Foreground Localization</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>FoM or F-measure</td>
<td>Figure of Merit</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>FOV</td>
<td>Field of View</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>FPS</td>
<td>Frame Per Second</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model</td>
</tr>
<tr>
<td>GP</td>
<td>Genetic Programming</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphic Processing Unit</td>
</tr>
<tr>
<td>GT</td>
<td>Ground Truth</td>
</tr>
<tr>
<td>GTI</td>
<td>Generalized Tverksy Index</td>
</tr>
<tr>
<td>HMI</td>
<td>Human Machine Interaction</td>
</tr>
<tr>
<td>HPC</td>
<td>High Performance Computing</td>
</tr>
<tr>
<td>ILSVRC</td>
<td>Large-Scale Visual Recognition Challenge</td>
</tr>
<tr>
<td>IoU</td>
<td>Intersection Over Union</td>
</tr>
<tr>
<td>KDE</td>
<td>Kernel Density Estimation</td>
</tr>
<tr>
<td>LBP</td>
<td>Local Binary Pattern</td>
</tr>
<tr>
<td>LBSP</td>
<td>Local Binary Similarity Pattern</td>
</tr>
<tr>
<td>LRE</td>
<td>Local Refinement Error</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long-Short Term Memory</td>
</tr>
<tr>
<td>LTP</td>
<td>Local Ternary Pattern</td>
</tr>
<tr>
<td>MCR</td>
<td>Missed-classification Ratio</td>
</tr>
<tr>
<td>MKFC</td>
<td>Multi-channel kernel fuzzy Correlogram</td>
</tr>
<tr>
<td>MP</td>
<td>Mega-Pixel</td>
</tr>
<tr>
<td>MRF</td>
<td>Markov Random Field</td>
</tr>
<tr>
<td>MRI</td>
<td>Magnetic Resonance Imaging</td>
</tr>
<tr>
<td>MVGMM</td>
<td>Multivariate GMM</td>
</tr>
<tr>
<td>MVGMD</td>
<td>Multivariate Gaussian Model Distribution</td>
</tr>
<tr>
<td>MV-FCN</td>
<td>Multi-View receptive field FCN</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Networks</td>
</tr>
<tr>
<td>ORDL</td>
<td>Online Robust Dictionary Learning</td>
</tr>
<tr>
<td>OR-PCA</td>
<td>Robust Online PCA</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PBAS</td>
<td>Pixel-Based Adaptive Segmenter</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>PFF</td>
<td>Pivotal Feature Flow</td>
</tr>
<tr>
<td>PRI</td>
<td>Probabilistic Rand Index</td>
</tr>
<tr>
<td>PTPF</td>
<td>Processing Time Per Frame</td>
</tr>
<tr>
<td>Acronym</td>
<td>Abbreviation</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RGB</td>
<td>Red Green Blue</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>ResNet</td>
<td>Residual Network</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>RRC</td>
<td>Radial reach correlation</td>
</tr>
<tr>
<td>RPCA</td>
<td>robust principal component analysis</td>
</tr>
<tr>
<td>ROI</td>
<td>Region of Interest</td>
</tr>
<tr>
<td>SACON</td>
<td>SAmple CONsensus</td>
</tr>
<tr>
<td>SDAE</td>
<td>Stacked Denoising Auto-Encoder</td>
</tr>
<tr>
<td>sEnDec</td>
<td>Slow Encoder-Decoder</td>
</tr>
<tr>
<td>SOM</td>
<td>Self-Organizing Map</td>
</tr>
<tr>
<td>SuBSENSE</td>
<td>Self-Balanced SENsitivity SEgmenter</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>TL</td>
<td>Transfer Learning</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>VAR</td>
<td>Visual Attention Region</td>
</tr>
<tr>
<td>VCA</td>
<td>Video Content Analysis</td>
</tr>
<tr>
<td>VCAP</td>
<td>Video Content Aware Processing</td>
</tr>
<tr>
<td>VCAA</td>
<td>Video Content Aware Applications</td>
</tr>
<tr>
<td>ViBe</td>
<td>Visual Background extractor</td>
</tr>
<tr>
<td>VoI</td>
<td>Variation of Information</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Overview

This chapter elaborates the motivation, aims, and objectives of the research, and provides the background information of foreground localization. Furthermore, the key contributions and methodology are briefly disclosed. Finally, the thesis structure is outlined.

1.2 Motivation

The advancement in the state-of-the-art technology for visual acquisition has resulted in low-cost digital video recording devices, and the progress in High-Performance Computing (HPC) for processing a large volume of data has promoted the installation of cameras everywhere, on fixed and moving platforms. According to a study, there were more than 245 million professionally installed video surveillance cameras globally used in a daily basis in 2014 [96]. In year 2016, it leaped by 100 million to an estimated 350 million, including an approximate of 62 million surveillance cameras in North America [97]. In China, at the same time, there are 176 million surveillance cameras active and the country extends them to 570 million, by 2020 that’s nearly one camera for every two citizens [142]. According another study in the late 2017, the total number of cameras in the world will reach about 45 billion that includes few billions of cell phones equipped with cameras, autonomous cars, security cameras, and smart home products, and so forth by 2022 [75]. The overwhelming volume of data captured through such deluge of devices raises the question of how to monitor and analyze the information? Humans cannot efficiently perform these tasks over such prevalence of information because after 20 minutes of monitoring a human operator misses 90% of activity on the screen. It urges to have intelligent systems that are
capable of automatically extracting application-specific information, for instance, the presence of an object, in the scene being monitored. The system can be a tool to help humans to perform intriguing and time-consuming tasks and to maintain the efficiency of video-based applications by processing only relevant detail. Most video data have redundant information, such as background information, which costs a massive volume of storage and computing resources.

Moreover, the scenes monitored exhibit illumination changes, motion changes, secondary illumination effects cast by the moving objects, and random pixel intensity variations due to capturing devices. To this end, what is the best way to deal with these challenges? At the same time, in the analysis of traffic flow or human activity, the performance of an intelligent system substantially depends on its robustness of foreground localization [170].

The goal of this thesis is to improve the fundamental algorithms and introduce new deep learning architectures for foreground localization that can be applicable to many Video Content Aware Applications (VCAA).

Figure 1.1: Examples for video surveillance camera placements.

Figure 1.2: Applications of video cameras: (a) AI motive’s customized Toyota Prius has cameras in front, on its sides, and in back. (b) Onboard cameras are key to developing truly independent robots.
The applications of FGL is myriad as a probable application environment depicted by Fig. 1.3. For example, the FGL-based automatic people and vehicle detection, counting, and tracking system can address the following sectors:

- **Object recognition**: In general, recognition identifies an object to be part of a group of objects. This process can be carried out by object segmentation and extraction followed by a classification task. For instance, in case of traffic signal recognition, where the traffic signal is segmented into a single object based on some primitive features, such as color intensities and texture information; later, it is recognized to identify the class of traffic signals. Thus, such systems require FG segmentation mechanism to locate the specific object in a video frame or image.

- **Security**: Many industries or public facilities such as airports are interested in locating and tracking people to monitor human presence in forbidden areas, like borders crossings, or people walking in wrong directions, and suspicious activities can be automatically detected and shared with security agents.

- **Human gesture or action recognition**: It is a rapidly growing application area of foreground detection/localization. The FG identification is employed at an early stage to extract the human or simply the gesture to process towards the recognition or classification stage \[124,125\].

- **Safety**: Anomaly detection in public areas, such as parks, lounges, railway stations, or bars to detect abandoned objects, and to estimate crowd sizes for safety purposes.
• **Data compression:** Content-aware data compression that keeps intact the essential foreground objects while applying higher data compression rate on the background for big video data.

• **Optimization:** Based on the detected FG, a system can perform object indexing for statistical analysis on the people on a platform, for instance, to optimize flows in railways. It can help to answer - How many people are waiting? How long? Moreover, where are they usually going? Thus, peak times can be identified to optimize operation.

• **Autonomous driving and traffic safety:** It assists in getting accurate visual perception around a vehicle resulting in improved road traffic safety and autonomous navigation.

### 1.3 Evaluation of foreground localization

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Actual Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP</td>
<td>FN</td>
<td></td>
</tr>
<tr>
<td>Negative</td>
<td>FP</td>
<td>TN</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1: Confusion matrix.

There are many objective evaluation methods, like Local Refinement Error (LRE), Missed-classification Ratio (MCR), Probabilistic Rand Index (PRI), Variation of Information (VoI), and F-measure. Among them, the f-measure is the most widely accepted and standardized method, which is a weighted harmonic mean measure of recall and precision, i.e., size of the intersection divided by the union of the two regions. It is also referred as Intersection-over-Union (IoU) or Figure-of-Merit (FoM) and defined as (1.1).

\[
F \text{- measure} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}, \quad (1.1)
\]

where recall is the detection rate and precision is the percentage of correct prediction compared to the total number of detections as positives. The recall and precision are given by (1.2), where TP, FN, and FP refer true positive, false negative, and false positive, respectively as described by the confusion matrix in Table 1.1.

\[
\text{Recall} = \frac{TP}{TP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP} \quad (1.2)
\]

This dissertation utilizes the f-measure for quantitative analysis.
1.4 Contributions to knowledge

A reliable FG localization algorithm should be robust and able to handle sudden and gradual illumination changes, high frequency moving objects, repetitive motion in the background (such as tree leaves, flags, waves in the sea or lake, etc.) and long-term scene changes (a car is parked for a month, for instance). Thus, there have been many algorithms proposed fundamentally based on GMM by the CV community over the past two decades since the pioneer work reported by Stauffer and Grimson [139]. For instance, Effective GMM [78], GMM-based conditional random filed [159,160], Variational clustered GMM [24], and Wavelet transformation-based GMM [91]. However, such high complexity algorithms are not necessary for specific surveillance purposes, including monitoring an automatic teller machine (ATM) in a shopping complex or bank. Because in such cases, the surveillance camera is fixed at a place and the background environment is known prior to monitoring. In such conditions, it is recommended to employ simplistic models to the moving objects in the given environment being monitored. To this end, this thesis, firstly, introduces two simplistic algorithms: a probabilistic based model with a non-supervised threshold and 3D-color space model using distance vector for constrained environment BGS. The outcomes of these models have been published in [144].

Although the multivariate GMM is well exploited for BGS in [139], the threshold value used to separate BG and FG in such algorithms requires tuning for every new video feed. In order to circumvent this, this thesis takes advantage of color similarity, color distortion, and illumination measures for an effective FG enhancement. The color similarity and distortion measures have been used in industries to measure color differences of fabricated materials. We explore another application of the measures in the framework of BGS by utilizing them to enhance the FG features and to find appropriate thresholds. The concept of using such measures is plausible since traditional GMM based BGS techniques tend to miss-classify pixels when the FG and BG pixels are similar in color or distorted by the brightness and illumination variation or by the capturing technology (including camera motion). We have successfully utilized the aforesaid measures in [143] to estimate an appropriate threshold through empirical mathematical derivations, which controls the minimum amount of data that model the BG. This basic model then has gone through further improvement, whereby it takes advantage of feature fusion to enhance the primary FG feature and then uses a histogram-based strategy to find an optimal threshold. The threshold is then employed to extract a final refined FG. Thus, the proposed algorithm consists of three
major processing stages: BG estimation and FG detection, FG feature enhancement, and FG refinement. The outcomes of the improved model have been published in [3].

It is also very crucial to adopt the cutting-edge CV technologies, like DL for FGL. Deep learning networks have been successfully applied to big data for knowledge discovery, knowledge application, and knowledge-based prediction. Consequently, the deep CNN has become a driving force of the modern era autonomous driving, video surveillance, drug and food inspection, and so forth. Thus, this thesis also extends the work on utilizing DL for FG. It harnesses the power of CNN-based image semantic segmentation ideas for video foreground localization. It improves the basic EnDec CNN through the following innovative approaches:

1. Slow encoder-decoder CNN with micro auto-encoder blocks, batch normalization, and channel-wise residual feature concatenation.
2. Multi-view receptive field to capture scale-invariant features of FG objects.
3. CNN-LSTM model to exploit spatiotemporal cues for delineate FGL.

Finally, this thesis analyses the performances of the proposed approaches on publicly available benchmark data-sets for video FGL. All the findings at every stage of the research have been published through or submitted to internationally recognized IEEE conferences, journals, and transactions.

1.5 Methodology

The following research methodology is pursued for this doctorate:

- Literature review: An extensive background study was conducted on the existing works in the area of foreground localization. Research articles and papers published by official international conferences and journals were reviewed in depth to understand the research carried out so far in the field of FGL to find research gaps and to set the milestones of this thesis. The papers studied are mainly from IEEE, ACM, IET, Elsevier, and Springer.

- Foundation: Various texts and online materials on image and video processing, neural networks and deep learning were studied for an understanding of the critical theoretical and practical methodologies towards FGL. The study also covered associated topics, including histograms, adaptive threshold, morphological operations for noise reduction, and deep CNNs. The texts are ranging from
the classical image processing book, Digital Image Processing by G. Woods [43] to online resources, the Stanford CS231n - Convolutional Neural Networks for Visual Recognition [38].

- Programming paradigm: Python with Keras using Tensorflow backend and MATLAB were used to develop algorithms and DL models for FGL and to test and verify the output of the proposed logic(s).

- Data-sets: In order to test the algorithms and models, publicly available benchmark video sequences with varying challenges are used. Moreover, the results achieved are analyzed with ground-truths and quantitatively compared to literature results using standard measurement matrices. The performance measures indicate the strength and weakness of the proposed algorithms and models over the existing works by other researchers.

- Dissemination: The outcomes are presented at and/or disseminated through IEEE international conferences, transactions, and international journals, like IET, and Springer whereby the proposed work are reviewed by experts in the field. Experts’ feedback is one of the rich sources towards the advancement of knowledge that is used for continuous improvement of the proposed methods.

- Knowledge exchange: We conduct regular seminars with our peers at our computer vision and sensing systems laboratory (CVSSL) and meet experts at various conferences as another source of gaining knowledge.

1.6 Research findings

During this doctorate research the proposed methodologies and related works have been published and/or presented in international IEEE conferences, journals, and transactions. The list of publications as follows:


• T. Akilan, and QMJ Wu, Double encoding - Slow decoding image to image CNN for foreground identification with application towards intelligent transportation, IEEE GreenCom 2018 (Submitted: #1570441560).


• T Akilan, QMJ Wu, et al., Effect of fusing features from multiple dcnn architectures in image classification, IET Image Process., 2018.


• T Akilan, QMJ Wu, et al., Video foreground detection in non-static background using multi-dimensional color space, Procedia Computer Science 70, 55-61, 2015.

1.7 Thesis structure

This thesis consists of ten chapters, initially with this introductory chapters, which provide a concise synopsis of the work. Gradually, it spans out on proposed algorithms and models. Finally, it concludes with discussions and directions for future work. It is structured as follows:

Chapter 1 discusses the motivation, aims and objectives, and research methodology. Chapter 2 discloses briefly the key background details from conventional CV perspective to modern deep DL with definitions, equations, derivations, and expositions. Chapter 3 reviews the classical and modern foreground detection/localization algorithms discovered in the literature. It is dedicated to brushing up the present and past
contributions in the field. However, in the core chapters also the relevant literature reviews are given in concise form whenever it is possible to enhance the understanding of the proposed ideas.

Chapter 4 proposes two sample-based FGL algorithms using probability mass function with a non-supervised threshold computation and a 3D-color space model using distance vector for background suppression. Empirical study is carried out with various color spaces, like RGB, YCbCr, YIQ, and YUV.

Chapters 5 and 6 intent to present novel ideas to update the threshold of GMM-based BGS with respect to color distortion, similarity and illumination measures in pixel-level. These cues are interesting as the CIDE2000 color similarity and color distortion have not been used for foreground localization by the CV community so far.

Chapter 7 introduces two elegant ideas to improve the learning ability of a basic image-to-image CNN network through micro-auto-encoder blocks in the subsampling subnetwork and slow decoding blocks in the upsampling subnetwork.

Chapter 8 introduces akin to Inception modules with multi-view receptive field to capture scale invariant FG clues. And to capture the spatiotemporal features, it uses a temporally median filtered BG model stacked as the third channel of input data that takes two consecutive frames as the first two channels.

Chapter 9 harnesses the ability of LSTM modules in handling time series data. It implements a 3D CNN-LSTM network that takes four frames at a time to predict the FG region in the current frame.

Chapter 10 concludes the thesis with overall discussions and intuitive directions for future work.
Chapter 2

Background

2.1 Overview

This chapter lays a strong foundation of the dissertation by providing the definitions, mathematical derivations, and background information on the key topics relevant to the work presented.

2.2 Computer vision

Computer vision is the science that focuses on the ultimate goal of creating a similar, if not better, the capability of the human vision system (the way eyes and brain work together) to a machine or computer. The core components of CV are: an automatic feature extraction, analysis and understanding of useful detail from a single visual or a sequence of visuals, i.e., videos. The development of the core units requires a depth knowledge of theoretical and algorithmic basis of signal processing and artificial intelligence or neural networks.

2.3 Foreground

In a formal dictionary definition, the part of a view that is nearest to the observer, especially in a picture or photograph. However, in this thesis context, the pixel or group of pixels that represent moving object(s) in a scene monitored.

2.4 Background

In a formal dictionary definition, the area or scenery behind the primary object of contemplation, especially when perceived as a framework for it. However, in this
thesis context, the pixels that are not part of FG region or the pixels that represent near static objects in a scene monitored.

2.5 Foreground localization

Locating the moving objects with a tight binary mask in video sequences that are captured from stationary/non-stationary cameras. It is a core task in computer vision for robust video surveillance applications.

2.6 Background subtraction

It is a technique in the field of computer vision, wherein the foreground FG$_t$ in a current scene $I_t$ is extracted by subtracting it from its known background BG$_t$ and regulated by a threshold $\tau$ as:

$$FG_t = \text{abs}(I_t - BG_t) > \tau$$  \hspace{1cm} (2.1)

The process is then described via flow diagrams in Fig. 2.1, where the $p_t$, $F_t$, and $M_t$ represent values of the new pixel, FG pixel, and the BG model pixel at the time $t$. Then, the extracted FG then can be used for high-level analysis, like object recognition. Generally, in a scene the region of interest (ROI) are objects, such as humans, vehicles, etc. in its FG.

Figure 2.1: A background subtraction paradigm.

2.7 Gaussian mixture models

It is a probabilistic model that considers all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown trainable param-
eters. The following subsections provide clear derivation of multivariate Gaussian distributions from univariate case.

### 2.7.1 Univariate Gaussian distribution

The univariate (1D) distribution is the normal distribution introduced by Johann Carl Friedrich Gauss in early 19th century. It is given by:

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right). \tag{2.2}$$

Where, $-\infty < x < \infty$, $\mu$ is the mean of the distribution, $\sigma$ is the standard deviation and the exponential term, $-\frac{1}{2\sigma^2}(x - \mu)$, is a quadratic function of $x$. Thus, as the coefficient of the quadratic term is negative the parabola points downwards. The $\frac{1}{\sqrt{2\pi\sigma^2}}$ is a normalizing factor to ensure that:

$$\frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right) = 1. \tag{2.3}$$

### 2.7.2 Multivariate distribution

In multivariate distribution, the exponential term in the univariate distribution (2.2) is replaced with the following quadratic form:

$$\exp\left[-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right], \tag{2.4}$$

where $x$ is a vector-valued random variable, $X = [X_1...X_n]^T$, $\mu \in \mathbb{R}^n$, and covariance matrix $\Sigma \in S_{++}^n$. The $S_{++}^n$ is the space of symmetric positive definite matrices, defined by: $S_{++}^n = \{A \in \mathbb{R}^{n \times n} : A = A^T$ and $x^T A x > 0$ for all $x \in \mathbb{R}^n$ such that $x \neq 0\}$. Therefore, for any non-zero vector $z$, $z^T \Sigma^{-1} z > 0$ and it satisfies (2.5) for any vector $x \neq \mu$.

$$\begin{align*}
(x - \mu)^T \Sigma^{-1} (x - \mu) &> 0 \\
-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) &< 0.
\end{align*} \tag{2.5}$$

Thus, the multivariate Gaussian distribution is written in the form:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right]. \tag{2.6}$$

Similar to the univariate distribution the coefficient $\frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}}$ is the normalizing factor to ensure that;
Figure 2.2: Plots of a bivariate normal distribution.

\[
\frac{1}{(2\pi)^{n/2}} \frac{1}{|\Sigma|^{1/2}} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp(\Delta) \, dx_1 dx_2 \cdots dx_n = 1,
\]

(2.7)

Where, \( \Delta = -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \). Figure 2.2 shows an example of a bivariate normal joint density of two independent (i.e. \( \Sigma \) is a diagonal matrix) variables \( X_1 \) and \( X_2 \) where, \( \mu_1 = \mu_2 = 0 \) and \( \sigma_1^2, \sigma_2^2 \) are 0.25 and 1.0 respectively.

### 2.7.2.1 The covariance matrix

The covariance matrix \( \Sigma \) of multiple variables play a center role in multivariate distribution. For a pair of random variables \( X \) and \( Y \), their covariance matrix is defined as;

\[
\Sigma[X,Y] = E[(X - E[X])(Y - E[Y])]
= E[XY] - E[X]E[Y],
\]

(2.8)

where \( E[i] \) is the expectation of \( i \), \( \Sigma \) is an \( n \times n \) matrix whose \( (p,q) \)th entry is \( \text{Cov}[X_p,X_q] \) and \( n \) is the number of random variables involve in the distribution. From (2.8) for any random vector \( X \in R^n \) with mean \( \mu \in R^n \) the \( \Sigma \) can be defined as;

\[
\Sigma[X,Y] = E[(X - \mu)(X - \mu)^T]
= E[XX^T] - \mu\mu^T.
\]

(2.9)

The \( \Sigma \) must be invertible in order for existence of \( \Sigma^{-1} \) as it is required in (2.7) of the multivariate Gaussian distribution. It follows that must be full rank and symmetric positive definite as well.
2.7.2.2 Generalization

In order to achieve a simplified multivariate model, which can easily be computed and implemented for real-time video FG classification the following derivations are necessary. The derivations are given from a bivariate case and then extended to general multivariate case. Let assume the bivariate random variables \( x_1 \) and \( x_2 \) are independent and \( x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \), \( \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \), \( \Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \) so that the joint probability density function of \( x_1 \) and \( x_2 \) is computed by using (2.6) as;

\[
p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{1/2}} \frac{1}{\sqrt{|\Sigma|}} \exp \left( -\frac{1}{2} \Psi^T \Sigma^{-1} \Psi \right),
\]

where \( \Psi = \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix} \), the inverse of the covariance matrix \( \Sigma^{-1} \) is nothing but taking reciprocal of the entries and \( |\Sigma|^{1/2} = (\sigma_1^2 \sigma_2^2)^{1/2} \) since \( \Sigma \) is a diagonal matrix. Calculation of the inverse covariance matrix for a bi-variate case is given below.

\[
\Sigma^{-1} = \frac{\text{Adj}(\Sigma)}{\det(\Sigma)} = \frac{(\text{cofactor } \Sigma)^T}{\det(\Sigma)},
\]

where, the cofactor of \( \Sigma \) is defined as \( c\Sigma \);

\[
c\Sigma = \begin{bmatrix} (-1)^{1+1} \sigma_2^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix} = \begin{bmatrix} \sigma_2^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix}
\]

and \( \det(\Sigma) = (\sigma_1^2 \cdot \sigma_2^2 - 0 \cdot 0) = \sigma_1^2 \cdot \sigma_2^2 \). Thus, \( \Sigma^{-1} \) is given by;

\[
\Sigma^{-1} = \begin{bmatrix} \sigma_2^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix} \times \frac{1}{\sigma_1^2 \sigma_2^2} = \begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix}.
\]

In general, case for an \( n \)-dimensional Gaussian with mean \( \mu \in \mathbb{R}^n \) and covariance matrix \( \Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, ..., \sigma_n^2) \) the determinant \( |\Sigma| = \sigma_1^2 \times \sigma_2^2 \times ... \times \sigma_n^2 \) and the inverse covariance matrix \( \Sigma^{-1} = \text{diag} \left( \frac{1}{\sigma_1^2}, \frac{1}{\sigma_2^2}, ..., \frac{1}{\sigma_n^2} \right) \). Thus, (2.10) is further reduced to (2.14).

\[
p(x; \mu, \Sigma) = \frac{1}{\sqrt{2\pi\sigma_1}} e^{-\frac{1}{2} \left( \frac{x_1 - \mu_1}{\sigma_1} \right)^2} \times \frac{1}{\sqrt{2\pi\sigma_2}} e^{-\frac{1}{2} \left( \frac{x_2 - \mu_2}{\sigma_2} \right)^2}.
\]

Therefore, the overall distribution can be computed from independent distributions of the variables as;

\[
P(x; \mu, \Sigma) = \prod_{i=1}^{n} p(x_i; \mu_i, \sigma_i).
\]
Then, for a set of \( n \) independent variables the multi-variate Gaussian distribution is defined as:

\[
p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}} \left( \prod_{i=1}^{n} \sigma_i \right)^{1/2} \exp\left[ -\frac{1}{2} \left( \sum_{i=1}^{n} \Delta \right) \right].
\]  
(2.16)

Where, \( \Delta = [(x_i - \mu_i)/\sigma_i]^2 \), which is square of the Mahalanobis distance between the concerned distributions.

### 2.8 Deep learning

It is a subset of machine learning (ML), like shown in Fig. 2.3 research based on extracting features and data representations, unlike task-dependant algorithms. The objective is of moving ML closer to the ever expected goal of artificial intelligence (AI). It includes forward neural netowrks, convolutional neural networks, recurrent neural networks, etc.

![Venn diagram showing how deep learning (DL) is a kind of representation learning, which is in turn a part of machine learning (ML).](image)

**Figure 2.3**: A Venn diagram showing how deep learning (DL) is a kind of representation learning, which is in turn a part of machine learning (ML).

### 2.9 Convolutional neural network

It is a biologically-inspired variant of DL as depicted in Fig. 2.4. It subsumes one or more Convolutional layers (Conv), non-linear activation layers, like Rectified Linear Unit (ReLU) and Batch Normalization (BN), and then followed by one or more Fully Connected (FC) or densely connected layers, like in a multi-layer DL architectures.

In Fig. 2.4, \( T(\cdot) \) denotes a transformation operation or transfer functions, including Conv, Pooling (max, average), Linear rectification, BN, Softmax probability.
estimation, Cost calculation, etc. An actual CNN implemented for handwritten digit recognition is shown in Fig. 2.5.

In Fig. 2.5, there are two parts: (i). Feature extractors - the input data goes through several convolutional layers to extract progressively more complex and abstract features, (ii). Classifiers the densely connected layers, and objective functions. The Feature extractors that map the raw data to a transformed feature space, then a score function that maps the extracted features to class scores, and finally a loss function that evaluates the relationship between the prediction and the ground truth numerically. Thus, the CNN algorithm casts this as an optimization problem in which it will minimize the loss function through backpropagation aka backpropagation (BP) with respect to the parameters of the score function.

The feature extractor in the majority of the CNN architectures used in these days for standard applications, like object recognition/ classification is implemented
based on the same principles to ensure some degree of shift, scale, and distortion-
invariant by using local receptive fields, shared weights, and spatial sub-sampling. It
is achieved by having series of Conv layers with interspersed pooling (Pool) and ReLU
layers followed by two or three FC top layers while the last layer being a Softmax
classifier \cite{64, 72, 132}. Besides that, the network architectures are as a rule-of-
thumb parametrized to be large and regularized during training using dropout \cite{134}.
It is has been empirically proven that the representation depth is beneficial for the
classification accuracy, and that state-of-the-art performance on various data-sets
can be achieved through ConvNet architecture \cite{76, 133} with substantially increased
depth.

2.9.1 Convolution layers

The convolution or conv layers are the heart of CNN, which contain neurons that take
their synaptic inputs from a local region of the input volume called local receptive
field (i.e., the filter, interchangeably the kernel), whereby neurons detect local visual
features: oriented edges, end-points, or corners. The weights of the convolution
neurons within a feature map are shared, so the position of the local feature becomes
less critical, thereby yielding shift invariant. The feature map of a conv layer w.r.t.
filter $\omega$, its associated bias $b$ and input image/ patch (aka set of input activations) $x$
is computed as:

$$ C(m, n) = b + \sum_{k=0}^{K-1} \sum_{l=0}^{K-1} \omega(k, l) \ast x(m + k, n + l), \quad (2.17) $$

where $\ast$ represents the convolutional operation. A sample feature map generation
through conv layer is shown in Fig. 2.6

2.9.2 Densely connected layers

Densely connected layers or commonly known as fully connected (FC) layers take
flattened, i.e., 1D output activations of the previous layer (be it a convolutional,
pooling, or fully connected) and connects it to every single neuron it activates. Thus,
the FC layers are not spatially located (just one-dimensional vectors), which implies
that there can be no convolutional layers once a FC layer is employed in the chain
of CNN. It can be expressed as $FC = f(matmul(input\_flat, W_{FC}) + b_{FC})$ which
is technically a matrix multiplication and vector addition where, $f(\cdot)$ performs an
activation operation, $matmul$ performs matrix multiplication on the flattened input
batch of vectors and weight matrix \((W_\text{FC})\), and \(b_\text{FC}\) is the bias vector associated with each output activations. FC layers are to support the final classification process as shown in fig. 2.7 where the Softmax function, \(\sigma(\cdot)\) is given by (2.20).

**2.10 Activation function**

The goal of using activation function \(f\) is to bring non-linearity into networks, thus to find the optimal weights. The adaptation of activation functions in neural networks
(NN) can be referenced to the work of McCulloch and Pitts in late 1943 [90], where the activation function rectifies the input to either 1 or −1 if its value is positive or negative, respectively. However, in CNNs, it takes a real-valued input (generally, the value strengthen after the convolved feature summation as shown in fig. 2.6) and performs a mathematical operation to convert it to range of [0, 1]. Historically, the sigmoid function is a default choice; however, it has recently fallen out of favor and it is rarely ever used [123] due to its technical drawbacks. Besides the sigmoid function, there are several other activation functions in practice such as Tanh non-linearity (tanh), Rectified Linear Unit (ReLU), Leaky ReLU, Parametric ReLU, and Randomized ReLU.

2.10.1 ReLU

The rectilinear units or ReLUs are nonlinear activation functions generally used after convolutional operations. The ReLU can be formally defined as in (2.18) when taken a case where there are \( K \) number of anchor vectors, denoted by \( \mathbf{w}_k \in \mathbb{R}^N, k = 1, 2, \ldots, K \). For a given input \( \mathbf{x} \), the correlations with \( \mathbf{a}_k \) and \( k = 1, 2, \ldots, K \), defines a nonlinear rectification to an output \( \mathbf{y} = (y_1, \ldots, y_K)^T \), where

\[
y_k(\mathbf{x}, \mathbf{a}_k) = \max(0, \mathbf{a}_k^T \mathbf{x}) \equiv \text{ReLU}(\mathbf{a}_k^T \mathbf{x}),
\]

(2.18)
i.e., it clips negative values to zero while keeping positive ones intact. The benefit of ReLU is sparsity, overcoming vanishing gradient issue, and efficient computation than other activation units.

2.10.2 Sigmoid

The Sigmoid activation function, on the other hand, has output in the range \([0, 1]\) for an input \( \mathbf{x} \) and it is defined by

\[
f(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{x})}.
\]

(2.19)

Therefore, it is very appropriate for binary classification tasks, like in this work and linear regression problems. A thorough exposition of the purpose of activation functions in NN with graphical examples can be found in [73].
2.10.3 Softmax

The Softmax activation is defined as,

\[
\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, \ldots, K
\]  

(2.20)

where, \( z \) is a K-dimensional vector of arbitrary real values and the output \( \sigma(z) \) K-dimensional vector of real values in the range \((0, 1)\) that add up to 1 as the normalization happens via the sum of exponents term dividing actual exponentiation term forming a valid probability distribution. Thus, if the final layer at the end of a CNN is a Softmax classifier, then it yields the actual probability scores for each class label. Some cases, the Softmax layer is modified to calculate the cross-entropy loss through taking negative logarithm on (2.20):

\[
L_j = -\log \left( \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \right)
\]  

(2.21)

where the logarithm is actually base \( e \) (natural logarithm).

2.10.4 Batch normalization

The batch normalization operation has multifaceted benefits:

i. Reducing internal co-variate shift - During training, there is a change in the distribution of activation maps as network parameters are being tuned. Such condition challenges the learning, but the BN alleviate pressure by maintaining the mean and standard deviation of the activation close to 0 and 1, respectively.

ii. Effect of regularization - Since the batch of examples given in the training are normalized, it increases the generalization of the model. It is also claimed that BN allows to reduce the strength of dropout.

iii. Counterbalancing vanishing or exploding gradients - When the BN is located prior to non-linearity, it avoids an undesirable situation, where the training saturates areas of non-linearities, solving the issues of vanishing exploding gradients.

Mathematically it can be defined as follows. Let the output of a layer \( \mathbf{X} \in \mathbb{R}^{N,D} \), where \( N \) is the number of samples available in the mini-batch and \( D \) is the number of hidden neurons, then normalized matrix \( \hat{\mathbf{X}} \) is given as in (2.22) [63].

\[
\hat{\mathbf{X}} = \frac{\mathbf{X} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}},
\]  

(2.22)
where $\mu_B, \sigma^2_B$, and $\epsilon$ refer to the mean and variance of the mini-batch, and a small value of 0.001 to prevent division by zero, respectively. Then the layer maintains its representational strength by testing the identity transform as:

$$y = \gamma \hat{X} + \beta,$$  \hspace{1cm} (2.23)

where, $\beta$ and $\gamma$ are trainable parameters that are initialized with $\beta = 0$ and $\gamma = 1$, in this work. Note that, when $\beta = \mu_B$ and $\gamma = \sqrt{\sigma^2_B + \epsilon}$ Eqn. (2.23) returns the previous layer’s activation map.

## 2.11 Long-short term memory

Long short-term memory or the LSTM NNs are the advanced version of the general recurrent neural networks (RNN). The LSTMs overcome the vanish and exploding gradient issues of general RNNs by using the Constant Error Carousel (CEC) cells that use an identity activation function. The LSTM-based NNs interconnect sequential memories both in the long and in the short term, which makes them apt architecture for time-dependent tasks, viz. handwriting recognition [111], speech/language identification [45], robot control/localization [151], driver distraction detection [163], and action recognition.

In every iteration, an LSTM module updates the state parameters and outputs by controlling a hidden vector $h$ given by (2.29) and a memory register vector $m$ given by (2.28). The LSTM networks inherit the ability to learn short-term temporal dependencies in sequences as Graves et al. [68] drive the associated parameter computation at an iteration step $t$, like:

$$g^u = \sigma(W^u h_{t-1} + I^u X_t), \hspace{1cm} (2.24)$$

$$g^f = \sigma(W^f h_{t-1} + I^f X_t), \hspace{1cm} (2.25)$$

$$g^o = \sigma(W^o h_{t-1} + I^o X_t), \hspace{1cm} (2.26)$$

$$g^c = \tanh(W^c h_{t-1} + I^c X_t), \hspace{1cm} (2.27)$$

$$m_t = g^f \odot m_{t-1} + g^u \odot g^c, \text{ and}$$

$$h_t = \tanh(g^o \odot m_t). \hspace{1cm} (2.29)$$

Where, $\sigma$ is the logistic Sigmoid function given by (2.19), $\odot$ denotes element-wise vector multiplication. Hence, the $W^u$, $W^f$, $W^o$, and $W^c$ are recurrent weight matrices while the $I^u$, $I^f$, $I^o$, and $I^c$ are the associated projection matrices. Thence,
the gates $g^u$, $g^f$, $g^o$, and $g^c$ refer to the input gate, the forget gate, the output gate, and the cell gate respectively.

2.12 Transfer learning

The essence of CNN architecture is how precisely modeling raw data samples through multiple processing layers with non-linearities; when a deep CNN has many parameters, for instance, the AlexNet [72] contains more than 60 million parameters, making it powerful function approximator. Such, deep neural networks have demonstrated robustness in visual-based classification, recognition, and detection. However, training these models requires, large amount of labeled data (for instance, 14,197,122 images, 21841 classes in ImageNet 2016, Sep.), because directly learning many such parameters from small data-sets, which contain only a few thousand training images is hard [103]. At the same time, there is a need for a careful initialization for optimized and efficient training. Hence, to train CNNs using large annotated images require specialized hardware, for instance, the GPUs (good GPU would cost few thousand dollars), GPU programming skills, energy and time.

![Block diagram describing transfer learning technique.](image)

Transfer learning (TL) techniques circumvent the shortcomings above. Its primary goal is to transfer learned knowledge between the source and target domains so that it can overcome the deficit of training samples for some categories in the new tasks. Transfer learning has been coexisted in Machine Learning (ML), Artificial Intelligence (AI), and Neural Network (NN) with various terminologies, such as knowledge transfer, meta-learning, inductive transfer, parameter transfer, life-long learning, or context-sensitive learning [29]. It is a strategy, which has been highly appreciated in recent years by CV communities to design NNs that can leverage learned properties from a source domain into a target domain.
Thus, when the task in target domain is trained on different statistics than the source domain, but using the learned parameters of the source task, it is conceived as transfer learning (refer Fig. 2.8). Here, the outcome of using TL is tri-fold: less time is required for learning the target domain task, less information is needed of experts (usually human), and more cases can be dealt efficiently [29], [2]. These benefits drive researchers to exploit TL techniques in many applications, including image/video/speech/action classification and recognition. Hence, it has progressed in modern-day DCNN, where it is employed to reduce training time, specialized hardware requirements, and effort to initialize parameters (weights and biases) of the network [116], [175]. For instance, in [116], Razavian et al. show the performance gain of employing transfer learning in image classification using a single pre-trained DCNN, which outmatches most of the state-of-the-art hand-crafted features. Besides that, initializing a CNN with transferred weights can boost the generalization of the network on the new domain that lingers even after fine-tuning [175]. Generally, in transfer learning a base network is trained on the source domain then its weights are copied to the new network in the target domain, and a back-propagation operation is carried through the entire network (end to end) to fine tune the new network toward the new task.
Chapter 3

Literate review

Recent state-of-the-art technology in transistor fabrication has driven the clock rate of Central Processing Unit (CPU) and Graphic Processing Unit (GPU) to reach super high frequency; thus, the computational speed of processors has enormously increased. Hence, it has paved the way for bigger available physical memory or Random Access Memory (RAM) and storage disk space in modern computers. Consequently, it has enabled application of computer vision technologies in several fields, such as Multimedia-based surveillance, Industrial Automation [82], Automotive, and Intelligent transportation systems. In such applications, the FGL module plays an integral role in guiding vision-based solutions to ignore unwanted region of a scene monitored by exploiting visual properties and shifting the attention to moving objects, such as running vehicles and walking people. It is crucial when it comes to priority specific data compression on Terabyte (TB) size multimedia (mainly video) surveillance footage.

For example, consider Fig. 3.1, where (a) is an actual scene\footnote{A scene in Railway data sequence downloaded from http://www.cs.cmu.edu/~yaser/new_backgroundsubtraction.htm} and (b) is the useful FG region, which is about 25 to 30\% of the actual scene. This piece of in-

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure3.1.png}
\caption{A scene and its useful region of interest.}
\end{figure}
formation is valuable for a video compression algorithm so that it can use selective techniques to compress the FG and BG regions differently, whereby details of the useful FG region remain intact while the BG region experiences higher compression rate. Besides, once the FG objects/regions are well localized, the fruitful low-level visual cues of the current scene can be extracted, and it can be, then, employed for high-level analysis. The followings are few applications as the product of such high-level analysis: autonomous/intelligent driving [6, 170], object indexing and retrieval [20, 74, 143], traffic monitoring (detecting, counting or tracking of objects) [167], human activity recognition (run, walk, jump, squat, etc.) [53], human-machine interaction (in general, human-machine interface- HMI) [46], moving object tracking (many live sports telecasting channels have adopted this) [184], scene classification, digital forensics [118, 157], image quality assessment [4, 28], and so forth.

The FGL algorithms and models can be categorized as the Venn diagram in Figure 3.2. There are five categories:

1. Sample-based [11, 62, 137, 144, 146],
2. Probability-based [3, 34, 65, 139, 149, 158],
3. Subspace-based [10, 52, 102],
4. Codebook-based [138, 164, 176], and
5. Neural network (NN)-based [8, 41, 122, 177, 181].
Then, all these five categories can be grouped into pixel-based, region-based, and the combination of the two strategies. The sample-based algorithms create a BG from the past set of $N$ frames, i.e., for each pixel location there are $N$ samples stored. If there are $k$ number of pixels in the BG that have a distance smaller than a threshold $\tau$ to the incoming pixel, then the pixel is classified as FG. The probabilistic models work on the principle of stochastic process, like Gaussian mixture models (GMM) [3, 94] and Conditional Random Field (CRF)-based algorithms [187]. The subspace-based approaches perform a transformation of data to a subspace, such as Eigenspace or Principal Component Analysis (PCA)-based subspace. Then, they form a BG model using the subspace and estimate the FG. The Code-book generates a dictionary that consists of color, intensity, temporal features, or similar representations. Same properties of a new pixel are compared with the dictionary values to determine it’s status. The NN-based models are kind of generating a classifier through training to handle the segmentation task. The trained weights of a NN serve as BG model and can be updated to reflect the changes occurred in the scene. Here, a learning system, which formulates FGL as a structured input-output matching problem. Such models have gained their reputation after breakthrough performances in the ImageNet-Large-Scale Visual Recognition Challenge (ILSVRC). The NN-based techniques have been exploited for image semantics/ labeling [130, 178], medical image partitioning [109, 119], and recently for video FG segmentation [169] as well. The main challenges in CNN-based FG detection is that dealing with time-dependent motion and the dithering effect at bordering pixels of FG objects. We address these issues in Chapters 7, 8, and 9, by excogitating an encoder-decoder (EnDec) CNN-LSTM that utilizes ResNet 56-like residual connections for lost feature recovery and LSTM units to handle spatiotemporal motion of FG objects. To facilitate the training process, we take advantage of intra-domain transfer learning.

### 3.1 Pixel-based background modeling

Pixel-based methods have received great acclamation since Stauffer and Grimson [139] introduced multi-model Gaussian or the GMM for real-time tracking for video surveillance. Due to its applicability, the GMM is utilized in many applications such as object detection, recognition, and tracking. The GMM is a statistical model, where each BG pixel is represented as a mixture of $k$ number of Gaussian models; then based on persistence and variance of each distribution, $m$ distributions
are chosen to represent the BG. Following the work of Stauffer and Grimson, several researchers in this domain have proposed various techniques in order to improve the statistical model. Among them, some researchers focus on techniques to update the model parameters, for instance, Zivkovic [186] implements an adaptive algorithm to choose a required number of Gaussian components per pixel and update the model parameters. Similarly, Dawei et al. [32] and Zhou et al. [182] take advantage of expectation-maximization (EM), k-means clustering, Kernel density estimation (KDE) and Markov random field (MRF) for parameter updates and to refine the estimated FG respectively. Since the MRF-based FG refinement is an iterative process, it generally consumes higher processing time. On the other hand, Yang et al. [160] utilizes a spatial relationship between neighboring pixels through a conditional random field (CRF), which influences the learning process of the GMM-based classification for in-door video segmentation. This model faces a significant toll in computational speed and implementation complexity. To address this issue, Mukherjee et al. [91] propose a wavelet-based decomposition and a variable number of clustering technique (referred as WavGMM hereafter). This approach decomposes the input scene into multi-resolution sub-bands so that useful features in different scales can be incorporated in the learning step as temporal data with the consideration that the sub-bands are relatively independent. At the same time, this method achieves better computational speed because the sub-bands are smaller compared to the original input image. Lee [78] proposes a technique to balance GMM convergence speed and stability through computing appropriate learning rate when parameters of a Gaussian are updated and integrates a Bayesian framework to isolate the most-likely BG Gaussian distributions and generate an intuitive representation of the believed-to-be-BG. This approach, however, falls into the cost of additional computation and updates.

Besides the GMM based approaches, Bayesian-based BG modeling frameworks also have frequently been adapted for BG-FG separation. In [113], Zhu et al. take a multilayered Gaussian distribution to initialize each pixel for dissimilar BG contents and a recursive Bayesian learning based on texture correlation to refine the detected FG. Likewise, Li et al. [79] use Bayesian decision rule that incorporates spectral, spatial, and temporal features to define BG. Vosters et al. [152] compute an Eigenstate model from a training set of BG images that are recorded under various lighting conditions from a fixed viewpoint and reconstruct a generalized BG.

Some researchers use local features, like texture and color to model a BG through that detecting the FG. For instance, Hofmann et al. [60] come up a Pixel-based
adaptive segmenter (PBAS) that follows the principle of a non-parametric paradigm. Thus, the BG is generated from a dictionary of recently observed pixel values and then the FG region is localized depends on a predetermined threshold. Charles et al. [137] coined a system as SuBSENSE, an abbreviation of Self-Balanced SESitivity SEgmeneter that adapts and integrates Local Binary Similarity Pattern (LBSP) as additional features to pixel intensities in a non-parametric BG model that is then automatically tuned using pixel-level feedback loops. Authors in [66] approach the background modeling as evidence collection of each pixel in a scene with a weight-sample-based method. They also use a minimum-weight and reward-and-penalty weighting strategy to account rapidly changing scenarios in a way that most inefficient sample is replaced instead of the oldest sample or a random sample. Then a pixel is classified as BG if the sum of the weights of the active samples is larger than a manually set specific threshold; otherwise, it is classified as FG. Besides the simplicity of the method, it records a poor FG detection accuracy.

Hence, there are more pixel-based algorithms proposed; notably with MRF [126], non-parametric models, such as ViBe (visual background extraction) [174] that rely on the sample consensus, non-parametric Bayesian models [50], algorithms based on artificial neural network (ANN), like [40, 87, 106, 114], and BG modeling with perception-inspired confidence interval [53]. A complete study can be found in [16] and [168]. Although the pixel-based approaches possess good characteristics, they perform poorly when input frames contain non-static backgrounds, illumination changes, or noise [143].

### 3.2 Region-based background modeling

The idea of region-based BG modeling is, incorporating the correlation between a pixel and its neighboring pixels or regions in bigger level to improve BG-FG classification accuracy [19, 26, 67, 117, 131]. Chiranjeevi and Sengupta [26] perform the classification via a multi-channel correlogram with kernel fuzzy c-mean membership function. They use inter-channel and multi-channel correlograms called Multi-channel kernel fuzzy Correlogram (MKFC) to improve classification accuracy. In contrast to GMM based BGS algorithms, which use a fixed threshold to decide if a pixel belongs to FG, they use Generalized Tverksy index (GTI) based similarity measure between the current frame and the BG estimated by MKFC to decide if a region belongs to FG. Since this method does not require clean BG sequences for initial training, there will be considerable miss-classification at first few frames of a sequence. It is a common issue
in GMM based algorithms as well. On the other hand, Varadarajan et al. [149] propose a method that takes the spatial relationship between pixels into account through a region-based GMM in contrast to the traditional GMM that works on individual pixels rather than regions [3]. In [19] Cao et al. use the spatial and temporal information to calculate total variation based on the Robust principal component analysis (RPCA) with the assumption that dynamic BG is sparser than the moving FG that has smooth boundary and trajectory. Likewise, authors of [50] and [93] also come up with models to exploit pixel distributions over the time. In [50], Haines and Xiang employ Dirichlet Process Mixture Models (DPMM) for primary FG detection and an MRF constructed using a 4-way neighborhood of a pixel to refine the detection. While, Narayana et al. [93] have a field of distributions with one distribution at each pixel location by separating the various aspects of a BG modeling system, the likelihood of background and foreground, and a prior, into disjoint entities. Then, they compute the posterior probability of BG and FG, conditioned on the observed pixel values using Bayes’ rule.

In the direction of utilizing spatiotemporal cues, the MRF is widely adopted. For instance, Jiuyue et al. [67] propose a model with a Bayesian formulation for temporal coherence, a Gaussian function for spatial FG representation, and KDE for combined spatial-temporal BG representation. Shih-Shinh et al. [131] carry out motion and color based regional segmentation, which exploits both intensity and edge information in an initial process. Then, they proceed with a statistical framework, the MRF labeling and perform an optimization process for final classification. Finally, they merge same label classes that have similar color regions to extract the BG. This approach produces an unsatisfactory result when an object moves at a high pace since it predominately depends on motion estimation. Similarly, Patras et al. [108] also take advantage of MRF for labeling regions, which are segmented initially by watershed segmentation. Although that algorithm takes static spatial and temporal interaction of region process into account, it over-segments regions in times due to region-merging. Thus, it raises complication in region classification. Crivelli et al. [31] employ a method so-called mixed-state statistical framework based on mixed-state MRF and mixed-state CRF to jointly model motion detection (symbolic) and BG image reconstruction (numeric).

On the other hand, Zhao et al. [180] use spatiotemporal patches, called video-bricks, to characterize both the appearance and motion information through observation of all the BG blocks at a given location under various lighting conditions lie in a low dimensional PCA-based subspace computed, while blocks with moving FG are widely distributed outside. This video-brick based technique fails in the surface
with strong specular reflection as it breaks the Lambertian assumption\(^2\). In [117], Ren et al. propose a region-based adaptive mean-shift algorithm with GMM clustering for saliency detection. In [23], Chen et al. carry out a region-based object recognition by using simplified pulse-coupled neural network based on both temporal pulsing detail and spatial distributions of the image.

Due to the nature of providing stable detail of non-static backgrounds, descriptor-based algorithms are also highly appreciated [49,136,153] in BG modeling. St-Charles et al. [135,136] integrate a similarity feature based on Local binary patterns (LBP) with colors in pixel-level to negate the sensitivity of LBP features to change in dynamic BG regions and highly contrasted/noisy areas. This method costs higher computational time unless the required color information has already been kept locally to compute inter-frame LBSP descriptors. Similarly, Nonaka et al. [101] perform an adaptive Radial reach correlation (RRC) to model the BG changes. The RRC is quite similar in nature to the LBP. Heikkila and Pietikainen [57] directly use the texture-based descriptor, LBP to model the BG. Wan-Chen et al. [153] and Guo-Hao [49] model the BG as a sample of a binary descriptor to identify foregrounds under illumination variations. Oliver et al. [102] use extracted image features rather than each pixel to classify the BG based on Eigen-background using characteristic roots decomposition. In [120], Russel and Gong apply a block matching algorithm on the image regions of incoming frames based on a fixed-size database which represents typical BG, where scores of the heuristic matching determine FG regions. In [48], the authors exploit Local Binary Patterns (LBP) with local Singular Value Decomposition (SVD) operator to extract invariant representation that is similar to the LBSP in [137]. Then, they use SAmple CONsensus (SACON) approach for building the BG model based on statistics of the pixel processes (about 300 frames). Then, they employ the Hamming distance, like applied in [177] with a threshold to classify each pixel as FG or BG. These models require static and clean background samples to build up the dictionary; thus, they lack application for real-world problems. Similarly, Allebosch et al. [5] also employ local features, such as, Local Ternary Pattern (LTP) based edge descriptors and RGB color cues to classify individual pixels. They form two backgrounds based on the aforesaid edge descriptors and color cues and create two FG masks. Then, using a pixel-wise logical AND operation they refine the detected FG region.

\(^2\) The radiant intensity observed from an ideal diffusely reflecting surface is directly proportional to the cosine of the angle between the direction of the surface normal and the incident light.
In the literature \cite{177}, Zhang et al. develop a NN that has a Stacked Denoising Auto-Encoder (SDAE) learning module and a binary scene modeling based on density analysis. Whereby, the SDAE encodes the intrinsic structural information of a scene. The encoded features of image patches are then hashed in Hamming space, and then based on the hash method a binary scene is modeled through density analysis, which captures the spatiotemporal distribution information. Similarly, Zhao et al. \cite{181} also take advantage of NNs with a stacked multilayer Self-Organizing Map (SOM) to model the BG. In which, the model is initially trained using some BG samples, and then, the trained model is used for FG detection for a new sample. At the same time, the BG model is updated online for BG maintenance. Gemignani and Rozza \cite{41} extend the basic SOM model of Zhao et al. with a self-balancing multi-layered SOM that tracks a long time pixel dynamics for better FG detection.

Besides the advantages of region-based BG modeling, it still suffers from false positives when FG objects are not rough-shape ones.

### 3.3 Hybrid background modeling

Some researchers try to capitalize on the best of pixel- and region-based approaches under an integrated framework called hybrid-based BG modeling \cite{35,36,65,99,158}. Ning et al. \cite{99} introduce an approach for moving object tracking for PTZ-based video sequences with a weighted Gaussian-like KDE to model the BG, where the features of the model estimation are local patterns like LBP rather than pure pixel intensities and the local patterns are extracted from a set of hierarchical ensembles. Javed et al. \cite{65} decompose the input scene into Gaussian and Laplacian images. Then, apply Robust Online PCA (OR-PCA) to both the images for BG modeling. It is a clever approach since the Gaussian image is robust against the noise of small pixel variations and Laplacian image preserves edge features. In the end, they utilize an MRF similar to \cite{67,108,131} to exploit structural information and similarities to improve the segmented FG. In contrast to the above methods, the BG modeling problem has been addressed through a layered operation flow. For example, Evengelio et al. \cite{36} propose two-layered pixel- and region-based feed-back system. Where, the classification results from region-based is fed-back to the pixel-based to achieve a better BG model. While, Wang et al. \cite{158} come up with a split Gaussian model that exploits the power of fusing a motion computation method based on spatiotemporal tensor formulation, and a multi-cue appearance comparison to cope with conditions such as shadows, illumination changes, dynamic background, stopped and removed objects.
Similar to [117], Escudero-Vinolo and Bescos [35] also use Mean-shift to cluster similar regions and perform an inter-cluster fusion based on color vector angle measure. Then, they extract confirmed BG regions, which are overlapping with a basic-pixel level segmentation. In general, at pixel-level, some rules are applied to predict the BG values, then integrated with region-level information, for instance, optical-flow motion. Some cases, frame-level, i.e., global information are also considered in the integration of pixel and region level BG modelings to handle sudden changes in the input video streams.

Besides the above uni-modal approaches, there is a shift in attention towards multi-modal FGL system too, like Bianco et al. [12] (IUTIS-5). They explore a way of harnessing multiple state-of-the-art motion detection algorithms for achieving an enhanced FG mask. They obtain a solution tree by Genetic Programming (GP). Sajid et al. [121] also focus on multi-modal framework that creates multiple BG models of the scene and use them as BG model bank. To identify the FG pixels from changing BG pixels, they apply Mega-Pixel (MP)-based spatial de-noising to pixel-level probability estimation on variant color spaces and get multiple FG regions. Then they are fused to localize the final FG.

Although researchers try to propose new methods for FGL task, it is equally important to improve the existing methods. At the same time, driving an ultimate model to address a diversity of scenarios remains a challenging task. Thus, to achieve higher accuracy and alleviate implementation complexity in Chapters 5 and 6 we propose the following improvements to Multivariate GMM (MVGMM) based BG-FG separation: (i). The distance measure between a pixel and Gaussian distributions is improved by utilizing properties of Bhattacharyya measure, (ii). A novel method is introduced to enhance FG feature and to find an optimal threshold to extract accurate FG region. The results of this threshold method are compared with the performance of highly credited Kittler-Illingworth algorithm, and (iii) A new framework is presented to validate the detected FG region by utilizing estimated posterior probability of MVGMD.

3.4 CNN-based semantic segmentation to FGL

The deep convolutional neural networks, have become a cutting-edge technique in CV systems. For instance, the CNNs have shown state-of-the-art performance in object segmentation, detection, and localization tasks over traditional methods, like GMM [3,143], Graph-cut, Non-parametric models [144], Visual Background extractor (ViBe)
The advanced solutions for FGL are built upon a CNN architecture, the Fully Convolutional Network, a.k.a. FCN. It is a pioneer model that reinterprets the standard visual classification network as layers of fully 2D convolutional computations without spatial pooling and fully connected layers. It has been exploited for many applications, including, semantic segmentation. This model introduces feature-level augmentations through skip connections that combine deep, coarse, semantic detail and shallow, fine, appearance cues from chosen mid-layers. In contrast, our models in Chapter 7, 8, 9 perform the coarse-level feature fusion in a structured manner that differs from the residual connections of ResNet, as an example shown in Fig. 3.3. The ResNet was built upon the philosophy of increasing the network depth via residual connections instead of widening for a better object representation. This architecture negates the vanishing gradient of deep networks by carrying important information from earlier to later layers. Although such shortcuts seem, like an addition to the conventional CNN connections, it alleviates training and reduces the number of parameters. An illustration for the ResNet connection is depicted in Fig. 3.3 (a), where \( X \) is input feature, \( H(X) \) is a desired transformation, and \( F(X) \) is a residual mapping. In [56], the feature fusion operation \( H(X) = F(X) + X \) is performed by a shortcut connection and element-wise addition. In contrast, our model stacks the features depth-wise as \( H(X) = F(X) \otimes X \), like in Fig. 3.3 (b), where \( \otimes \) denotes coarse-level feature concatenation. This favors to have less number of filters in conv layers resulting less computation.

Figure 3.3: CNN feature flows: (a) ResNet flow, and (b) the residual feature mapping of our models.

Olaf et al. [119] restructures the FCN as EnDec CNN shown in Fig. 3.4 referred U-net for bio-medical cell segmentation. Wherein, the activation maps after each convolution (conv) in the encoding stage are concatenated with the spatially matching
activation maps in the decoding stage. It allows the network to exploit the original contextual information to supplement the features after upsampling at the higher layers. In other words, it is a remedy for the lost spatial resolution due to pooling operations or consecutive convolutional kernel striding. Milletari et al. [37] extend this model to be trained end-to-end on MRI volumes depicting prostate and learns to predict segmentation for the whole volume at once. They name the network V-net as it does volumetric medical image segmentation. The significant difference of V-net from U-net is the volumetric convolutions as the input is a slice-wise volume (3D patch).

Figure 3.4: U-net architecture. Blue boxes correspond to a multi-channel feature map. The number of channels is denoted on top of the boxes. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps and the arrows refer to the different type of operations [119].

Followed by the successful campaign of the FCN and the U-net, there have been several DL-based models developed to perform FGL. In the literature [80] Lim et al. implement a CNN that is similar to the U-net to roughly detect regions of moving object. It is then complemented by a contour information produced by a BG model. The BG model is updated frame-by-frame for the unchanged pixels, i.e., for the non-FG regions as

$$B_{\text{temp}} = (1 - C_t) \otimes F_t + C_t \otimes B_{t-1},$$

(3.1)

where $\otimes$ refers to element-wise product, 1 is the binary mask that has all values are
$C_t$ and $F_t$ denote the predicted binary FG mask and the current frame at time $t$, and $B_{t-1}$ is the BG model belongs to the previous frame at $t-1$ time stamp. The BG information and two consecutive frames are stacked depth-wise and fed to the network. In our case, also the proposed models receive the same set of information as input; however, the BG model is precomputed to avoid run-time computational complexity. Similarly, Yang et al. [170] exploit the FCN for a Deep Background modelling (DBFCN). In that, they modify the original FCN structure by including blocks of three Atrous convolutional branches to capture spatial information from different neighbourhoods and BN blocks for regulating the features. They also include a Conditional Random Field (CRF) to remove any ambiguous information due to large receptive in the last layer.

Meanwhile, Babaee et al. [8] employ a conventional CNN, similar to the network for handwritten digit classifier, LeNet [77] with three conv layers and two FC layers. The main argument of the approach is that background subtraction can be performed without temporal information, given a sufficiently good background image. Thus, the network is trained with randomly selected video frames with segmentation ground-truths from all different categories and temporally-median filtered SuBSENSE [137] generated background images patch-wise, like in [177]. At the end, authors carry out a post-processing to smooth the output through a spatial-median filtering. This model’s performance is limited by the performance of SuBSENSE in BG generation and lack of computational efficiency due to patch-based prediction. Similar model is used in [17] as illustrated by Fig. 3.5. The network is trained with frame patches size of $T = 27$, where the BG model is created from few first samples (150 frames) of each sequence and the network is trained dataset specifically by taking the first half of images that have ground truths is considered as training data while the second half is used as a test set. The pixel classification process is determined by feeding the trained model with the two patches extracted from the input and median images centred on that pixel.

\[3\] It allows user to explicitly control the resolution at which feature responses computed within DCNNs. It also allows us to effectively enlarge the FOV of filters to incorporate larger context without increasing the number of parameters or the amount of computation.
Figure 3.5: The network is trained with two small patches extracted from the input and background images in gray-scale. The network is inspired by LeNet-5 network [17].
Chapter 4

Video foreground detection in non-static background using multi-dimensional color space

4.1 Summary

Foreground localization through background subtraction is a vital task in video sequence-based applications. It is a very challenging process when the BG is non-static. Although there have been many algorithms proposed in the literature, most of them are complex in terms of either mathematical modeling or computational requirements. This chapter proposes two computationally simple algorithms for video FG detection using multi-dimensional color space when the BG is non-static. The algorithms utilize pixel level temporal intensity for FG and BG classification. The algorithms are tested on two sets of outdoor video sequences where the backgrounds are non-static. The experimental results show that the algorithms adequately perform well on the given environments.

4.2 Introduction

Over the past decade utilizing video surveillance systems has largely increased since it is a way of eco-friendly approach in comparison to using multiple electromagnetic (EM) wave-based sensors and sensor networks for surveillance purposes. As a consequence, researchers in computer vision related fields proposed many robust BG suppression/subtraction models for real-time FG detection [22, 51, 53, 70, 85, 115, 145, 171, 173]. For an automated video surveillance system, the BG suppression plays an inevitable role. BG subtraction fundamentally helps the system to ignore the
unwanted area of a scene being monitored and bring the attention to moving objects, for instance, a moving car or a walking man. Thus, the important foreground information can be extracted for further analysis, such as traffic monitoring (vehicle detection, counting, and tracking), human activity recognition (run, walk, jump, squat), human-machine interaction or interface (HMI), moving object tracking (many live sports telecasting channels adopts this) and so forth.

A reliable BG subtraction algorithm should be robust and able to handle sudden or gradual illumination changes, high frequency moving objects, repetitive motion in the background (such as tree leaves, flags, or sea waves) and long-term scene changes (a car is parked for a month, for instance). Thus, there were many algorithms proposed fundamentally based on statistical analysis like Gaussian mixture models (GMM) over the past two decades since the pioneer work of Stauffer and Grimson [139]. For instance, Effective GMM [78], GMM-based Conditional Random Filed [159, 160] Variational clustered GMM [24], and Wavelet transformation-based GMM [91]. However, such high complexity algorithms are not necessary for certain surveillance purposes, such as monitoring an automatic teller machine (ATM) in a shopping complex or bank. Because, in such cases the surveillance camera is fixed at a place and the background environment is known prior to monitoring. In such conditions, it is recommended to employ simplistic models to detect the foregrounds, i.e., the moving objects in the given environment being monitored. To this end, this chapter presents two computationally efficient algorithms: probabilistic-based model with non-supervised threshold and 3D-color space model using distance vector for BGS. The contribution of this chapter is twofold; proposing simple techniques for non-static BG suppression for a constrained environment and reporting their experimental results and limitations. The remainder of this chapter is organized as follow: Section 4.3 describes the algorithms in detail, Section 4.4 presents the experimental results, and Section 4.5 concludes the chapter with discussions and recommendations for future work.

4.3 The algorithm

4.3.1 Probabilistic-based background suppression with non-supervised threshold

For a known environment, its BG can be modelled based on stochastic theories using collected samples prior to actual monitoring. Considering that, every pixel in
the scene has its own Probability Mass Function (PMF), \( PMF_{N,c}(I_{u,v}) \) with intensity ranging from 0 to \( 2^n \), where \( N \) is number of samples collected to estimate the BG, the channel parameter \( c \in \mathbb{R}^D \). In the case of RGB colour space, \( c \in \mathbb{R}^3 \equiv \{Red, Green, Blue\} \), \( I_{u,v} \) is the pixel intensity value at image coordinate \((u, v)\) and \( n \) is number of bits used to represent the intensity values of each channel. Figure 1 describes this concept where, a scene is monitored over the time \( t = 1 \) to \( t = N \). Then, PMF of the pixel at coordinate \((u, v)\) is calculated for the first channel, for instance, red channel followed by for all other channels, for instance, blue and green channels. Similarly, the PMF for all other pixels in the scene respect to all the channels, also, can be calculated easily. Once the probabilistic based BG model of each pixel respect to concerned channels is calculated, then total likelihood probability of an incoming new pixel \((I_{u,v})\) to be the BG can be calculated as in (4.1).

\[
\Psi(I_{u,v}) = \prod_{c=1}^{D} P(I_{u,v}|I_{u,v}), \quad (4.1)
\]

where the conditional probability of the pixel is \( P(I_{u,v}|I_{u,v}) \), \( D \) is the dimension of the colour model, and \( c \) is the channel index. If it is RGB colour model then, \( D = 3 \) and \( C \equiv \{1 = Red, 2 = Green, 3 = Blue\} \). Now the new pixel \((I_{u,v})\) can be classified as either FG or BG based on the condition given in (4.2).

\[
FG_{u,v} = \Psi(I_{u,v}) < \tau_{u,v}, \quad (4.2)
\]

where \( \tau \) the threshold specific to the pixel at can be estimated in an unsupervised manner from the prior BG probabilities of the pixel respect to each channel as given in (4.3). The value, \( T_s \) determines the minimum prior probability of the pixel to be in the BG. Thus, if a new pixel has a likelihood probability, \( \Psi(I_{u,v}) \geq \tau_{u,v} \) then \( I_{u,v} \in BG \) else \( I_{u,v} \in FG \).

\[
\tau_{u,v} = \arg\min(P_R(I_{u,v}), P_G(I_{u,v}), P_B(I_{u,v})) \quad (4.3)
\]

Fig. 4.1 describes an example for a scene monitored over a discrete time interval \( t = 1, 2, ..., N \), where the PMF is calculated for a pixel \( I \) at coordinate \((u, v)\) for the first channel, in this case, the \( R \) (red) channel. Similarly, the PMF for all other pixels in the scene respect to all the channels, also, can be calculated easily.
4.3.2 Euclidean distance-based background suppression in 3D-color space

A point in 3D space can be, uniquely, represented as shown in Fig. 4.2 where \( P(x, y, z) \) and \( P'(x', y', z') \) are two points located at the coordinates \( (x, y, z) \) and \( (x', y', z') \), respectively.

In Fig. 4.2, \( \vec{OP} \) is the distance vector of the point, \( P \) from the origin, \( O \) and its magnitude is measured as in (4.4)

\[
|\vec{OP}| = \sqrt{x^2 + y^2 + z^2}. \tag{4.4}
\]

Similarly, magnitude of the distance vector, \( \vec{OP}' \) is calculated as \( |\vec{OP}'| = \sqrt{x'^2 + y'^2 + z'^2} \) by using (4.4). Then if \( P \) and \( P' \) refer the same point in the space, the Euclidean distance \( (ed) \) between them will be a null vector, \( 0 \). The Euclidean distance between any two points in a 3D space is measured as in (4.5).
\[ |\vec{ed}| = \sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2} = \delta, \quad (4.5) \]

where \( \delta \to 0 \) as \( P \) and \( P' \) come closer. Thus, to check if two points refer the same location in the space (5) can be used with a preferred precision (against a threshold value). This simple mathematical model can be used to classify background and foreground pixels in a scene if the scene is represented in a 3D-color space, for instance, RGB, YCbCr, YUV, or YIQ, like shown in Fig. 4.3. The reader may refer Douglas [69], Gonzalez [43], and Nixon [100] for detail of various colour models.

![Figure 4.3: Example of 3D-color space.](image)

Generally, in color spaces, the type of lights can be distinguished wholly in terms of human perception of colors; and each color has two technical aspects: (i) Luminance, the indication of the brightness of light, (ii) Chromaticity, the property that distinguishes red from blue and red from pink [69]. Thus, color spaces facilitate the specification of colors in some widely accepted standards. Technically, a color space is a specification of a coordinate system and subspace within that system, where a single point represents each color [43]. There are numerous color models in use today due to the fact that color science is a broad field that encompasses many areas of application. In this chapter, some of the interesting ones, namely, RGB, YCbCr, YIQ, and YUV are used. In the case of RGB color space, to eliminate illumination variance in the frames the model in (4.6) [148] is employed. The illumination variance generally occurs due to exposure of camera rapidly changes while scene capturing process. It will cause missed-classification of significant number of image pixels.

\[
\begin{bmatrix} \hat{R} \\ \hat{G} \\ \hat{B} \end{bmatrix}^T = \begin{bmatrix} \frac{(R - G)}{\sqrt{2}} \\ \frac{(R + G - 2B)}{\sqrt{6}} \\ \frac{(R + G + B)}{\sqrt{3}} \end{bmatrix}^T \quad (4.6)
\]
In this algorithm, from a set of collected frames prior to actual monitoring task a generalized BG model is formed for each channel by using median filtering at pixel level as in (4.7).

\[ I_{c,\psi}(u, v) = \text{arg } \text{median} \left( I_{c}(u, v, k) \right), \]  (4.7)

where \( k = 1, 2, ..., N \) the sequence of samples of a pixel at image coordinate \((u, v)\) of the channel \(c\) and \(\psi\) denotes the generalized background model of the channel. If \( M_{c,\psi} = \text{arg } \text{med}(I_{c,k}) \) then the overall generalized background model will be \( BG = [M_{R,\psi}, M_{G,\psi}, M_{B,\psi}] \). Then an incoming new pixel at image coordinate \((u, v)\) can be classified as foreground if it satisfies the condition in (4.8), which is similar to (4.5).

\[ \left| \vec{d}(u, v) \right| = \sqrt{\left[ M_{R,\psi}(u, v) - I_{R}(u, v) \right]^2 + \left[ M_{G,\psi}(u, v) - I_{G}(u, v) \right]^2 + \left[ M_{B,\psi}(u, v) - I_{B}(u, v) \right]^2} < \delta_{u,v}, \]  (4.8)

where the pixel level threshold \( \delta_{u,v} \) can be determined as in (4.9) from Euclidean distances of each pixel between the first frame and the subsequent frames till the last one collected in modelling the generalized BG in (4.7).

\[ \delta_{(u,v)} = \text{arg } \text{mean} \left[ \left| \vec{d}(u,v,1,2) \right|, \left| \vec{d}(u,v,2,3) \right|, ..., \left| \vec{d}(u,v,k-1,k) \right| \right] \]  (4.9)

4.4 Experimental results

This section provides the experimental results of the presented algorithms for two different video sequences in four different colour spaces: RGB, YCbCr, YIQ, and YUV. The WavingTree video sequence has 287 frames taken from Wallflower [147]. In this sequence, there is a tree swaying continuously causing a non-static background while a man entering the scene at frame no. 242 from the right and leaving to the left at frame no. 260. The WaterSurface video sequence is taken from perceptual computing datasets [79]. It has 560 frames of a waving sea at the background while a man entering the scene at frame no. 483 and staying at the middle of the scene for a while. In both the cases, the background is non-static. Figures 4.4 and 4.5 show visual outputs of the algorithms: Algorithm I simple probabilistic and Algorithm II Euclidian distance-based with respect to hand-segmented ground truths of notable few frames. The results also provide a comparison with two GMM-based BGS models; the standard GMM model introduced by Stauffer and Grimson [139] and the GMM model with Wavelet transformation proposed by Mukherjee et al. [91]. These two
models are denoted as oriGMM and wavGMM hereafter. Note that, wavGMM works on grayscale input images only.

Figure 4.4: Sample results for the WaterSurface dataset.

Figure 4.5: Sample results for the WavingTree dataset.

Table 4.1: Performance comparison of the proposed algorithms for WaterSurface dataset.

<table>
<thead>
<tr>
<th>Fr. ID</th>
<th>Ground Truth</th>
<th>Algorithm I</th>
<th>Algorithm II</th>
<th>oriGMM</th>
<th>wavGMM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RGB</td>
<td>YCbCr</td>
<td>YIQ</td>
<td>YUV</td>
<td>RGB</td>
</tr>
<tr>
<td>499</td>
<td>0.9815</td>
<td>0.9527</td>
<td>0.9743</td>
<td>0.9870</td>
<td>0.9849</td>
</tr>
<tr>
<td>511</td>
<td>0.9584</td>
<td>0.3774</td>
<td>0.6652</td>
<td>0.9253</td>
<td>0.9526</td>
</tr>
<tr>
<td>520</td>
<td>0.9807</td>
<td>0.9736</td>
<td>0.9829</td>
<td>0.8952</td>
<td>0.8579</td>
</tr>
<tr>
<td>530</td>
<td>0.8584</td>
<td>0.5439</td>
<td>0.7934</td>
<td>0.8902</td>
<td>0.8927</td>
</tr>
</tbody>
</table>

Table 4.2: Performance comparison of the proposed algorithms for WavingTree dataset.

<table>
<thead>
<tr>
<th>Fr. ID</th>
<th>Ground Truth</th>
<th>Algorithm I</th>
<th>Algorithm II</th>
<th>oriGMM</th>
<th>wavGMM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RGB</td>
<td>YCbCr</td>
<td>YIQ</td>
<td>YUV</td>
<td>RGB</td>
</tr>
<tr>
<td>244</td>
<td>0.9610</td>
<td>0.8907</td>
<td>0.9366</td>
<td>0.9830</td>
<td>0.9580</td>
</tr>
<tr>
<td>249</td>
<td>0.9466</td>
<td>0.2754</td>
<td>0.6428</td>
<td>0.9701</td>
<td>0.9838</td>
</tr>
<tr>
<td>254</td>
<td>0.9545</td>
<td>0.9774</td>
<td>0.9804</td>
<td>0.9186</td>
<td>0.9565</td>
</tr>
<tr>
<td>257</td>
<td>0.9505</td>
<td>0.4297</td>
<td>0.7765</td>
<td>0.9436</td>
<td>0.9707</td>
</tr>
</tbody>
</table>

The Tables 4.1 and 4.2 tabulate the performance of the proposed algorithms in comparison to the oriGMM and wavGMM, where the best figures are in red ink. We
can see that the Algorithm II shows consistent performance than Algorithm I. When an average performance is considered as shown in Fig. 4.6 across all the color-spaces and datasets, the Algorithm II achieves \( \sim 15\% \) improvement in terms of f-measure compared to oriGMM and wavGMM, while the Algorithm I shows competitive results only.

### 4.5 Discussion and conclusion

The results show that the computationally simple algorithms perform well on the tested two non-static BG video sequences in comparison to the two GMM-based algorithms. Although the algorithms find required thresholds which determine the FG without any fixed parameters, i.e., in a non-supervised manner, they are not robust enough for various scenarios. For instance, the video streams do not have set of initialization frames prior to actual monitoring or the video inputs have congested moving objects in the scene. It is because performance of the algorithms largely depends on the collected samples, which estimate each pixel value in the BG either by a conditional probability or Euclidian distance values. These models also do not have the ability to update themselves on-line to adopt the changes in the scene. In contrast, the GMM-based models have rules to absorb the scene progress since they model each pixel value as a mixture of \( k \) number of Gaussian distributions. Then, based on persistence and variance of each distribution, \( m|m < k \) distributions are chosen to represent the BG. However, in such models there are application depended parameters, such as learning rate and a threshold which decides the amount of scene to be fall into BG have to be tuned for the best performance. Therefore, the simplicity of...
the proposed algorithms is useful for indoor surveillance systems, where the cameras are stably fixed and used to monitor a predetermined zone. On the other hand, effect of various colour spaces is found to be applicable in these BG subtraction techniques, but there is no significant improvement compared to the results achieved using the original RGB colour space.

As for future improvement:

i A weight parameter can be introduced to control the prior probabilities of each intensity level at each pixel. For instance, if an intensity value is classified as FG in the current frame it will have less probable to be in BG at the same pixel coordinate, so its weight in the BG prior probability can be set to lower than its initial value. By doing so the model will have the knowledge to adopt a new BG intensity level and to remove an old intensity value, which becomes least probable as the scene evolve.

ii The Euclidean distance-based algorithm can be improved by taking variance information of each channel into the distance calculation like in Mahalanobis distance so that it will be able to adopt rapid scene changes.
Chapter 5

A Unified Threshold Updating Strategy for Multivariate Gaussian Mixture Based Moving Object Detection

5.1 Summary

Moving object detection is vitally used in video surveillance applications. Traditional Gaussian mixture model (GMM)-based background subtraction (BGS) methods usually perform well when the background is stationary. However, they require parameter tuning to deal with dynamic backgrounds, whose pixel values change over time. Notably, the threshold which determines the pixels associated with moving objects from the resultant of BGS. Since there is no ultimate solution this Chapter intents to present a novel idea to update the threshold of GMM based BGS with respect to color distortion, similarity and illumination measures in pixel level. Extensive experiments are carried out to demonstrate the effectiveness of the proposed method in comparison to some of the long-familiar GMM-based BGS methods in the literature. However, note that this chapter is not attempted to provide an approach for real-time operation, instead it investigates a potential way of utilizing the measures above to set a threshold automatically to detect moving objects in video sequences.

5.2 Introduction

Moving object detection is a crucial process in applications related to computer vision mainly in video surveillance. This process ignores trivial information in a scene and
raises attention to moving objects. In order to achieve this, over the past years, many algorithms have been proposed either based on a predictive or probabilistic mechanism \cite{89,105,156,161}. For example, approaches that utilize filters like Wiener \cite{147} come under the first category while the approaches which model the background (BG) based on probabilistic distributions, like Gaussian \cite{139} come under the second category. In general, all these approaches fall into three strategies namely pixel-, region-, and hybrid- based methods.

Among them the Gaussian distribution-based approaches have received greater attraction since Stauffer and Grimson \cite{139} proposed GMM for real-time tracking in a video surveillance. Due to its applicability, there have been several methods such as \cite{78,84,91,92,160} proposed to improve the performance of \cite{139}. Zhou et al. \cite{84} introduces Markov random field (MRF) in an iterative process to refine the foreground (FG) with expectation maximization (EM). Mukherjee et al. \cite{92} come up with a support weight mechanism (SWM) and histogram of gradients (HoGs) for better distance measurement and in \cite{91} they use wavelet-based decomposition and a variable number of clustering technique to improve the GMM. Yang et al. \cite{160} employ conditional random field (CRF) in spatial domain to support GMM based video segmentation. In \cite{78}, Lee attempts to incorporate incremental EM type of learning into a recursive filter such that parameter learning of each Gaussian follows a predefined schedule.

In all the standard GMM-based FG detection algorithms, the threshold which classifies the FG and BG is tuned exclusively for each video sequence \cite{144}. This approach does not perform effectively, in times, even after the threshold is tuned for the particular video due to illumination changes or when objects appear with similar color as the BG. To address this issue, this chapter attempts to achieve a unified model which exploits pixel-based color similarity and distortion measures along with illumination coefficient to update the threshold adaptively.

The rest of this chapter is organized to provide details of the algorithm are described in Section 5.3. The proposed method is applied for foreground localization (i.e., moving object detection) on various datasets and the results are presented in Section 5.4 while conclusions are drawn in Section 5.5.
5.3 Proposed Algorithm

5.3.1 Multivariate Gaussian Mixture Model

The Gaussian mixture model used in this work differs from the standard GMM introduced by Stauffer and Grimson [139] in the following ways:

i. It assumes that each channel is independent with unequal variance value while [139] assumes that each channel is independent with same variance value.

ii. It uses a new distance measure based on Bhattacharyya and Mahalanobis distances in choosing appropriate Gaussian components which model the BG while [139] uses Mahalanobis distance measure for the same purpose.

iii. In the standard GMM based BGS, FG classification is based on a fixed threshold which has to be tuned for better performance exclusive to each video sequence. This is the same technique used in some of the improved versions such as in Yang et al. [160], Lee [78], and Mukherjee et al. [91]. Contrastingly, in this work the threshold is automatically updated based on color distortion, color similarity, and illumination coefficient measures.

iv. This work also process a FG refinement based on FG estimation directly from pixel wise posterior probability.

![Block diagram](image)

**Figure 5.1:** Process-flow of the proposed algorithm.

Block diagram shown in Fig. 5.1 depicts the process flow of the proposed algorithm. The algorithm exploits a Multivariate Gaussian Mixture Model (MVGMM) for estimating the BG and probability of each pixel to belong to the BG. The probability calculation of a multivariate Gaussian distribution is expressed as in (5.1).

\[
p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} \Psi \Sigma^{-1} \Psi \right], \quad (5.1)
\]
where $x$ is a vector-valued random variable $X = [X_1...X_n]^T$, mean vector $\mu \in \mathbb{R}^n$, covariance matrix $\Sigma \in S^n_{++}$, and $n$ is the dimension of the multivariate variable. The $S^n_{++}$ is the space of symmetric positive definite $n \times n$ matrices and the coefficient $1/\left[(2\pi)^{n/2} \times |\Sigma|^{0.5}\right]$ is a normalizing factor, and $\Psi = (x - \mu)$. The multivariate variable in this case is pixel intensities correspond to each pixel values in RGB color space. As noted earlier channels of the color space are assumed to be independent and have unequal variance level. Thus, the covariance matrix of the color space become a diagonal matrix as in (5.2), which results an efficient determinant and inverse matrix computation of the covariance matrix.

$$\Sigma_{j,t} = \sigma^2_{j,t} I, \quad (5.2)$$

where $I$ is the identity matrix and $j$ is the pixel index of rolled input frame at time $t$ in a video sequence. Then, to determine the matching Gaussian distributions of each new pixel a new distance measure is used by utilizing parts of Bhattacharyya measure and Mahalanobis distance. It is because, among various distance measures such as Chi-squared, Mahalanobis, Euclidian, and Matusita, the Bhattacharyya measure is stable, unbiased, self-consistent, and applicable to any distribution [1]. In Bhattacharyya measure, similarity between two multivariate distributions is calculated as;

$$B = \frac{1}{8} (M_2 - M_1)^T \left[\frac{\Sigma_1 - \Sigma_2}{2}\right]^{-1} (M_2 - M_1) + \frac{1}{2} ln \left|\frac{\Gamma}{\Upsilon}\right|, \quad (5.3)$$

where $M_i$ is the mean vector $\Sigma_i$ is the covariance matrix of $i^{th}$ distribution $\Gamma = |(\Sigma_1 + \Sigma_2)/2|$ and $\Upsilon = \sqrt{|\Sigma_1||\Sigma_2|}$. Meanwhile, the Mahalanobis distance ($D_M$) of multivariate distribution is given by;

$$D^2_M = \sum_{i=1}^{n} [(x_i - \mu_i)/\sigma_i]^2, \quad (5.4)$$

which is already part of probability calculation of multivariate GMM as in (5.1). Thus, to alleviate computational burden the Bhattacharyya measure in (5.3) is modified as in (5.5).

$$newB = \frac{1}{8} \sum_{i=1}^{n} [(x_i - \mu_i)/\sigma_i]^2 + \frac{1}{f} ln \left(\sqrt{|\sigma_i|^2}\right), \quad (5.5)$$

where $f$ is the number features involve for example, if a 3-D color space is used then $f = 3$ and $\Sigma = \prod_{c=1}^{f} \sigma^2_c$. Introducing the logarithmic term of variance values provides better stability in the distance measure. Hence, accuracy in the BGS and FG
detection are enhanced. Then, parameters of the multivariate Gaussian distributions are updated if the following condition is met.

\[
\sum_{f=1}^{d} \text{new} B_{j,t} < \gamma \sum_{f=1}^{d} \sigma_{j,t},
\]

where \( \gamma \) is a control value set to in the range of 2.025 and \( j \) is the pixel index of rolled input video frame at time \( t \). Consequently, rest of the parameters are updated as suggested by Stauffer and Grimson \[139\];

\[
\begin{align*}
\mu_t &= (1 - \beta) \mu_{t-1} + \beta X_t, \\
\sigma_t^2 &= (1 - \beta) \sigma_{t-1}^2 + \beta (X_t - \mu_t)^T (X_t - \mu_t), \\
\omega_t &= (1 - \alpha) \omega_{t-1} + \alpha,
\end{align*}
\]

where \( \alpha \) and \( \beta \) are application dependent learning rates set to be in the range \([0, 1)\). For unmatched distributions means and variances remain unchanged while weight will be updated as \( \omega_t = (1 - \alpha) \omega_{t-1} \), i.e., reduced by the factor of \( (1 - \alpha) \). If a pixel does not match with any of the distributions, then the least weighted distribution is updated with: \( \omega_t = \omega_{\text{initial}}, \sigma_t^2 \) to a highest value, and \( \mu_t = X_t \). Considering that changes in the BG is sparse, then the BG can be represented by the distributions associated with higher weights. Thus, once the distributions are ranked in descending order of the weight matrix, and the first \( L \) distributions satisfying the following precedent are selected to represent the background;

\[
BG = \arg \min_{L} \left( \sum_{c=1}^{C} \omega_{t,c} > \text{Th} \right),
\]

where \( \text{Th} \) controls the minimum amount of data that form the BG. In the standard GMM based BGS algorithms, the threshold which classify the FG pixels in resultant of estiamted BG subtracted from current frame is set to a fixed value throughout the whole sequence. This approach causes missed-classification of pixels when the BG and FG pixels are similar in color. For instance, when the standard GMM model given in \[139\] is tuned for best performance for Watersurface video sequence, it fails to classify part of pixels inside the red rectangular region shown in Fig. 5.2 as FG since BG pixels in that region are also quite similar to the FG color. To address this issue we consider there is no ultimate solution for a problem and explore a method which can utilize color distortion (\( \partial \)), color similarity (\( \varpi \)), and illumination (\( \vartheta \)) measures to set unique threshold for every pixel in the current frame. Thus, the threshold \( \text{Th}1 \) is updated at pixel-level on the run as described in the following subsection.
5.3.2 Threshold updating

Setting a fixed threshold to classify FG and BG causes missed-classification due to color similarity, distortion, and illumination changes. In order to overcome this situation, the following hypotheses are taken.

i. The fixed threshold has to be lowered if a FG pixel in the current frame is very similar to BG pixel at the same coordinate, so that the pixel will be correctly classified as FG.

ii. Similarly, when a pixel experiences higher color distortion or illumination variance the threshold to classify the pixel to be FG has to be updated to account the changes.

Considering the aforementioned hypotheses, the fixed threshold $Th_1$ is modified as

$$Th_1 = Th_1 \times \mathfrak{N},$$

where $\mathfrak{N}$ is a control variable that adjusts the fixed threshold $Th_1$ having an initial value of 60 to adopt the changes and it is calculated by (5.12). We achieve the mathematical derivations through empirical method to support the stated hypotheses.

$$\mathfrak{N} = \frac{\text{abs}[\varpi - (\vartheta + \vartheta)]\vartheta + \vartheta}{\partial + \vartheta}.$$  

The color similarity measure ($\varpi$) is calculated by (5.13) based on CIEDE2000 Color-Difference aka ciede00 formula as described in Sharma et al. We would like to give credit to Sharma et al.
for the source code of the color-difference calculation made publically available for researchers. Hence, the color distortion (∂) is measured by (5.14) as described in [54].

\[ \varpi_{j,t} = \frac{abs(ciede00_{j,t} - \text{argMed}(ciede00))}{\text{argStd}(ciede00)}, \quad (5.13) \]

where \( \text{argMed}(\cdot) \) and \( \text{argStd}(\cdot) \) are the operations to extract median and standard deviation values from the calculated \( ciede00 \). Note that lower value of \( ciede00 \) indicates that the two pixels are very similar in terms of appearance. So for that pixel, considering the hypothesis-I, the threshold value has to be lowered so that chances of this pixel to be classified as FG increases.

\[ \partial_{j,t} = \sqrt{\left( \sum_{c=1,j,t} f I_{c,j,t} \right)^2 - \left( \sum_{c=1,j,t} f (I_{c,j,t} \times \mu_{c,j,t})^2 \right) \left( \sum_{c=1,j,t} f \mu_{c,j,t}^2 \right)}, \quad (5.14) \]

where \( I_{c,j,t} \) and \( \mu_{c,j,t} \) are the intensity value and running mean respectively of a pixel \( j \) respect to time \( t \) and channel. The illumination coefficient (\( \vartheta \)) is defined by (5.15).

\[ \vartheta_{j,t} = \frac{L_{j,t}^*}{a_{j,t}^*} + \frac{L_{j,t}^*}{b_{j,t}^*}, \quad (5.15) \]

where \( L_{j,t}^*, a_{j,t}^*, \) and \( b_{j,t}^* \) are values of each channel in Lab color space of a pixel \( j \) at time \( t \).

### 5.3.3 Foreground refinement

The detected FG is refined based on the following rules. First of all two different FG estimations are performed by (5.16) and (5.17).

\[ \text{FGE}_1 = abs(I_t - B_t) > Th_1, \quad (5.16) \]

where \( I_t \) and \( B_t \) represent pixel values in the current frame and reconstructed BG by (5.10) respectively at time \( t \) and \( Th_1 \) is the threshold value defined by (5.11). At the same time, another FG estimation is carried as in (5.17) by thresholding the probability \( p \) calculated in (5.1).

\[ \text{FGE}_2 = p > ciede00 \times |ln(\partial + 1/\vartheta)|, \quad (5.17) \]

where \( ciede00, \partial \) and \( \vartheta \) are color similarity, color distortion and illumination coefficient values, respectively. Then, to refine the FG outlier pixels (\( \zeta \)) when compared \( \text{FGE}_1 \) and \( \text{FGE}_2 \) are taken to another FG validation process with lower threshold as;
\( \zeta = \text{FGE}_1 \oplus \text{FGE}_2 \), \hspace{1cm} (5.18)

where \( \oplus \) represents a binary XOR operation.

\[
\text{FGE}_3 = \text{abs}(I_{\zeta,t} - B_{\zeta,t}) > \text{Th}_{1,\zeta} \times \left( \frac{\text{ciede00}_{\zeta}}{\text{ciede00}_{\zeta}} \right). 
\] \hspace{1cm} (5.19)

Finally, the validated FG is achieved by merging the FGs determined by (5.16) and (5.19) as

\[
\text{FG} = \text{FGE}_1 \cup \text{FGE}_3. 
\] \hspace{1cm} (5.20)

### 5.4 Experimental results

#### 5.4.1 Nature of the experiments

This section demonstrates the performance of the proposed method on various datasets as a comparison to other GMM-based algorithms: GMM [139], crfGMM [160], and Effective GMM (EGMM) [78]. A short description of the datasets used is given in Table 5.1. The experiments are carried out with number of mixture components \( k = 5 \) while other parameters were tuned for best results. Processing time per frame (PTPF) is recorded based on MATLAB R2015a on Windows 8 64bit, with i74770 CPU at 3.40 GHz PC. The proposed method utilizes a standard RGB to Lab color space conversion since the color similarity measure is based on Lab color space.

<table>
<thead>
<tr>
<th>Datasets (DS)</th>
<th>Sequence</th>
<th>Frame size</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a). Wallflower [147]</td>
<td>WavingTrees</td>
<td>120×160</td>
<td>Sawing tree in the BG as a person enters the scene.</td>
</tr>
<tr>
<td>(c). Carnegie mellon [129]</td>
<td>Camera motion</td>
<td>240×360</td>
<td>Man walks across as car enters the scene.</td>
</tr>
<tr>
<td>(d). Change detection [158]</td>
<td>Pedestrian</td>
<td>240×360</td>
<td>Pedestrians and cyclist are crossing across.</td>
</tr>
</tbody>
</table>

Table 5.1: Description of the datasets used for validation.

53
5.4.2 Visual results

Visual results of key frames are shown in Fig. 5.3 and 5.4, where columns from left to right show the input frame in RGB, ground truth, and results from the proposed method, GMM, crfGMM, and EGMM respectively.

Figure 5.3: Visual comparisons of the results for frames no. 250 - 1st row and no. 559 - 2nd row form datasets (a) and (b), respectively.

Figure 5.4: Visual comparisons of the results for frames 435 - 1st row, 626 - 2nd row, and 827 - 3rd row from datasets (c) - (e) respectively.

5.4.3 Quantitative analysis

The quantitative analysis exploit evaluation matrices described in (1.1), which is a weighted harmonic mean measure jointly with recall and precision. Table 5.2 tabulates the performance evaluation of the proposed and other algorithms, where the figures in red ink represent the best performances. According to the visual and quantitative comparisons shown in Figures 5.3, 5.4, 5.5, and 5.5b the proposed algorithm produces competitive results. It is because the standard GMM and its improved versions such as EGMM and crfGMM do not focus on automatically setting a threshold based on pixel level variations presently occur in terms of color and illumination. On the hand, crfGMM takes greater processing time due to repeated neighborhood
computations. Similarly, the EGMM also considerable computational cost since it employs incremental EM-based learning recursively to update parameter of each Gaussian. In our case, the computational cost higher than the GMM and EGMM due to validation process resulting in less than a frame per second when considered an
average processing time across all the experiments, while the standard GMM is able to reach higher than a frame per second. When an average performance is considered across all the data sequences, the proposed model gains $\approx 20\%$ improvement to the traditional GMM in terms of f-measure.

## 5.5 Conclusion

The proposed algorithm exploits color distortion, color similarity, and illumination variation measures to derive a mathematical expression to update pixel-based threshold per frame, which detects moving objects through multivariate Gaussian mixture model-based background subtraction. The results proved that the taken hypotheses and the empirical method used to drive the threshold automatically are valid for the datasets used in the experiments. Hence, the performance evaluation shows that the proposed algorithm is more robust than the compared algorithms in terms of f-measure. However, the standard GMM takes lesser processing time per frame compared to the proposed algorithm.

Future work can be dedicated to further exploring robust methods to utilize color distortion and similarity measures along illumination measures in the region and global levels to facilitate accurate foreground localization.
Chapter 6

Fusion-based Foreground Enhancement for Background Subtraction Using Multivariate Multi-model Gaussian Distribution

6.1 Summary

These days, detection of visual attention regions (VAR) such as moving objects has become an important pre-processing stage in many computer vision applications. In this chapter, we examine the problem of separating moving objects a.k.a. Foreground (FG) in a scene, which has a near-static background (BG). We address the difficulty in setting an adaptive threshold in the multi-model Gaussian-based BG-FG separation through a novel FG enhancement strategy by assimilating color and illumination measures. We formulate the problem mathematically by using a histogram of a fused feature of color and illumination measures. The proposed method is an extension of Chapter 5 and it improves the BG-FG separation by introducing the following items:

i A new distance measure to check if a pixel matches a Gaussian distribution.

ii A new strategy to enhance the primary resultant of traditional background subtraction (BGS) with fusion of color and illumination measures.

iii A computationally speedy histogram-based methodology to find an appropriate threshold adaptively to separate BG and FG.

iv A FG validation process through probability estimation of Multivariate Gaussian Model Distribution (MVGMD).
We test the proposed algorithm on five different video sequences. The experimental results demonstrate that the proposed approach works well in challenging conditions, at the same time, it performs competitively against state-of-the-art GMM-based algorithms and few other methods as well.

### 6.2 Introduction

![Figure 6.1: A scene and its useful region of interest.](image)

The availability of High Performance Computing (HPC) has enabled application of Computer Vision (CV) technology in several fields, for instance, Multimedia video-based surveillance, Industrial Automation [82], Automotive, and Transportation. In such applications, BG-FG separation module plays an integral role in guiding a vision-based system to ignore unwanted region of a scene being monitored by utilizing visual properties and bringing attention to moving objects, such as running cars and walking people. It is very important when it comes to content aware processing and analysis, like video compression of surveillance footage. For example, consider Fig. 6.1 where the sub-figures show: (a) an actual scene and (b) useful FG region, which is $\sim 25 - 30\%$ of the actual scene. This piece of information is valuable for a video compression algorithms to compress the FG and BG regions differently, whereby details of the useful FG region remain intact while the BG region undergoes higher rate of compression. Besides, once the FG region is defined, the fruitful low-level visual cues of the current scene can be extracted and then employed for high-level analysis, such as object indexing and retrieval, traffic monitoring (detecting, counting or tracking of objects) [165], human activity recognition (run, walk, jump, squat, etc.) [55], human-machine interaction [83], moving object tracking (many live sports telecasting channels have adopted this), scene classification, digital forensics [118, 157], and so forth.
The core issue in background subtraction and foreground localization is, what exactly models the BG. In recent literature, for this task, many algorithms have been proposed either with pixel-based, region-based, or a hybrid strategy using cues of both pixel and region. There are several surveys can be found on these methods; among them the articles by Bouwmans [16] and Xu et al. [168] are very comprehensive and good to refer when more detail is required. This thesis also provides review on these methods in Chapter 3.

6.3 The proposed algorithm

Before the discussion on the proposed algorithm the following subsection summaries the major steps involved in traditional GMM-based BG modeling.

6.3.1 The applied multivariate Gaussian mixture model

The GMM-based BG modeling introduced by Stauffer and Grimson [139] assumes that the variables, i.e., the color channels (Red, Green, and Blue) are independent and having the same variance values so that the Gaussian probability density function of the history of each pixel\(^1\) \(\{X_1, ..., X_t\}\) is given by

\[
P(X_t|\mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp(-\Delta).
\] (6.1)

Here, at time \(t\), \(X_t\) is a vector that contains values of intensities of each channel of a pixel, \(\mu_t\) is the mean value of each channel, \(\Sigma\) is the covariance matrix of the concerned multi-variable, the intensity values of the RGB color space, \(\Delta = -\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)\), \(\frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}}\) is a normalizing factor, and \(n\) is number of channels. The covariance matrix \(\Sigma\) becomes \(\sigma^2 I\) since the channels are assumed to be independent, where \(I\) is an identity matrix. The Gaussian distribution in (6.1) have to be evaluated for every pixel per frame basis and the parameters \(\mu_t\) and \(\Sigma\) need to be updated according to every new pixel observation.

Then, when \(K\) number of Gaussian distributions are used, the weights of the distributions at time \(t\), \(\omega_{k,t}\) are computed as follows if a pixel value falls within \(\tau \times \sigma\) of a distribution \(k\), where \(\tau\) is set to 2.0 - 3.0.

\[
\omega_{k,t} = (1 - \theta)\omega_{k,t-1} + \theta(M_{k,1}),
\] (6.2)

\(^1\) The pixel history is the pixel process over time about a pixel, \(\{x_0, y_0\}\). It records the pixel’s intensity or color values.
where \( \theta \) is the learning rate and \( M_{k,t} \) is 1 for the matching distribution and 0 for others. Thus, the parameters \( \mu_t, \sigma_t, \) and distribution weight \( \omega_t \) are updated with application specific learning rates \( \beta \) and \( \theta \) based on the following rules.

\[
\mu_t = (1 - \beta)\mu_{t-1} + \beta X_t. \tag{6.3}
\]

\[
\sigma_t^2 = (1 - \beta)\sigma_{t-1}^2 + \beta (X_t - \mu_t)^T (X_t - \mu_t). \tag{6.4}
\]

\[
\omega_t = (1 - \theta)\omega_{t-1} + \theta. \tag{6.5}
\]

At the same time, for unmatched distributions the means and variances remain unchanged and weight is updated like \( \omega_t = (1 - \theta)\omega_{t-1} \). In the case of non-matching distributions, the least possible distribution is updated as \( \omega_t = \omega_0 \) and \( \sigma_t^2 \) to a highest value, and \( \mu_t = X_t \). Then, the first \( D \) distributions of ordered Gaussian distributions with respect to the value of \( \omega/\sigma \) are selected as BG model, where

\[
BG = \arg\min_D \left( \sum_{k=1}^{D} \omega_k \right) > T. \tag{6.6}
\]

Here, \( T \) is a measure of the minimum set of the data that must be considered for the BG.

### 6.3.2 The proposed framework

The MVGMD is well exploited for BGS in [139]; however, the threshold value used to separate BG and FG in MVGMD based BGS algorithms require tuning for every new video feed. In order to circumvent this, the proposed algorithm takes advantage of color similarity, color distortion, and illumination measures for an effective FG enhancement. The color similarity and distortion measures have been used in industries to measure industrial color differences of fabricated materials. We explore another application of the measures in BGS by utilizing them to enhance the FG features and to find appropriate thresholds. The concept of using such measures is plausible since traditional GMM based BGS techniques tend to miss-classify pixels when the FG and BG pixels are similar in color or distorted by the brightness and illumination variation or by the capturing technology (including camera motion). We have successfully utilized the aforesaid measures in Chapter 5 and in [143] to estimate an appropriate threshold through empirical mathematical derivations, which controls the minimum amount of data that model the BG. However, the method presented in this work is entirely different as it takes advantage of feature fusion approach to enhance the primary FG feature and then uses a histogram-based strategy to find
an optimal threshold. The optimal threshold is then employed to extract a final refined FG. Thus, The proposed algorithms consists of three major processing stages: BG estimation and FG detection, FG feature enhancement, and FG refinement. The following subsections elaborate the stages, clearly.

### 6.3.2.1 Background estimation

Background estimation is the core stage in this algorithm. This algorithm employs an applied MVGMD, which is a slight variation from the GMM introduced by Stauffer and Grimson [139], in which they consider that the variables are independent with same variance value. However, in this work we assume that the channels are independent, but do not have the same variance so that concept of MVGMD is fully exploited. Besides that, Mahalanobis distance is used to check if a pixel matches a
GMM in [139], but in this work we introduce a new distance measure to pick the matching distribution(s).

The condition stated in [139] that checks if a pixel falls within $\tau \times \sigma$ of a respective distribution to update mixture parameters does not suffice in a multivariate environment. To address this, in this work, the condition is improved as in (6.7), which is a varied version of Bhattacharyya measure. Utilizing this measure is reasonable since among the affinity groups of distance measures: Chi-squared distance, Mahalanobis distance, Euclidian distance, and Matusita the Bhattacharyya measure is stable, unbiased, self-consistent, and applicable to any distribution [1].

$$\sum_{f=1}^{d} B_f < \nu \sum_{f=1}^{d} \sigma_f, \quad (6.7)$$

where $\nu$ is a control value which is set to 2.0 based on experiments, $\sigma_f$ is the standard deviation of the feature (i.e. a channel in RGB) $f$, $d$ is the number of features in the multivariate distribution, and $B_f$ is a new distance measure defined by (6.8). The choice of $\nu$ has slight impact on the model. Larger and smaller value of $\nu$ may cause some percentage of data points generated by Gaussian model to match and not to match, respectively, with the current pixel. The random resulting noise can be removed by a simple connected component analysis [139], [7].

$$B_f = \frac{1}{8} \sum_{i=1}^{n} \left[ \frac{X - \mu_i}{\sigma_i} \right]^2 + \frac{1}{d} \ln \sqrt{|\Sigma_i|}, \quad (6.8)$$

where $X$, $i$, and $n$ denote the vector value of a new pixel, the $i^{th}$ Gaussian distribution, and total number of distributions, respectively. If the condition is satisfied, then the parameters $\mu, \sigma$, and distribution weight $\omega$ are updated according to (6.3) - (6.5) as described in [139]. Once the parameters are updated accordingly, the background $BG_t$ at time $t$ can be estimated as (6.9).

$$BG_t = BG_{t-1} + \Omega_t, \quad (6.9)$$

where $\Omega$ denotes updated weight and mean multiplied matrix of the three channels (red, green, blue) and calculated as in (6.10). Note that, the initial background $BG_0$ is simply initialized as array of zeros.

$$\Omega_t = [\Omega_{R,t}; \Omega_{G,t}; \Omega_{B,t}]^T, \quad (6.10)$$
where \( \Omega_{R,t} = \sum_{k=1}^{K} \omega_{r,k} \times \mu_{r,k}, \Omega_{G,t} = \sum_{k=1}^{K} \omega_{g,k} \times \mu_{g,k}, \Omega_{B,t} = \sum_{k=1}^{K} \omega_{b,k} \times \mu_{b,k}, \) \( k \) is the \( k \)-th distribution in \( K \) number of matching distributions that model the current BG. Consequently, the foreground \( FG_{t} \) is extracted through general matrix subtraction operation as expressed by (6.11).

\[
FG_{t} = \text{abs} \left( CF_{t}^{*} - BG_{t}^{*} \right), \quad (6.11)
\]

where superscript * denotes the data have been converted to 1-D grayscale space(GS), \( CF_{t} \) is the current frame, and \( BG_{t} \) is the estimated BG at time \( t \). This conversion is required since our target is to get a 1-D binary foreground mask to identify the moving objects. The conversion from a 3-D RGB color space to 1-D gray scale is given by (6.12), which is a weighted sum of the \( R, G, \) and \( B \) components of a respective input. The weights are taken from NTSC standard for luminance, which are plethoraically used values in computer vision domain for transforming an 8-bit color image to 8-bit gray scale image.

\[
GS = 0.2989 [R] + 0.5870 [G] + 0.1140 [B], \quad (6.12)
\]

where \([.]\) denotes matrix of values.

### 6.3.3 Color similarity measure

To address the missed-classification of pixels when BG and FG are similar in color, it is necessary to analyze per pixel color similarity between the current frame \( CF_{t} \) and the previous frame \( CF_{t-1} \). Although, there are many other color similarity measures such as KL-divergence based algorithms they are computationally cumbersome and generally, work on image patches not on pixel-level [150]. Thus, we exploit CIEDE2000 color difference to compute color similarity, which provides a color similarity map in pixel-level with less complex computation. The CIEDE2000 formula was introduced to compute industrial color differences in the early 2000s, which is based on the CIE1976Lab* color space [128]. At this juncture we acknowledge openly available source code from Sharma et al.. Given a pair of color values in CIE1976Lab* space \( L_{1}^{*}, a_{1}^{*}, b_{1}^{*} \) and \( L_{2}^{*}, a_{2}^{*}, b_{2}^{*} \), the color similarity \( \Delta C_{s} \) is calculated as:

\[
\Delta C_{s}(L_{1}^{*}, a_{1}^{*}, b_{1}^{*}; L_{2}^{*}, a_{2}^{*}, b_{2}^{*}) = \sqrt{\sum_{f=\{L,C,H\}} \left( \frac{\Delta f}{k_{f}S_{f}} \right)^{2} + R_{T} \frac{\Delta C}{k_{C}S_{C}} \frac{\Delta H}{k_{H}S_{H}}} \quad (6.13)
\]

where \( f \) - feature refers to \( L \) - lightness, \( C \) - chroma, and \( H \) - hue, \( R_{T} \) is a hue rotation term used to correct deflection in the blue region of the ellipse axis direction for
visual perception \cite{172} and the weights or compensations for neutral colors are given by $S_L$, $S_c$, and $S_H$ for the visual perception action on the three features: lightness, chroma, and hue respectively. The correction factors $k_L$, $k_C$, and $k_H$ are related with observation environment and set to unity. $\Delta L$, $\Delta C$, and $\Delta H$ are respective channel differences between $CF_t$ and $CF_{t-1}$. The entire derivation of CIEDE2000 consists of twenty long expressions including the computation of aforesaid correction factors and channel difference coefficients. Since there are clear derivations available in \cite{128} and \cite{172}, we omit them in this chapter to save chapter length.

A block diagram shown in Figure 6.4 describes the process of calculating $\Delta C_s$, where $XYZ$ denotes the operation that converts $RGB$ color space into $CIE1931XYZ$ color space. This conversion is required since the input video frames we have for experiments are in $RGB$ color space and there is no direct conversion from $RGB$ to $CIE1976Lab^*$ color space.

The calculated color similarity measure at time, $t$ is taken as $FG2_t$:

$$FG2_t = \Delta C_{s_t}. \quad (6.14)$$

Note that higher the similarity value indicates a change in the monitored scene i.e. a moving object while a lower value indicates a stationary scene, i.e., the BG.

### 6.3.4 Color distortion measure

Global and local illumination and brightness changes cause color distortion. As the result of that, the traditional GMM based BS techniques suffer from false FG detection. To address this, we account the color distortion measure which is parameterized
by running mean of pixel process $S_t$, brightness sharpened value $V_t$, and brightness-weighted value $P_t$ in the normalized color channel at time $t$. This approach has gained its success in code-book model based applications [54]. The color distortion $\Delta Cd$ in the current frame $CF_t$ is measured as described in (6.15) - (6.19). This measurement is a variation from the color distortion method described in [54] to account the history of pixel process $H_t$ at time $t$.

$$H_t = H_{t-1} + CF_t,$$  \hspace{1cm} (6.15)

$$S_t = \sum_{f=1}^{d} (H_{f,t}/N_t)^2,$$  \hspace{1cm} (6.16)

$$V_t = \sum_{f=1}^{d} (H_{f,t}/N_t \times CF_{f,t}),$$  \hspace{1cm} (6.17)

$$P_t = V_t/S_t,$$  \hspace{1cm} (6.18)

$$\Delta Cd_t = \sqrt{\left[ \left( \sum_{f=1}^{d} CF_{f,t}^2 \right) - P_t \right]^2},$$  \hspace{1cm} (6.19)

where $H_t$ is a periodically accumulated pixel values over time $t$, $f$ represents the color features, $d$ is number of features, $N_t$ is the number of frames up to time $t$ accumulated in $H_t$. Thus, the estimated color distortion at time $t$ is assigned to $FG3_t$:

$$FG3_t = \Delta Cd_t.$$  \hspace{1cm} (6.20)

Note that higher the distortion value indicates a change in the monitored scene i.e. a probable moving object.

### 6.3.5 FG feature enhancement and detection

The traditional BGS approach does not effectively detect FG when the moving object and the BG possess same or similar color values. The detection is further affected due to noise created by illumination changes. Such unfavorable conditions can be addressed by the proposed techniques shown in Fig. 6.2 and Fig. 6.3. Here, an enhanced potential FG region is formed by fusion of the initial foreground region $FG1_t$, the color similarity map $FG2_t$, and the color distortion map $FG3_t$. This fusion process recovers most of the missing FG pixels in $FG1_t$ by additive and range
normalizing \( rNorm \) operations carried out, sequentially. The normalizing operation is nothing but to bring the resultant of additive operation in the range of values that can be stored in the number of bits used to represent pixel intensities in the raw image. Thus, the enhanced FG feature \( eFG \) is derived as in (6.21). The contribution of each of the features: \( FG_1, FG_2, \) and \( FG_3 \) in the FG detection is also analyzed and results are presented in section 4.4.

\[
eFG_t = rNorm(FG_{1t} + FG_{2t} + FG_{3t}). \tag{6.21}
\]

By checking \( eFG \) against a threshold value, an accurate FG can be determined. Finding an optimal threshold value to separate the FG from \( eFG \) is discussed after a short discussion on traditional way of BGS. In traditional BGS, the binary FG region is extracted as:

\[
FG_B = abs \left( CF - BG \right) > Th, \tag{6.22}
\]

where the threshold value \( Th \) is set experimentally depend on nature of the video sequence to get better performance. In fact, it is a grueling task to tune thresholds automatically to suit input video feeds. Although the aforementioned approaches are better than parametric models, still certain level of tuning is required for different set of input feeds. Thus, to overcome such shortcomings, in this work, the threshold is adaptively set in an online process by exploiting intensity histogram of the enhanced FG feature \( eFG \) as shown in Figure 6.5. In Figure 6.5

\[
Th_1 = i \mid f(i) = \arg \max(f), \tag{6.23}
\]

and

\[
Th_2 = \arg \max(i) \mid i \in hist(eFG). \tag{6.24}
\]
(i.e., $Th_1$ is the intensity value corresponds to the highest frequency and $Th_2$ is the maximum intensity value appeared in the histogram of $eFG$). Here, $eFG < Th_1$ is definite BG region and $eFG > Th_2$ is the definite FG region. It is because in $eFG$ the higher intensity values represent the moving objects and lower intensity values represent the BG. At the same time larger area in the scene is BG. This is an axiom in surveillance videos. However, these two values: $Th_1$ and $Th_2$ are not optimal to achieve noise reduced FG binary mask. So the optimal threshold $\tau_{opt}$ must be chosen between $Th_1$ and $Th_2$ and we compute it as described in (6.25). The expression is derived from empirical method with an axiom of considering that more number of blighter pixels appear in $eFG$ higher the threshold value can be set to separate the FG.

$$\tau_{opt} = Th_1 + \text{abs}(Th_2 - Th_1)/(-\log(\Psi)),$$

where

$$\Psi = \int_{i_1}^{i_2} f(i)di / \int_0^{i_2} f(i)di.$$  (6.26)

Here, $i_1$ and $i_2$ are two intensities chosen around the FG peak in the histogram shown in Fig. 6.5. The value $\Psi$ can be approximated discretely as $n/N$, where $n =$ number of pixels in the range $i_1$ to $i_2$ and $N$ is total number of pixels in the raw image. To simplify the calculation, $i_2$ is set to the maximum intensity appeared in the histogram of $eFG$ and $i_1$ is set to $i_2 - 1$. Thus, FG binary mask $FG_B$ is determined by (6.27).

$$FG_B = eFG > \tau_{opt}. $$ (6.27)

This proposed threshold method is computationally simple yet performs better than the sophisticated Kittler-Illingworth threshold algorithm noted as $KIth$ hereafter. Experiments were carried out by using $KIth$ in place of the proposed threshold technique without any other changes in the proposed FG detection method, and the results are analyzed in section 4.4. As an example, the detected FG binary mask by applying (6.27) for the frame $CF_t$ in Fig. 6.2 is shown in Figure 6.6 along with corresponding enhanced foreground $eFG$ and ground truth $FG_G$ on its left and right respectively. The binary mask $FG_B$ has, excessively, detected some regions compared to the $FG_G$. This can be overcome by a novel validation process described in the following subsection.
6.3.6 Foreground validation

The FG binary mask is determined effectively as described in the previous subsections, nevertheless, it has to be validated to overcome fault detection. In this work, we devise a method which utilizes the posterior probability of the pixels given by (6.1) to refine...
the FG determined by (6.27). The validation process is described in the following steps while Figure 6.8 depicts the entire process of the proposed framework.

Step 1. A potential FG region $FG_p$ is detected from the posterior probability $Pb$ as given in (6.28).

$$FG_p = Pb < \Gamma,$$  \hspace{1cm} (6.28)

where $\Gamma$ depends on initial learning rate $\beta$, color illumination coefficient $\Upsilon$, and a weighted color measure $h$ in $YCbCr$ color space as expressed in (6.29) - (6.31).

$$\Upsilon = \frac{Y}{Cb} + \frac{Y}{Cr} + \Delta C_d,$$  \hspace{1cm} (6.29)

where $Y, Cb$, and $Cr$ represent individual channel of the current frame in $YCbCr$ color space. Meanwhile, $\Delta C_d$ is color distortion measure given by (6.19).

$$h = \frac{|\Delta C_s - \arg \text{med}(\Delta C_s)|}{\arg \text{std}(\Delta C_s)},$$  \hspace{1cm} (6.30)

where $\Delta C_s$ is color similarity measure given by (6.13), $\arg \text{med}(.)$ and $\arg \text{std}(.)$ are the operations to find median and standard deviation values of $\Delta C_s$. By using (6.29) and (6.30), $\Gamma$ is derived as;

$$\Gamma = \beta \times \sqrt{[h - \Upsilon]^2} / \Upsilon.$$  \hspace{1cm} (6.31)

Step 2. By comparing $FG_p$ and $FG_B$ a set of outlier pixels $\varphi$ can be found. Then, for those set of pixels, value of $\Gamma$ is increased by 30% so that some missing potential FG pixels can be recovered and $FG_p$ is updated as $FG_p(\varphi) = Pb(\varphi) < \Gamma(\varphi) \times 1.3$.

Step 3. Similar to Step 2, another set of outlier pixels $\psi$ in $FG_B$ is found by comparing $FG_B$ with the updated $FG_p$. Then, for those set of pixels, the threshold $\tau_{opt}$ is increased by 50% so that some possible BG pixels can be removed and $FG_B$ is updated as $FG_B(\psi) = eFG(\psi) > \tau_{opt} \times 1.5$.
Step 4. Finally, a convex hull region $FG_C$ is formed on the updated $FG_p$ using a convex hull operator $\hat{C}(\Phi)$ and a binary AND logical operation is performed between $FG_C$ and the updated $FG_B$ to achieve an validated foreground region $FG_v$ as $FG_v = \hat{C}(\Phi) \cap FG_B$. For instance, the validated FG for the frame $CF_t$ given in Fig. 6.2 is shown in Fig. 6.7 along with respective $FG_B$ and $FG_C$.

Note that in Step 2 and 3 excessive increment of $\Gamma$ and $\tau_{opt}$ would cause pixel missed-classification; those values were set empirically in this work and fixed for all the experiments.

### 6.4 Experimental results

<table>
<thead>
<tr>
<th>Video Seq.</th>
<th>Benchmark database</th>
<th>Frame size</th>
<th>General description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water surface</td>
<td>Complex background [79]</td>
<td>120×160</td>
<td>A person is walking on a shore of a water body. Illumination changes.</td>
</tr>
<tr>
<td>Railway</td>
<td>Carnegie melon [129]</td>
<td>240×360</td>
<td>Man is walking across as a car enters the scene. A strong breeze causes the camera to be jittery during scene capturing.</td>
</tr>
<tr>
<td>Canoe</td>
<td>CD-net [158]</td>
<td>240×320</td>
<td>Exhibits dynamic background motion. People on a canoe paddle on waging water channel which has continuous ripples.</td>
</tr>
<tr>
<td>Highway</td>
<td>CD-net [158]</td>
<td>240×320</td>
<td>Tree branches sway as vehicles move on a road. Frequent natural illumination changes.</td>
</tr>
</tbody>
</table>

Table 6.1: Description of the video sequences used for validation.

The proposed algorithm is evaluated for its efficacy by comparing its performance on five different video sequences listed in Table 6.1. The comparisons are done qualitatively and quantitatively. Since the major motivation of the proposed method is to improve the accuracy of GMM-based BG-FG separation the comparisons are shown against the standard GMM, and its improved versions such as CRF, EGMM, and WavGMM. For these algorithms we use their source codes in MatLab to test on the stated data sequences, so their processing times are also compared. In addition, comparisons are made against non-GMM based algorithms as well, where the
results of the methods are adopted from [167]. The non-GMM based methods compared in this chapter are Bayesian object detection in dynamic scenes (BOD), Online robust dictionary learning (ORDL), Mairal et al. [88], Background subtraction via robust dictionary learning (BRDL) [179], Fuzzy color histogram (FCH), Lin et al. (RPCA) [81], Wang et al. (RPMF) [155], Xiaowei et al. (DECOLOR) [183], and Cheng et al. (S-SVM) [25]. For these methods their processing times are not compared, since they were not reported in the literature and we do not have their source codes.

Hence, experiments are carried out to demonstrate the effectiveness of the proposed threshold method by comparing the results with Kittler-Illingworth method of threshold (noted as Ours\textsuperscript{KIth} here after). Besides that, further analysis are presented:

i to see which feature either color similarity or distortion contribute the most in the FG feature enhancement,

ii to evaluate the improvements achieved through the assumption made earlier which states that all the channels are independent but do not have the same variance. For these two experiments visual results are avoided to limit length of the chapter, and

iii to elaborate the limitations of the proposed model.

6.4.1 Qualitative analysis

Visual results are shown in comparison to other methods from different categories on different data sequences. As stated earlier the results of non-GMM based algorithms are adopted from [167]. The quantitative evaluation is provided in subsection III-B.

Railway: This is one kind of surveillance video to monitor a railway crossing. As shown in Figure 6.9(a), car running on the road and man walking down the road should be separated as FG. Although, the BG is relatively static the task of FG separation becomes challenging due to camera jitter, illumination changes, and closer color similarity between FG and BG. In Figure 6.9, one can see that the proposed method effectively separated the FG regions in comparison to GMM, EGMM, BOD, S-SVM, and DECOLOR. It is found that methods like S-SVM and DECOLOR also produced very competitive results. Results for more challenging scenarios are provided in Figure 6.10 and 6.11.
Figure 6.9: Comparisons of foreground extraction for Railway data sequence: (a). original scene, (b). ground truth, and (c) to (i) show the detected FG of different methods GMM, EGMM, BOD, S-SVM, DECOLOR, Ours, and Ours\(^{Kth}\) respectively.

Figure 6.10: Comparisons of FG separation for WaterSurface data sequence: (a). original scene, (b). ground truth, and (c) to (i) show the detected FG of different methods GMM, EGMM, FCH, RPCA, DECOLOR, Ours, and Ours\(^{Kth}\) respectively.

*WaterSurface*: This video sequence is a regular outdoor video recorded at a shore, where the water surface has continuous waves and there is also considerable level of illumination changes that cause a non-static BG. As shown Figure 6.10(a) the man walking along the shore should be classified as FG. Figure 6.10 presents the BG-FG separation results of different algorithms, where we can see that the proposed method separated the FG region closer to ground truth and better than other algorithms.

*Pedestrians*: It is an outdoor video sequence which consists of pedestrians and bicyclist passing on a street segment, where frame to frame illumination changes drastically. Figure 6.11(a) shows results of some key frames for comparison, where the walking people and the passing bicyclist are considered as FG. In Figure 6.11(g), we can see that results of the proposed method are better than other algorithms shown in Figure 6.11(c) to (f). Although, the WavGMM \(^\text{91}\) performs quite well in

\[^3\] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html
Figure 6.11: Comparisons of FG separation on Pedestrians data sequence: Columns (a) to (h) show original significant scenes with frame IDs embedded, ground truth, and detected FG of different methods CRF, EGMM, GMM, wavGMM, Ours, and Ours\textsuperscript{Kth} respectively.

Separating FG, it produces accountable noisy FG when level of illumination changes gets high and it is obvious in frame no. 700, 846, and 942, for instance. Meanwhile, the proposed method is not affected by the illumination changes.

Highway: This is a typical traffic monitoring video of a surveillance application. As shown in Figure 6.12(a), cars moving on the highway are to be classified as FG and the rest of the scene including swaying tree branches at the side of the highway to be classified as BG. As we can see in Figure 6.12(g), the proposed approach successfully classified the FG region regardless of motion in the BG and unpredictable illumination changes. For instance, in frame no. 1654 and 1664 the swaying tree branches and in frame no. 1674 and 1700 the swaying tree branches and edge of the road due to illumination changes are also detected in a considerable amount as FG by the other algorithms, except the proposed method.

Canoe: A canoe crossing a water body, where the BG is very unstable due to waving water surface, swaying trees at the shore, and slight illumination changes. Besides, the captured video of the scene is in low quality, as well. Thus, separating the FG becomes very challenging in this video sequence and the compared algorithms detect lot of waving water surface as part of FG region. As shown in Figure 6.13(a), the moving canoe and the people in it are considered as FG and the rest of the scene to be BG. In Figure 6.13(g), we can see that the proposed algorithm is able to detect the FG region better than the other algorithms, though the scene is very challenging.
6.4.2 Quantitative analysis

The ground truth labels provided in databases of each respective data sequence can be utilized to analyze the performance of the proposed algorithm in pixel-level in terms f-measure and processing time per frame (PTPF). The f-measure or figure of merit (FoM) is a weighted harmonic mean measure with recall and precision given by (1.1).

Tables 6.2 - 6.5 compare performance of the proposed method with other algo-
Table 6.2: Average F-measure Comparison on Railway data sequence

<table>
<thead>
<tr>
<th>Algorithm</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM</td>
<td>0.4192</td>
<td>0.736</td>
<td>0.866</td>
<td>0.833</td>
<td>0.794</td>
<td>0.9084</td>
<td>0.8902</td>
</tr>
<tr>
<td>EGMM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORDL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRDL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mairal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours$^{Kth}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In tables 6.2 - 6.4, average f-measure of other algorithms are compared with the proposed method, where boldface values are the best results. The average values are calculated respect to the available GTs in each video sequence as listed in table 6.1. In Table 6.5, although wavGMM exhibits the best PTPF; it is because it works on 1-D grayscale image while others process each frame in 3-D RGB color space. On the other hand, it is understood that using KIth provides competitive results with our simple thresholding method it consumes extra processing time since KIth depends mostly on complex mathematical computation. Hence, further analyses are presented in Figure 6.14 - 6.17. Figure 6.14 shows f-measure comparison in terms of box-plots on three data sequences: Pedestrian, Highway, and Canoe, where we can see that the proposed method possess the best results. Figure 6.15 shows PTPF comparison on the same set of data sequences aforementioned, while Figure 6.16 describes pattern of PTPF of the proposed algorithm, GMM, CRF, EGMM and wavGMM, respectively for first 500 frames from respective data sequences. From this timing plot, one can see that the proposed method takes high processing time during settling period and afterwards its processing time falls into a closer range. In the future work, if the method can be improved through programming to lower the initial processing time then the over all average PTPF will be reduced. Note that the processing times are recorded when the respective algorithms are executed individually on a PC with Intel(R) Core(TM) i7-3770 CPU@3.40GHz, 16.0GB RAM, and 64-bit operating system.

Table 6.3: Average F-Measure Comparison On Watersurface Data Sequence

<table>
<thead>
<tr>
<th>Algorithm</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM</td>
<td>0.7647</td>
<td>0.534</td>
<td>0.721</td>
<td>0.730</td>
<td>0.840</td>
<td>0.8718</td>
<td>0.8440</td>
</tr>
<tr>
<td>EGMM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPCA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPMF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours$^{Kth}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Average F-Measure Comparison On Pedestrian, Highway, And Canoe Data Sequences

<table>
<thead>
<tr>
<th>Video</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedestrian</td>
<td>0.4857</td>
<td>0.5599</td>
<td>0.5492</td>
<td>0.6584</td>
<td>0.8507</td>
<td>0.8526</td>
<td>0.8526</td>
</tr>
<tr>
<td>Highway</td>
<td>0.5211</td>
<td>0.6071</td>
<td>0.5752</td>
<td>0.6423</td>
<td>0.8522</td>
<td>0.8456</td>
<td>0.8456</td>
</tr>
<tr>
<td>Canoe</td>
<td>0.3127</td>
<td>0.3866</td>
<td>0.4411</td>
<td>0.4840</td>
<td>0.8128</td>
<td>0.8128</td>
<td>0.6456</td>
</tr>
</tbody>
</table>
Table 6.5: Average PTPF (in Sec.) Comparison On Pedestrian, Highway, And Canoe Data Sequences

<table>
<thead>
<tr>
<th>Video</th>
<th>CRF</th>
<th>EGMM</th>
<th>GMM</th>
<th>WavGMM</th>
<th>Ours</th>
<th>Ours&lt;sup&gt;KLth&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedestrian</td>
<td>1.3922</td>
<td>0.2615</td>
<td>0.2428</td>
<td>0.0921</td>
<td>0.4844</td>
<td>0.5018</td>
</tr>
<tr>
<td>Highway</td>
<td>1.5385</td>
<td>0.2478</td>
<td>0.2734</td>
<td>0.0794</td>
<td>0.4732</td>
<td>0.5363</td>
</tr>
<tr>
<td>Canoe</td>
<td>1.6270</td>
<td>0.3458</td>
<td>0.2541</td>
<td>0.0780</td>
<td>0.4436</td>
<td>0.4865</td>
</tr>
</tbody>
</table>

Figure 6.14: F-Measure comparison on various input feeds vs methods.

6.4.3 Contribution of the features

In order to analyze the impact of the fused features in melioration of FG detection, the features are fused one by one with the main feature $FG_1$ and the resulted $FoM$ of FG detection with respect to level of feature fusion is tabulated. Figure 6.17 describes the fused features’ combined $FoM$ confidence intervals on all the data sequences through an error bar based on descriptive statistics of repeated measures by group, where the fused features are $f_1 \rightarrow FG_1, f_2 \rightarrow FG_1 + FG_3, f_3 \rightarrow FG_1 + FG_2$ and $f_4 \rightarrow FG_1 + FG_2 + FG_3$. One can see that fusion of $FG_3$ with $FG_1$ achieve better improvement around 2.5% than fusion of $FG_2$ with $FG_1$, while all feature fusion gains more than 5% improvements when comparing average $FoM$ of $FG_1$ alone.
6.4.4 Fully utilized MVGMD

From the above all results presented it is well understood that the proposed method improves the FG detection considerably. However, what is the impact caused by fully
utilizing a MVGMD which considers all the features are independent with different variance per feature?. To examine this, (a)- experiments are carried out by considering all the features have same channel variance referred as partially utilized MVGMD without any other changes in the proposed method and the results are compared in table 6.6 with (b)- the actual method presented in this chapter. The results show that using fully utilized version of MVGMD gains remarkable performance in terms of FoM.

Table 6.6: Impact of fully utilized MVGMD in terms of FoM

<table>
<thead>
<tr>
<th></th>
<th>Railway</th>
<th>WaterSurface</th>
<th>Highway</th>
<th>Pedestrian</th>
<th>Canoe</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>0.5617</td>
<td>0.8096</td>
<td>0.8150</td>
<td>0.3399</td>
<td>0.4784</td>
</tr>
<tr>
<td>(b)</td>
<td>0.9063</td>
<td>0.8460</td>
<td>0.8522</td>
<td>0.8507</td>
<td>0.8121</td>
</tr>
</tbody>
</table>

6.4.5 Limitations of the proposed model

Although the model performs better than other GMM-based algorithms compared in the chapter, it does not have a strategy to tackle with shadows of moving objects and sudden changes in region-level. We demonstrate this limitation using two video sequences in CD-net database: BusStation and Bungalows. These sequences contain scenes with prevalent hard and soft shadow and intermittent shades.

Two failed cases, one per each data sequence to show the limitation of the proposed algorithm are shown in Fig. 6.18 and Figure 6.19. In Figure 6.18, we can see that the
model has considered the moving shadow as FG and suffered from region-level sudden changes as people walking on the payment. The region-level sudden changes happen as at one instance, the moving object blocks the Sunlight and in the next instance, the region is exposed to the Sunlight. Due to such high sudden region-level changes and intermittent moving shadows the model achieves poor FoM of 0.7103 compared to the average performance of 0.8639 across tested baseline data sequences in Table \ref{table:results}.

Similarly, in Bungalows video sequence also the proposed model performs poorly with FoM of 0.5828, since every moving object appears in the scenes comes with nearly equal volume of shadow region. At the same time, the other GMM-based prior art algorithms compared in this work also suffer from this weakness. For instance, the WavGMM \cite{91}, which is the best among the others also weakly performs and achieves FoM of 0.5292 and 0.4015 on Bus station and Bungalows sequences respectively. On the other hand, the state-of-the-art algorithm the SuBSENSE \cite{135} produces average FoM of 0.8890 in these challenging sequences.

These experiments demonstrate the weakness of the proposed model. However, in the baseline datasets employed in the experiments, the moving shadows are considered as part of the foreground in the ground truth since the proportion of the shadow regions is small enough in comparison to the actual FG regions. Thus, the algorithm performs better on those data sequences. Note that, by employing a pre-processing
stage to remove shadows effectively at each frame can easily improve the proposed algorithms performance.

### 6.5 Conclusion

Although many researchers and scientists propose new algorithms for BG-FG classification, it is equally worth to come up with new techniques in order to meliorate the existing methods. Thus, This work presents a genuine effort of exploiting measures such as color similarity, color distortion, and scene illumination variation to enhance foreground features along with a novel adaptive threshold and foreground validation rule to improve BG-FG separation. The qualitative and quantitative analysis on the selected data-sets proves that the concept of fusing the aforesaid measures works effectively. The drawback of the algorithm exists in higher processing time due to the layered validation process, and it does not have a mechanism to counterbalance the effect of moving object shadow.

The future work, thus, would be dedicated to fixing the processing time from either the framework perspective or programming perspective, for instance, utilizing GPUs and handling the shadows jointly with BG modeling. In addition, thoughts can be given to exploiting color and illumination features in the local-temporal level instead of global-temporal level to enhance the foreground features and to extract the foreground region robustly.
Chapter 7

sEnDec: An Improved Image to Image CNN for Foreground Localization

7.1 Summary

Although it is not immediately intuitive that Deep Convolutional Neural Network (DCNN) can yield adequate feature representation for a Foreground Localization (FGL) task, recent architectural and algorithmic advancements in Deep Learning (DL) have shown that the DCNN can be a forefront methodology for this pixel-level classification problem. In FGL, the DCNNs face an inherent trade-off between moving objects, i.e., the foreground (FG) and the non-static background (BG) scenes, through learning from local- and global-level features. Driven by the recent success of innovative structures for image classification and semantic segmentation, this work introduces a novel architecture, called Slow Encoder-Decoder (sEnDec) that aims to improve the learning capacity of a traditional image-to-image DCNN. The proposed model subsumes two subnets for contraction (encoding) and expansion (decoding), wherein both phases it uses intermediate feature map up-sampling and residual connections. In this way, the lost structural detail due to spatial subsampling is recovered. It helps to get a more delineated FG region. Comparative experimental results on sixteen benchmark video sequences, including baseline, dynamic background, camera jitter, shadow effects, intermittent object motion, night time, and bad weather show that the proposed sEnDec model performs very competitively against the prior- and state-of-the-art approaches.
7.2 Introduction

Foreground localization is a fundamental task in various problems in the computer vision domain, like salient object detection and recognition [15], content-aware image/video processing [47], object segmentation [20], foreground object extraction [143], image quality assessment [28], visual tracking [184], object discovery [74], and human-robot interaction [46]. The main objective of FGL is to place a tight mask on the most probable region of pixels, i.e., the moving objects in a scene. Such mask is, in many ways, very informative than a simple detection with bounding box as it allows close localization of objects, which is important for applications, such as autonomous driving [178]. The FGL can be formulated as

\[
F_z = \frac{I_z - (1 - \alpha_z)B_z}{\alpha_z},
\]  

(7.1)

where for an observed pixel \(z\), the \(F_z\), \(I_z\), \(B_z\), and \(\alpha_z\) are the observed color, FG color, BG color, and alpha parameter of the capturing device, respectively. It has been automated with myriad algorithms, including graph-cut that requires a user supplied scribble or box on the FG and BG [86], probabilistic models like Gaussian Mixture Models (GMM) [186] and Markov Random Field (MRF) [187], and top-down approaches that firstly detect objects then classify pixels inside the detected object boundary based on shape priors [42]. Recently, DCNN-based approaches, like the image-to-image basic architecture depicted in Fig. 7.1 for localizing FG regions in video sequences have gained wider adoption [170].

![Figure 7.1: Block diagram of a basic image-to-image CNN.](image-url)
The applications of Neural Networks (NN) for visual-based problems possess a long history; arguably, from one of the pioneer computer vision systems, the Mark I Perceptron machine by Rosenblatt in late 1950s [14]. Presumably, concurrent with that Hubel and Wiesel’s [61] discovery of neural connectivity pattern of cat’s visual cortex inspires Fukushima to come up with a network, coined Neocognitron [39], which is invariant to image translations. The Neocognitron devised with a backpropagation mechanism paved a way for the modern-day DCNN, a multi-layered NN that integrates layers of several convolution, rectification, sub-sampling, and normalization operations. In which, the low-level convolutional (conv) layers operate similar to Gabor filters and color blob detectors [2] that extract primitive information, like edges and textures, while the top-level layers provide the abstractive meaning of the input visuals, like shapes and structures. In contrast to the traditional machine learning (ML) theories, the DCNNs emphasize automatic feature-extraction and learning from large amount of data. The practical theories of advanced CNN architectures was proposed by Hinton et al. [58]. Thence, the DCNNs have become the front runner for a numerous visual-based tasks followed by surpassing achievements of Hinton’s team comprising Alex Krizhevsky in the ImageNet 2012 Large-Scale Visual Recognition Challenge (ILSVRC) [72]. Hence, there has been an upsurge of success in applying DCNNs to semantic segmentation/labeling [130], [30], instance segmentation [178], medical image segmentation [109], [119], and so forth.

Thereupon we are interested in developing a deep CNN for the problem of FG object/region localization. To this end, the key contribution of this work as follows:

i Improving the learnability of a basic image-to-image CNN through a strategy, called slow-encoding and slow-decoding. In this, an input feature map at a stage in the contraction path is encoded twice before reaching completely to the next-level of reduced spatial dimension. Such each level of double encoding functions, like a micro auto-encoder. While in the expansion path each spatial dimension of decoded feature maps is enhanced by two sets of residual feature concatenations that bring to fuse two individual encoded feature maps of same spatial dimensions from the contraction path.

ii Providing empirical evidence to prove the effectiveness of the proposed strategy.

iii Thoroughly testing the proposed sEnDec network on various categories of video sequences and comparing its outcomes with prior- and state-of-the-art methods.
iv. It analyses four binarization methodologies, namely dataset-specific global threshold, Otsu’s threshold, Kittler-Illingworth thresholding, and segmentation using variational Bayesian estimation of a Gaussian mixture for creating binary FG mask from the probability saliency map.

Rest of this chapter organization as follows: Sections 7.3, 7.4, and 7.5 present the proposed model-sEnDec, the detail of experiments, and conclusion with future directions, respectively.

### 7.3 The proposed CNN architecture: sEnDec

![Schematic drawing of the proposed DCNN: sEnDec.](image)

In common with most of the DL-based pixel-level classifiers, the proposed sEnDec CNN has two architectural phases: encoding and decoding, where both stages exploit structured residual feature fusions. The model does not use any pooling or hidden FC layers but subsumes conv, convT, and cat layers networked to capture spatiotemporal contextual cues of FG objects.

A layer-wise depiction of the proposed image-to-image CNN is shown in Fig. 7.2. Where, the type of layer and its operation is differentiated with color notation as follows. Tender blue - input/output, tender red - conv that preserves the spatial dimensionality of its input, tender green - conv that reduces the spatial dimension by half of its input (subsampling), gray - depthwise cat, yellow - convT, and tender orange - conv that maintains its input spatial dimensionality and includes BN before ReLU operation. The network is capable of taking any spatial dimensions of images,
as it resizes them to 240 × 320 by using nearest-neighbor scaling algorithm to meet the input layer requirement. The input layer expects a three-channel data-feeds in Gray-scale, where the first two channels are the current frame and a frame before \( n \) timestamps, and the last channel is a generalized BG information. Hence, in the model, all the conv layers employ a filter of 3 \( \times \) 3 kernel with a stride rate of 1 in all directions, except the sub-sampling conv layers that perform convolutions with a stride rate of 2. The output feature map of a convolution \( C \) w.r.t. a kernel \( \omega \), its associated bias \( b \), and input image/patch (aka set of input activations) \( x \) is computed as:

\[
C(m, n) = b + \sum_{k=0}^{K-1} \sum_{l=0}^{K-1} \omega(k, l) \ast x(m + k, n + l),
\]

(7.2)

where \( \ast \), \( K \), \( \{m, n\} \), and \( \{k, l\} \) refer to convolutional operation, size of the kernel, origin of the patch, and element index of the filter respectively. The outputs of each conv layer are immediately processed by Rectilinear unit (ReLU) activation defined, as \( f(x) = \max(x, 0) \), where \( x \) is an input neuron.

![2D Convolution and Deconvolution](image)

**Figure 7.3:** Illustration of convolution and transpose (de) convolution operations: A 2D conv with \( K = 3, S = 2, \) and \( P = 1 \), and its corresponding convT with \( K' = K, S' = 1, \) and \( P' = K - P - 1 \).

The convT layer performs up-sampling, i.e., the gradient of 2D conv such that its output spatial dimension becomes twice as the input (expanded), as Fig. 7.3 illustrates without losing the connectivity pattern. The insight of using convT over image resizing process, is that it has trainable parameters. It is achieved by inserting zeros between consecutive neurons in the input receptive field, then sliding the conv kernel with unit stride [33]. If a conv operation is governed by kernel size of \( K \), stride rate of \( S \), padding size of \( P \), and its output has size of \( i' \) then the associated convT can be generated with a kernel \( K' = K \), stride \( S' = 1 \), padding \( P' = K - P - 1 \), \( \tilde{i}' \), and \( \alpha \), where \( \tilde{i}' \) is the size of the diluted input obtained by imputing \( S - 1 \) zeros

85
between each input neuron, and \( \alpha = (i + 2P - K) \mod S \) denotes the number of zeros inserted to the top and right edges of the input resulting an output map size of \( O' = S(i' - 1) + \alpha + K - 2P \).

The BN plays a vital role in avoiding training saturation around non-linearities and vanishing/exploding gradients. When an output feature map of a layer \( Z \in \mathbb{R}^{N,H} \), where \( N \) and \( H \) denote the number of samples in the batch and hidden neurons, respectively; then the normalization of \( Z \) is computed as

\[
\hat{Z} = \frac{Z - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{7.3}
\]

where \( \hat{Z}, \mu_B, \) and \( \sigma_B^2 \) refer to the normalized matrix, mean and variance of the batch, respectively.

### 7.3.1 Deep slow encoding

In contrast to the basic image-to-image CNN referred to Fig. 7.1, the proposed model encodes an input feature map twice such a way firstly the input is encoded by spatial sub-sampling conv, followed by up-sampling through conv\(T\) to match with the spatial dimension of the original input to the conv layer, then the up-sampled feature maps are concatenated depthwise with the original input features. At this stage, the new feature maps (after cat) are encoded to generate spatially halved output feature maps. This sequence of operations can be generalized, as when the output of \( \text{layer}_i \) is sub-sampled at \( \text{layer}_{i+1} \), the proceeding \( \text{layer}_{i+2} \) performs up-sampling and produces output feature maps with the same spatial dimension as the output of \( \text{layer}_i \). Since, the outputs of \( \text{layer}_i \) and \( \text{layer}_{i+2} \) have equal spatial dimensions they can be depthwise fused, like shown in Fig. 7.4. This chapter refers this deep slow encoding strategy as a block of \textit{micro auto-encoder}. There are three such blocks interconnected in the encoding subnetwork of the proposed sEnDec. The blocks are highlighted by enclosed broken blue lines in Fig. 7.2.

### 7.3.2 Deep slow decoding

The up-sampling subnetwork exploits a slow decoding strategy, whereby spatially matching two sets of feature maps from every stage of encoding subnetwork are concatenated through lost feature recovery mechanism, the residual feature flows to create high resolution representation. Each concatenation is interspersed with a layer that carries out conv, BN, and ReLU operations sequentially. This work denotes this sequential layer interconnections as a block of slow decoder. There are three such
blocks, as highlighted by enclosed broken red lines in Fig. 7.2. Thus, the hierarchical layers of transponse convolutional and symmetric residual feature cat operations extract the different level of textural and shape information of the scene. In this way, the network directly takes object-specific shape cues into account for FG object localization, which is often missed out in other approaches based only on conv layers [130]. It defines a non-linear local to global representation of objects based on their shape prior details.

In summary, the proposed slow encoder-decoder network takes a multi-channel input with dimension of $240 \times 320 \times 3$, terminates encoding process when the spatial dimension reaches of $15 \times 20$. Then it carries out the decoding sequentially with structured residual feature flows from lower to higher layers. Finally, the top layer generates a sigmoid probability map that has same spatial dimensionality as the bottom input layer. In total, the sEnDec utilizes 5,376,977 trainable parameters. Since sEnDec is trained with exemplar segmentation ground truths it does not require any additional inference aids, such as region proposals. Therefore, the network differs from scene understanding, where the co-occurrences of objects and other spatial and contextual properties are explicitly exploited to perform the FGL.

### 7.3.3 Training strategy

**Input:** To capture the spatio-temporal features we feed the input layer of sEnDec with gray-scale two consecutive frames ($f_t, f_{t-1}$) and a temporally median filtered generic BG model ($b_0$) stacked depthwise as depicted in Fig. 7.5. Where, the target segmentation, $y_t$ corresponds to the current frame, $f_t$. As this work uses the widely accepted benchmark database, Change detection (CDnet) [158], the initial frames
that have no segmentation ground truths are utilized to generate the BG model, $b_0$. The number of such frames used in this study is listed in Table 7.3.

![Figure 7.5: Input data configuration: G.T-Ground truth, I-Image, Med(\cdot)- Median filtering in temporal domain over collected $k$ samples prior to training, $b_0$ - The precomputed BG model, and $f_t$ - Input scene at time $t$.](image)

**Transfer learning:** Since the number of samples available with ground truths is less, it is crucial to initialize the network parameters, i.e., weights and biases appropriately. To this end, the sEnDec is trained on samples from one video sequence, then retrained on a target video sequence using previously learned parameters as initial values. This intra-class transfer learning approach provides better generalization of the model and enhances the FGL performance. The dataset pairs involved in this approach are tabulated in Table 7.1.

<table>
<thead>
<tr>
<th>Target</th>
<th>Source</th>
<th>Target</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>Traffic</td>
<td>Traffic</td>
<td>Highway</td>
</tr>
<tr>
<td>Office</td>
<td>CopyMachine</td>
<td>Boulevard</td>
<td>Highway</td>
</tr>
<tr>
<td>Pedestrians</td>
<td>Overpass</td>
<td>CopyMachine</td>
<td>Pedestrians</td>
</tr>
<tr>
<td>PETS2006</td>
<td>Overpass</td>
<td>PeopleInShade</td>
<td>Overpass</td>
</tr>
<tr>
<td>Canoe</td>
<td>Boats</td>
<td>BusStation</td>
<td>Pedestrians</td>
</tr>
<tr>
<td>Boats</td>
<td>Canoe</td>
<td>Sofa</td>
<td>BusStation</td>
</tr>
<tr>
<td>Overpass</td>
<td>Pedestrians</td>
<td>Transtop</td>
<td>Boulevard</td>
</tr>
<tr>
<td>Fall</td>
<td>Highway</td>
<td>SnowFall</td>
<td>Boulevard</td>
</tr>
</tbody>
</table>

Table 7.1: The dataset pairs involved in model initialization.

**Optimizer:** The network is trained to optimize the binary cross-entropy loss defined by (7.4), where the loss is added up over all the pixels in a mini-batch. We achieve this with Adam optimizer\(^1\) and a base learning rate of 0.0002.

\(^1\) It performs stochastic optimization by merging both optimizer RMSProp and AdaGrad. Refer [https://www.tensorflow.org/](https://www.tensorflow.org/)
\[ E = \frac{1}{n} \sum_{n=1}^{N} [p_n \log \hat{p}_n + (1 - p_n) \log (1 - \hat{p}_n)], \quad (7.4) \]

where it takes two inputs. First one is the output from the final layer of the network (refer conv.17 in Fig. 7.2) with dimension of \( N \times C \times H \times W \), which maps the probability predictions \( \hat{p}_n = \sigma(x_n) \in [0, 1] \) using Sigmoid non-linearity function \( \sigma(.) \) given by

\[ y = \frac{1}{1 + \exp(-x)}. \quad (7.5) \]

The second one is target \( p_n \in [0, 1] \) with the same dimension as the first one, where \( N, C, H, \) and \( W \) represent the batch size, number of channels, height, and width respectively of the inputs.

### 7.3.4 Binarization

It is important to create a binary mask from the probability map. In this work, three methods: a dataset specific global threshold (G-th) value empirically set in the range of \([0.15, 0.85]\), an automatic binarization approach, the Otsu’s method (O-th), and Gaussian smoothened version of Otsu’s method (OG-th) are used to generate binary masks. The Otsu’s technique iteratively computes the threshold value, \( \tau \) that lies in-between two peaks of the intensity histogram of a bi-model image, whereby intraclass variances are minimum \([104]\). The weighted sum of within-class variance is defined as

\[ \sigma^2_\rho(\tau) = \rho_0(\tau)\sigma^2_0(\tau) + \rho_1(\tau)\sigma^2_1(\tau), \quad (7.6) \]

where the weights \( \rho_0 \) and \( \rho_1 \) are the probabilities of BG and FG discriminated by a threshold \( \tau \), and the variances of these two classes are \( \sigma^2_0 \) and \( \sigma^2_1 \) respectively and they are computed as follows:

\[ \rho_0(\tau) = \sum_{i=1}^{\tau} P(i) \quad \& \quad \rho_1(\tau) = \sum_{i=\tau+1}^{T} P(i) \]

\[ \mu_0(\tau) = \sum_{i=1}^{\tau} \frac{iP(i)}{\rho_0(\tau)} \quad \& \quad \mu_1(\tau) = \sum_{i=\tau+1}^{T} \frac{iP(i)}{\rho_1(\tau)} \]

\[ \sigma^2_0(\tau) = \sum_{i=1}^{\tau} [i - \mu_0(\tau)]^2 \frac{P(i)}{\rho_0(\tau)} \quad \& \quad \sigma^2_1(\tau) = \sum_{i=\tau+1}^{T} [i - \mu_1(\tau)]^2 \frac{P(i)}{\rho_1(\tau)} \]

An explicit derivation of the method can be found in \([104]\). This binarization process is part of predicting/inferencing stage only as the numerical analysis is made on the binary FG mask.
7.3.5 Environment

The entire training and testing are carried out on an environment summarized in Table 7.2. The training and testing require about $16.5s$ and $15ms$ per sample respectively, when the batch size is set to 8. Thus, on that particular setting the speed of FG probability prediction is $\sim 67$ frames per second (FPS).

<table>
<thead>
<tr>
<th>#</th>
<th>Component</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>CPU</td>
<td>Intel(R) Core (TM) i7-6850K CPU @ 3.60GHz</td>
</tr>
<tr>
<td>2.</td>
<td>GPU</td>
<td>GeForce GTX 1080 Ti with 12GB memory</td>
</tr>
<tr>
<td>3.</td>
<td>RAM</td>
<td>Kingston DDR 4 2400 MHz 6412GB</td>
</tr>
<tr>
<td>4.</td>
<td>Software</td>
<td>Python with Keras using Tensorflow backend</td>
</tr>
</tbody>
</table>

Table 7.2: The environment.

7.4 Experimental setup, results, and discussion

7.4.1 Dataset

Experiments are conducted on sixteen video sequences from the CD-net [158] benchmark database. It consists of diversified change and motion, including near static and dynamic backgrounds, camera jittery, shadow, intermittent object motion, and conditions, like night times, and bad weather. Table 7.3 provides a succinct description of the datasets.

The Baseline represents a mixture of mild challenges, like subtle background motion, isolated shadows, swaying tree branches, and natural illumination changes. The Highway, Office, PETS2006 and Pedestrians sequences are from this kind. The Highway sequence contains high-volume of traffic flow on a highway with dynamic branches of trees and their shadows on the highway surface. The color of the vehicles oftentimes alike to the highway (dark gray) or shadows (black). The Office is an indoor scene that contains a near-stationary person but it experiences intensive illumination when his position or action have slight change. The Pedestrians sequence also faces strong illumination variation, which causes the foreground object appearance becoming similar to the background. The PETS2006 sequence contains crowded people walk up and down at a station with shining floors and soft shadows of the people casting on the floor.

The Dynamic background category contains video sequences that have strong dynamic background motion: boats and canoes on shimmering water, swaying tree
branches and leaves or a man walking on a park bridge in front of a shimmering water body. The *Canoe, Boats, Overpass* and *Fall* belong to this group.

The **Camera jitter** datasets have outdoor videos captured by vibrating cameras due to strong wind or unstable mount. The magnitude of jitter varies from one video to another. The *Traffic* and *Boulevard* sequences are from this type.

The **Shadow** category comprises indoor and outdoor scenes with strong as well as faint shadows. Here, some shadows are cast by moving objects as well. We take *CopyMachine, PeopleInShade,* and *BusStation* sequences from this category. Hence, we chose the *Sofa* and *Tramstop* sequences from the **Intermittent object motion** set that has videos containing background objects moving away, abandoned objects, and objects stopping for a short while and then moving away. Lastly, from the **Bad weather** category we select the *SnowFall* sequence for investigation.

### Training and test sets

To create exclusive training and test data samples, the available sequential frames that have ground truth segmentation samples of FG objects are split in an orderly manner such that the first 70% goes to the training set and the rest (30%) of the frames goes to the test set. This splitting approach is more appropriate as the images in the target datasets are frames from videos. Otherwise, a random selection of samples may pick a *frame* \( t \) for training set while picking a temporally closest frame, like *frame* \( t+1 \) for test set as depicted in Fig. 7.6. There can be many such instances.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Frame size ((W \times H))</th>
<th>Category</th>
<th>(N) frames w/t GT</th>
<th>(N) Frames for BG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>320 \times 240</td>
<td></td>
<td>1229</td>
<td>520</td>
</tr>
<tr>
<td>Office</td>
<td>360 \times 240</td>
<td>Baseline (BL)</td>
<td>1447</td>
<td>580</td>
</tr>
<tr>
<td>Pedestrians</td>
<td>360 \times 240</td>
<td></td>
<td>753</td>
<td>300</td>
</tr>
<tr>
<td>PETS2006</td>
<td>720 \times 576</td>
<td></td>
<td>900</td>
<td>300</td>
</tr>
<tr>
<td>Canoe</td>
<td>320 \times 240</td>
<td>Dynamic</td>
<td>342</td>
<td>840</td>
</tr>
<tr>
<td>Boats</td>
<td>320 \times 240</td>
<td>Background (DB)</td>
<td>6026</td>
<td>1930</td>
</tr>
<tr>
<td>Overpass</td>
<td>320 \times 240</td>
<td></td>
<td>440</td>
<td>2330</td>
</tr>
<tr>
<td>Fall</td>
<td>720 \times 480</td>
<td></td>
<td>564</td>
<td>1400</td>
</tr>
<tr>
<td>Traffic</td>
<td>320 \times 240</td>
<td>Camera jitter (CJ)</td>
<td>609</td>
<td>900</td>
</tr>
<tr>
<td>Boulevard</td>
<td>320 \times 240</td>
<td></td>
<td>1004</td>
<td>790</td>
</tr>
<tr>
<td>CopyMachine</td>
<td>720 \times 480</td>
<td>Shadow (SH)</td>
<td>1869</td>
<td>495</td>
</tr>
<tr>
<td>PeopleInShade</td>
<td>380 \times 244</td>
<td></td>
<td>829</td>
<td>280</td>
</tr>
<tr>
<td>BusStation</td>
<td>360 \times 240</td>
<td></td>
<td>832</td>
<td>300</td>
</tr>
<tr>
<td>Sofa</td>
<td>320 \times 240</td>
<td>Intermittent object motion (IO)</td>
<td>2243</td>
<td>495</td>
</tr>
<tr>
<td>Tramstop</td>
<td>480 \times 295</td>
<td>Night videos</td>
<td>1250</td>
<td>495</td>
</tr>
<tr>
<td>SnowFall</td>
<td>720 \times 480</td>
<td>Bad weather (BW)</td>
<td>656</td>
<td>795</td>
</tr>
</tbody>
</table>

Table 7.3: Dataset summary.
in this method, resulting in mere exclusiveness of training and test sets. Thus, it will lead the model to produce higher f-measure or FoM inappropriately. Figure 7.6 demonstrates the two methods, in which $n$ is the total number of samples in the sequence and $k = \text{round}(n \times 0.7)$ that is the dividing point (frame no.) for the ordered split.

7.4.2 Evaluation

The quantitative analysis is performed according to a standardized evaluation scheme, called $F - \text{measure}$. It ensures the performance of FGL as a weighted harmonic mean measure of recall and precision, i.e., size of the intersection divided by the union of the two regions. It is also referred as intersection-over-union (IoU) or figure of merit (FoM) and defined as defined by (1.1).

7.4.3 Step-by-step analysis

7.4.3.1 Sanity check

To validate the performance gain of our slow encoding-decoding strategy, firstly we compare performance of the basic structure (Fig. 7.1) with our improved architecture (Fig. 7.2). For this test, we choose three video sequences: two from the Baseline category (Highway and Office) and another from the Camera jitter category (Traffic). Both the cases, the models are trained from scratch. Table 7.4 and Bar-charts in Fig. 7.7 compare the sanity check results.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>G-th</th>
<th>O-th</th>
<th>OG-th</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>Basic</td>
<td>0.9244</td>
<td>0.9223</td>
<td>0.9205</td>
<td>0.9224</td>
</tr>
<tr>
<td></td>
<td>sEnDec</td>
<td>0.9545</td>
<td>0.9519</td>
<td>0.9506</td>
<td>0.9523</td>
</tr>
<tr>
<td>Office</td>
<td>Basic</td>
<td>0.9333</td>
<td>0.9343</td>
<td>0.9352</td>
<td>0.9343</td>
</tr>
<tr>
<td></td>
<td>sEnDec</td>
<td>0.9527</td>
<td>0.9551</td>
<td>0.9549</td>
<td>0.9542</td>
</tr>
<tr>
<td>Traffic</td>
<td>Basic</td>
<td>0.7650</td>
<td>0.7258</td>
<td>0.7317</td>
<td>0.7408</td>
</tr>
<tr>
<td></td>
<td>sEnDec</td>
<td>0.7853</td>
<td>0.7337</td>
<td>0.7419</td>
<td>0.7536</td>
</tr>
</tbody>
</table>

Table 7.4: Sanity check results of the proposed sEnDec compared to the basic model.

Figure 7.7: Bar chart comparison- sanity test results of the sEnDec and basic model: F-measure vs video sequences.

The sanity test on diversified three datasets shows that our sEnDec model improves the FGL nearly by 1% to 3% in terms of F-measure compared to the basic image-to-image CNN. Taking this as an empirical evidence, we proceed to investigate the performance of the sEnDec on all the datasets described in Section 7.4.1 with training strategy elaborated in Section 7.3.3.

7.4.3.2 Visual analysis

Performance of the sEnDec is investigated through visual comparisons between the predicted FG regions and the respected ground truths for all the test sequences and samples. However, it is irrational to show all the outputs here; thus, one sample per dataset is shown in Figures 7.8 and 7.9. From these comparisons, it is noticed that the proposed model has localized the moving objects closely to the ground truths. Nonetheless, it is necessary to numerically analyse for a quantitative validation.
Figure 7.8: Sample results. Row 1-8: Highway, Office, Pedestrians, PETS2006, Canoe, Boats, OverPass, and Fall video sequences. Col. 1-5: input frames, ground truths, sEnDec’s score-maps, binary FG masks generated with G-th and O-th, respectively.
Figure 7.9: Sample results. Row 1-8: Traffic, Boulevard, CopyMachine, PeopleInShade, BusStation, Sofa, Tramstop, and SnowFall video sequences. Col. 1-5: input frames, ground truths, sEnDec’s score-maps, binary FG masks generated with G-th and O-th, respectively.
7.4.3.3 Numerical analysis

Table 7.5 compares the performance of our sEnDec with some of the results published in the literature for prior-art and state-of-the-art techniques numerically. These methods include the probabilistic-based as well as NN-based approaches.

The proposed sEnDec shows a greater level of consistency when compared to the prior-art and state-of-the-art methodologies, based on the results tabulated in Table 7.5 and their mean average performances per category shown in Fig. 7.10. When it is further analyzed considering performances across all the datasets as elaborated via box-plots in Fig. 7.11, it is clear that the proposed model exhibits a stable performance that surmounts most of the existing approaches.

For instance, it gains the following improvements in terms of $f$-measure: 18%, 9%, 2%, 5%, 4.5%, and 4% when compared to EGMM (2004), PBAS (2012), IUTS-5 (2017), Lim et al. (2017), MBS (2017), and DBFCN (2018) respectively. Hence, compared to the sanity check results (trained with random initialization) of Traffic sequence, the transfer learning approach provides $\approx 10\%$ improvement. This percentage can be varied for other sequences based on the nature of the scenes.

Figure 7.10: Sequence category-wise performance analysis.
<table>
<thead>
<tr>
<th>Method</th>
<th>Highway</th>
<th>Office</th>
<th>Pedestrians</th>
<th>PETS2006</th>
<th>Canoe</th>
<th>Boats</th>
<th>OverPass</th>
<th>Fall</th>
<th>Traffic</th>
<th>Boulevard</th>
<th>CopyMachine</th>
<th>PeopleInShade</th>
<th>BusStation</th>
<th>Sofa</th>
<th>Tramstop</th>
<th>SnowFall</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGMM</td>
<td>0.9038</td>
<td>0.9451</td>
<td>0.9535</td>
<td>-</td>
<td>0.9217</td>
<td>0.9412</td>
<td>0.9573</td>
<td>0.9557</td>
<td>0.9558</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBAS</td>
<td>0.6564</td>
<td>0.9420</td>
<td>0.9686</td>
<td>0.9586</td>
<td>0.9719</td>
<td>0.9236</td>
<td>0.9713</td>
<td>0.9700</td>
<td>0.9702</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IUTIS-5</td>
<td>0.9597</td>
<td>0.9363</td>
<td>0.9669</td>
<td>-</td>
<td>0.9566</td>
<td>0.8394</td>
<td>0.9514</td>
<td>0.9404</td>
<td>0.9429</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lim</td>
<td>0.8327</td>
<td>0.8736</td>
<td>0.9354</td>
<td>-</td>
<td>0.8648</td>
<td>0.9059</td>
<td>0.9465</td>
<td>0.9327</td>
<td>0.9362</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MBS</td>
<td>0.8851</td>
<td>0.7196</td>
<td>0.9462</td>
<td>-</td>
<td>0.9345</td>
<td>-</td>
<td>0.9554</td>
<td>0.9379</td>
<td>0.9398</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBFCN</td>
<td>0.7474</td>
<td>0.3611</td>
<td>0.7332</td>
<td>-</td>
<td>0.9041</td>
<td>-</td>
<td>0.9262</td>
<td>0.9227</td>
<td>0.9224</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours: sEnDec</td>
<td>0.8673</td>
<td>0.7925</td>
<td>0.9272</td>
<td>-</td>
<td>0.8990</td>
<td>-</td>
<td>0.9355</td>
<td>0.9178</td>
<td>0.9188</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.4239</td>
<td>0.8714</td>
<td>0.9361</td>
<td>-</td>
<td>0.5668</td>
<td>0.8203</td>
<td>0.9292</td>
<td>0.9299</td>
<td>0.9303</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.6137</td>
<td>0.7485</td>
<td>0.8302</td>
<td>-</td>
<td>0.6781</td>
<td>-</td>
<td>0.8701</td>
<td>0.8407</td>
<td>0.8419</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5654</td>
<td>0.6602</td>
<td>0.7680</td>
<td>0.7990</td>
<td>0.8672</td>
<td>-</td>
<td>0.9284</td>
<td>0.9300</td>
<td>0.9300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.6597</td>
<td>0.8727</td>
<td>0.9260</td>
<td>-</td>
<td>0.8711</td>
<td>-</td>
<td>0.9402</td>
<td>0.9409</td>
<td>0.9409</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8819</td>
<td>0.8919</td>
<td>0.9103</td>
<td>-</td>
<td>0.9016</td>
<td>-</td>
<td>0.9535</td>
<td>0.9549</td>
<td>0.9552</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.7874</td>
<td>0.8609</td>
<td>0.8826</td>
<td>0.8565</td>
<td>0.8695</td>
<td>-</td>
<td>0.8886</td>
<td>0.8747</td>
<td>0.8753</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.6524</td>
<td>0.7381</td>
<td>0.7914</td>
<td>-</td>
<td>0.8455</td>
<td>0.8645</td>
<td>0.9240</td>
<td>0.9210</td>
<td>0.9235</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5781</td>
<td>0.8243</td>
<td>0.6080</td>
<td>0.8780</td>
<td>0.8856</td>
<td>-</td>
<td>0.8899</td>
<td>0.8294</td>
<td>0.8284</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.7631</td>
<td>-</td>
<td>0.8453</td>
<td>-</td>
<td>0.8782</td>
<td>-</td>
<td>0.9408</td>
<td>0.8643</td>
<td>0.8716</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.5: F-measure-based performance comparison: G-th, O-th, and OG-th stand for global, Otsu, and Gaussian smoothed Otsu threshold methods applied, respectively. The values in red and blue are the best and the second best figures, respectively.
7.4.4 Extended experiment

This section is part of our submitted conference paper in GreenCom 2018. It refers the SEnDec as Double Encoding - Slow Decoding (DESD) CNN. Thus, here after we use DESD instead of SEnDec. This extended experimental study is to investigate the impact of various approaches for transforming the salient map, i.e., probability scores generated by the proposed network to binary FG mask. To this end, the binarization process is carried out with:

i A dataset specific global threshold (G-th) value is empirically set in the range of [0.15, 0.85].

ii Otsu’s method (O-th) as described in [104].

ii Kittler-Illingworth thresholding (KI-th) as derived in [71].

iv A variational Bayesian estimation of a Gaussian mixture (BGM) distribution as in [95, 98].

The Otsu’s technique iteratively computes a threshold value that lies in-between two peaks of the intensity histogram of a bi-model image, whereas Kittler-Illingworth’s
method formulates each cluster or class as individual Gaussian-distributions with a mean and variance independent of the chosen threshold and uses a novel principal to minimize the clustering error (minimum error principle). The BGM is a variant of the GMM with variational inference algorithm that maximizes a lower bound on model evidence rather than data likelihood [185].

This extended experiment is conducted on ten video sequences chosen from Table 7.3. The subjective and objective studies as follow.

7.4.4.1 Subjective analysis

![F-measure vs model](image)

Figure 7.12: F-measure vs model: Performance analysis across all the video sequences.

The subjective evaluation of the DESD is performed through visual comparisons between the predicted FG and the respected ground truth. To manage the presentation of this chapter, one sample per dataset is shown in Fig. 7.16. From these comparisons, it is noticed that the proposed model has identified the FG objects tightly to the ground truths, regardless of the binarization method. The following subsection provides the objective analysis in terms of f-measure.

7.4.4.2 Objective analysis

Table 7.6 compares the performance of our DESD CNN with some of the results published in the literature for prior-art and state-of-the-art techniques. These methods include the traditional approaches as well as NN-based models. Referring to
Table 7.6 and their performances across all data sequences shown in Fig. 7.12, the proposed network shows a greater level of consistency when compared to the prior-art and state-of-the-art methodologies. It is clear that the proposed model exhibits a stable performance that surmounts most of the existing approaches. Among the traditional approaches the IUTS-6 (2017) achieves the best results, while among the DL-based models the MBS (2017) performs better. From these best models, the proposed network improves the FG identification by $\sim 3\%$ and $\sim 5\%$, respectively.
Hence, compared to the sanity check results (trained with random initialization) of Traffic sequence, the transfer learning approach provides $\approx 10\%$ improvement. This percentage can be varied for other sequences based on the nature of the scenes.

Figures 7.13 and 7.14 investigate performance of various binarization algorithms that transform the salient maps to binary FG mask. Among them, the global thresholding (G-th) performs well with the best f-measure and processing time per frame. At the same time, the variational Bayesian Gaussian distribution (BGM) method records the worse performance. The Otsu’s (O-th) and Kittler-Illingworth’s (KI-th) methods performs competitively between them in terms of processing time; however, O-th outperforms KI-th in f-measure significantly. In over all, considering the model’s inference time and the FG binary mask generation the proposed system produces very high speed of foreground localization as shown in Fig. 7.15.
Figure 7.16: Subjective analysis on the CD-net benchmark database. Sample results - Row 1-10: Highway, Office, Pedestrians, PETS2006, Fall, Traffic, Boulevard, BusStation, Sofa, and Tramstop video sequences.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>0.9038</td>
<td>0.9451</td>
<td>0.9535</td>
<td>-</td>
<td>0.9217</td>
<td>0.9412</td>
<td>0.9629</td>
</tr>
<tr>
<td>Office</td>
<td>0.6564</td>
<td>0.9420</td>
<td>0.9686</td>
<td>0.9586</td>
<td>0.9719</td>
<td>0.9236</td>
<td>0.9713</td>
</tr>
<tr>
<td>Pedestrians</td>
<td>0.9597</td>
<td>0.9363</td>
<td>0.9669</td>
<td>-</td>
<td>0.9566</td>
<td>0.8394</td>
<td>0.9541</td>
</tr>
<tr>
<td>PETS2006</td>
<td>0.8327</td>
<td>0.8736</td>
<td>0.9354</td>
<td>-</td>
<td>0.8648</td>
<td>0.9059</td>
<td>0.9465</td>
</tr>
<tr>
<td>Fall</td>
<td>0.4239</td>
<td>0.8714</td>
<td>0.9361</td>
<td>-</td>
<td>0.5668</td>
<td>0.8203</td>
<td>0.9371</td>
</tr>
<tr>
<td>Traffic</td>
<td>0.6137</td>
<td>0.7485</td>
<td>0.8302</td>
<td>-</td>
<td>0.6781</td>
<td>-</td>
<td>0.8842</td>
</tr>
<tr>
<td>Boulevard</td>
<td>0.5654</td>
<td>0.6602</td>
<td>0.7680</td>
<td>0.8990</td>
<td>0.8672</td>
<td>-</td>
<td>0.9395</td>
</tr>
<tr>
<td>BusStation</td>
<td>0.7874</td>
<td>0.8609</td>
<td>0.8826</td>
<td>0.8565</td>
<td>0.8695</td>
<td>-</td>
<td>0.8920</td>
</tr>
<tr>
<td>Sofa</td>
<td>0.6524</td>
<td>0.7381</td>
<td>0.7914</td>
<td>-</td>
<td>0.8455</td>
<td>0.8645</td>
<td>0.9358</td>
</tr>
<tr>
<td>Tramstop</td>
<td>0.5781</td>
<td>0.8243</td>
<td>0.6080</td>
<td>0.8780</td>
<td>0.8856</td>
<td>-</td>
<td>0.8899</td>
</tr>
</tbody>
</table>

Table 7.6: F-measure performance comparison. G-th, O-th, KI-th, and BGM stand for binarization methods with global, Otsu and Kittler-Illingworth thresholdings, and variational Bayesian Gaussian distribution, respectively. The values in red and blue are the best and the second best figures, respectively.
7.5 Conclusion

The deep learning architectures seamlessly have a more significant impact on our entire society, the task of creating them better will become ever-more critical. Thence, this work introduces new structural improvements to the basic image-to-image DCNN. Unlike classification, much computation in foreground localization is spent in optimizing background modeling and post-processing stages to get a few more valid pixels in the foreground. For this reason, we advocate that the proposed slow encoder-decoder DCNN is an elegant solution for accurate and real-time foreground localization.

To evaluate the performance of the sEnDec DCNN, we tested on sixteen challenging video sequences collected from benchmark CD-net database. The experimental results show that the implemented model performs better than or very competitive to the prior- and state-of-the-art methods without any ad-hoc post-processing. Thorough experiments are also carried out to investigate the necessity of appropriate model parameter initialization and type of binarization.

This network can apply to many other applications, including intelligent vehicular technologies and medical image segmentation. Besides that, the model can be further optimized for the number of filters and layers, thus to optimize the number of learnable parameters. Finally, it must be noted that a perfect FGL is still an intriguing task and a good FGL system should use the knowledge derived from its ultimate purpose.
Chapter 8

An Improved Video-foreground Extraction Strategy Using Multi-view Receptive Field and EnDec CNN

8.1 Summary

Foreground (FG) pixel labeling plays a vital role in video surveillance. Recent engineering solutions have attempted to exploit the efficacy of deep learning (DL) models initially targeted for image classification to deal with FG pixel labeling. One major drawback of such strategy is the lacking delineation of visual objects when training samples are limited. To grapple with this issue, we introduce a multi-view (receptive field) fully convolutional neural network (MV-FCN) that harness recent seminal ideas, such as, fully convolutional structure, inception modules, and residual networking. Therefrom, we implement a system in an encoder-decoder fashion that subsumes a core and two complementary feature flow paths. The model exploits inception modules at early and late stages with three different sizes of receptive fields to capture invariant features to various scales. The features learned in the encoding phase are fused with appropriate feature maps in the decoding phase through residual connections for achieving enhanced spatial representation. These multi-view receptive fields and residual feature connections are expected to yield highly generalized features for an accurate pixel-wise FG region identification or localization.

It is, then, trained with sequence-specific segmentation samples to predict desired FG objects. The comparative experimental study on eleven benchmark video sequences validates that the proposed model achieves very competitive performance...
with the prior- and state-of-the-art algorithms. The experiments cover various conditions, like input configurations with a single frame 3-channel (RGB) and two consecutive grayscale frames stack with a grayscale generic background model. It also reports that how well a transfer learning approach can be used to enhance the performance of the proposed MV-FCN.

8.2 Introduction

Foreground region labeling is a crucial task in video surveillance used to detect moving objects in challenging conditions. It requires robust algorithms to handle varying environmental factors, like illumination changes and dynamic backgrounds [16]. It is also an integral part of various machine-vision problems, such as object segmentation [20], [143], [3], image quality assessment [28], object discovery [74], visual tracking [184], and human-robot/machine interaction [46]. The primary objective of FG labeling is to place a tight mask on the most probable regions, in which moving objects mostly humans and vehicles can be identified. Such mask is, in many ways, very informative than a simple detection with bounding box as it allows close localization of objects, which is essential for retrieval, recognition, autonomous driving, and object preserved data compression for cloud-based systems [178]. Besides, obtaining pixel-level foreground labels is also an important step towards general machine understanding of scenes. An example application setup is drawn in Fig. 1.3 to conceive the importance of this work.

Much attention has been paid to automate this process; and thus, there has been myriad of algorithms proposed that mainly include statistical approaches. For instance, Gaussian mixture models (GMM) [94], clustering algorithms, like conditional random field (CRF) [187] and graph-cut [86]. However, some researchers focus on neural network (NN) models, like Self-Organizing Maps (SOM) [181] for this task. Here, a reasonable approach is to formulate it as a structured output problem that can be solved by training a system in an image-to-image fashion. This approach has been adopted in recent years’ deep convolutional neural networks (DCNN/ deep convnets) for FG region labeling and gained wider acceptance.

One of the main challenges in DCNN-based methods is that dealing with objects of very different scales and the dithering effect at bordering pixels of FG objects. To deal with these challenges, we propose a new model inspired by Google introduced Inception module [141] that performs convolution of multiple filters with different scales on the same input by simulating human cognitive processes in perceiving multi-scale
information and Microsoft introduced ResNet [56] that acts as a lost feature recovery mechanism. To enhance the knowledge of proposed network, we exploit intra-domain transfer learning that boosts the correct FG region prediction. Using this methodology is also inspired by human-like reasoning, in which the network learns new task precisely and more quickly by applying already learned knowledge, i.e., the weights and biases [154].

Subsequently, the deep CNNs have been effectively exploited for semantic segmentation/labeling [130], instance partitioning [178], and medical image sectioning [109], [119]. Thereupon we are interested in implementing a DCNN for the problem of FG object/region identification. Thus, the key insight of this chapter is to propose a deep convnet that enhances feature learning for a better FG-region localization based on novel strategies introduced in the recent literature.

Here, we formulate the FGL problem as a binary classification task, where a DCNN is trained end-to-end with exemplary FG segmentations to predict the most probable region of moving objects in a given input frame. The proposed model is a multi-view receptive field fully convolutional neural network taking advantage of ResNet-like connections and Inception-like modules. It has two architectural phases: an encoder and a decoder that mainly combines inception modules and residual connections. Besides, the network is fully convolutional without any max pooling and fully connected layers.

In 2015, researchers from Microsoft introduced a CNN architecture with residual connections termed as ResNet that won the 1st place in the ILSVRC image classification competition with 3.57% top-5 error. This network was built upon the philosophy of increasing depth of the network instead of widening, through residual connections to provide a better data representation. The ResNet architecture negates the vanishing gradient issue raises in deep networks by carrying important information in the previous layer to the next layer. Although such connection seems like an addition to the traditional CNN approach, it alleviates the training of the network and reduces number of parameters [56]. An illustration for the ResNet connection is given in Fig. 8.1a, where $X$ is input feature, $H(X)$ is any desired mapping, and $F(X)$ is a residual mapping. In [56], the residual feature fusion operation $H(X) = F(X) + X$ is performed by a shortcut connection and element-wise addition. Contrastingly, our model stacks the futures depthwise as $H(X) = F(X) \otimes X$, like shown in Fig. 8.1b, where $\otimes$ denotes feature-map concatenation. This favors to have less number of filters in convolutional layers at the same time to carry forward earlier layer’s features intact.
The **inception** module was a micro-architecture first introduced in [140] by Szegedy *et al.*, following the success of ResNet [141], [27]. The module acts as computation of multiple filters with different scales on the same input as in Fig. 8.1c. Also, it performs average pooling at the same time. Where, all the branches maintain the same spatial dimension of the previous layer’s output by using stride of 1 (S1) in all the operations. Finally, all the outcomes are aggregated along the channel dimension that to take advantage of multi-level feature representation, resulting in a higher discriminatory encoding. However, our module allows us to striding rate > 1 in each branch and performs the feature concatenation whenever the spatial dimension of the branches are matching as an example shown in Fig. 8.1d. Thus, the proposed model has Inception-like feature fusion at different stages of down-sampling and up-sampling sub-networks as depicted in Fig. 8.2.

Figure 8.1: The ResNet-like and Inception-like modules.

The rest of this chapter organization as follows. Section 8.3 describes the proposed model. Sections 8.4 presents details of the experimental setup and results along with discussion on existing FG detection algorithms. Finally, Section 8.5 concludes the chapter with future directions.
8.3 Proposed MV-FCN architecture

Figure 8.2: Layer schematic of the MV-FCN: Conv, Si, CTransk, Concat, and BN stand for convolution using kernel size of \( k \) and stride of \( i \), transpose convolution with filter size of \( k \), activation maps concatenation, and batch normalization operations, respectively.

Figure 8.2 abstracts away details of the proposed MV-FCN with a schematic. The MV-FCN integrates two complementary feature flows (CFF) and a pivotal feature flow (PFF). The PFF is essentially an encoder-decoder CNN while CFF1 and CFF2 complement PFF’s learning ability. The PFF only uses convolution kernels size of \( 3 \times 3 \), while CFF1 and CFF2 utilize filters size of \( 5 \times 5 \) and \( 9 \times 9 \) respectively in their first conv layers. However, after their first sub-sampling convolutional operations they use filter size of \( 3 \times 3 \) in their subsequent layers, so their output activation maps match a middle layer in the PFF for a successful feature-level augmentation. Thus, the features learned in the complementary layers are merged with appropriate intermediate feature maps in the PFF through residual connections. Using such heterogeneous convolutional kernels captures information available from different scales and provides both local and global context [9] and the fusion of feature maps from encoding layers that hold high-frequency detail resulting to sharper foreground boundaries.

In the encoding phase of PFF, four convolutional layers are networked sequentially after the very first conv layer that generates 16 channels with spatial dimension same as the input. Each of the four conv layers performs spatial subsampling by using a kernel size \( 3 \times 3 \) with stride of 2 such that the encoding process outputs activation map with a dimension of \( 15 \times 20 \times 96 \). The decoding phase, on the other hand, employs
<table>
<thead>
<tr>
<th>Layer ID</th>
<th>Layer type ((k, s))</th>
<th>Output Shape</th>
<th>Input (layer ID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input Layer</td>
<td>((b, 240, 320, 3))</td>
<td>mini-batch</td>
</tr>
<tr>
<td>2</td>
<td>Conv2D ((3, 1))</td>
<td>((b, 240, 320, 16))</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Conv2D ((5, 1))</td>
<td>((b, 240, 320, 16))</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Conv2D ((9, 1))</td>
<td>((b, 240, 320, 16))</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 120, 160, 16))</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 60, 80, 32))</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Conv2D ((5, 4))</td>
<td>((b, 60, 80, 32))</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>Concatenation</td>
<td>((b, 60, 80, 64))</td>
<td>6, 7</td>
</tr>
<tr>
<td>9</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 30, 40, 32))</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 30, 40, 32))</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>Conv2D ((9, 8))</td>
<td>((b, 30, 40, 32))</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>Concatenation</td>
<td>((b, 30, 40, 96))</td>
<td>9, 10, 11</td>
</tr>
<tr>
<td>13</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 15, 20, 32))</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>Conv2D ((5, 4))</td>
<td>((b, 15, 20, 32))</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 15, 20, 32))</td>
<td>11</td>
</tr>
<tr>
<td>16</td>
<td>Concatenation</td>
<td>((b, 15, 20, 96))</td>
<td>13, 14, 15</td>
</tr>
<tr>
<td>17</td>
<td>Conv2D ((3, 1))</td>
<td>((b, 15, 20, 64))</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 30, 40, 64))</td>
<td>17</td>
</tr>
<tr>
<td>19</td>
<td>Concatenation</td>
<td>((b, 30, 40, 160))</td>
<td>18, 12</td>
</tr>
<tr>
<td>20</td>
<td>Conv2D ((3, 1))</td>
<td>((b, 30, 40, 32))</td>
<td>19</td>
</tr>
<tr>
<td>21</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 60, 80, 32))</td>
<td>20</td>
</tr>
<tr>
<td>22</td>
<td>Concatenation</td>
<td>((b, 60, 80, 96))</td>
<td>21, 8</td>
</tr>
<tr>
<td>23</td>
<td>Conv2D ((3, 1))</td>
<td>((b, 60, 80, 32))</td>
<td>22</td>
</tr>
<tr>
<td>24</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 120, 160, 16))</td>
<td>23</td>
</tr>
<tr>
<td>25</td>
<td>Concatenation</td>
<td>((b, 120, 160, 32))</td>
<td>24, 5</td>
</tr>
<tr>
<td>26</td>
<td>Conv2D ((3, 1))</td>
<td>((b, 120, 160, 32))</td>
<td>25</td>
</tr>
<tr>
<td>27</td>
<td>Conv2D ((3, 2))</td>
<td>((b, 240, 320, 64))</td>
<td>26</td>
</tr>
<tr>
<td>28</td>
<td>Concatenation</td>
<td>((b, 240, 320, 112))</td>
<td>27, 2, 3, 4</td>
</tr>
<tr>
<td>29</td>
<td>BatchNorm.</td>
<td>((b, 240, 320, 112))</td>
<td>28</td>
</tr>
<tr>
<td>30</td>
<td>Conv2D ((3, 1))</td>
<td>((b, 240, 320, 128))</td>
<td>29</td>
</tr>
<tr>
<td>31</td>
<td>Dropout</td>
<td>((b, 240, 320, 128))</td>
<td>30</td>
</tr>
<tr>
<td>32</td>
<td>Conv2D ((3, 1))</td>
<td>((b, 240, 320, 1))</td>
<td>25</td>
</tr>
</tbody>
</table>

Total number of trainable parameters: 494,337

\(k\) - kernel size, \(s\) - stride rate, \(b\) - mini-batch size

Table 8.1: Layer detail of the MV-FCN.

Four transpose convolutional layers interspersed with residual feature concatenations and regular conv layers. Consequently, the decoding stage ends up with an inception module (layer 28 in Table 8.1) that merges the first stage activations from PFF, CFF1, and CFF2 with the final stage decoding activations resulting to a feature map of 240 \(\times\) 320 \(\times\) 112.

The extracted features from various conv layers in the encoding path are also combined with spatially matching up-sampled feature maps in the decoding path systematically. As stated earlier in the previous sections and chapters, this strategy
is an elegant solution for the lost of spatial resolution due to series of subsampling and convolutional operations carried out over the encoding process \[119\]. Hence, all the convolutions are immediately followed by ReLU activation functions, except the transpose convolution (it is generally referred as deconvolution) and the final layer. Top classification layers consist of a batch-normalization, conv with 128 channels followed by drop out of 0.3, and finally a single channel output conv with Sigmoid activation function. Table 8.1 summarizes the network detail, where conv2D and conv2DT denote 2D convolution and its transpose, respectively. The integers in the parentheses in layer type refer the kernel size and stride rate in the order while the \(b\) in output shape refers the mini-batch size. In total, the proposed model takes 494,337 trainable parameters.

In summary, the MV-FCN does not employ max pooling or hidden fully connected (FC) layers, but subsumes convolutional (conv), transpose convolutional (CTrans), and symmetric expanding paths with inception and residual connections to capture contextual information for an accurate FG inferencing. The network is capable of taking any spatial dimensions of input images and resize them into \(240 \times 320\) by using nearest-neighbor scaling algorithm to match with the input layer dimension. The convolutional layers use stride rate of 1 in all directions, except the sub-sampling layers that perform convolution with a stride rate of \(k - 1\), where \(K\) is the kernel size.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Frame size ((W \times H))</th>
<th>Nature</th>
<th>(N) frames</th>
<th>(N) frames for BG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>320 \times 240</td>
<td>Baseline</td>
<td>1229</td>
<td>520</td>
</tr>
<tr>
<td>Office</td>
<td>360 \times 240</td>
<td></td>
<td>1447</td>
<td>580</td>
</tr>
<tr>
<td>Canoe</td>
<td>320 \times 240</td>
<td>Dynamic background</td>
<td>342</td>
<td>840</td>
</tr>
<tr>
<td>Boats</td>
<td>320 \times 240</td>
<td>Dynamic background</td>
<td>6026</td>
<td>1930</td>
</tr>
<tr>
<td>Overpass</td>
<td>320 \times 240</td>
<td></td>
<td>440</td>
<td>2330</td>
</tr>
<tr>
<td>Traffic</td>
<td>320 \times 240</td>
<td>Camera jitter</td>
<td>609</td>
<td>900</td>
</tr>
<tr>
<td>Boulevard</td>
<td>320 \times 240</td>
<td></td>
<td>1004</td>
<td>790</td>
</tr>
<tr>
<td>CopyMachine</td>
<td>720 \times 480</td>
<td>Shadow</td>
<td>1401</td>
<td>495</td>
</tr>
<tr>
<td>PeopleInShade</td>
<td>380 \times 244</td>
<td>Shadow</td>
<td>829</td>
<td>280</td>
</tr>
<tr>
<td>TwoPositionPTZCam</td>
<td>570 \times 340</td>
<td>PTZ camera</td>
<td>449</td>
<td>750</td>
</tr>
<tr>
<td>Turnpike_0.5fps</td>
<td>320 \times 240</td>
<td>Low Framerate</td>
<td>350</td>
<td>750</td>
</tr>
</tbody>
</table>

Table 8.2: Dataset summary.

\(a\) Number of frames considered with ground truths, in which both the FG and BG are presented in the same frame.
8.3.1 Training strategy

8.3.1.1 Exclusive sets

We target the widely used benchmark database the change detection 2014 [158]. Table 8.2 briefs the properties of the datasets. To form exclusive sets of training and test data, the available samples are divided in sequence order, whereby training set takes 70% while test set takes 30% of the total number of samples that have ground truths with FG and BG information in a particular dataset. This way of data splitting is more appropriate rather than a random selection since the images in the datasets are frames from video sequences. Because, a random choice of samples may pick a frame for training set while picking a temporally closest frame, like frame_{t+1} or frame_{t-1} for test set. There can be many such instances in random selection resulting in mere exclusiveness of training and test sets. Figure 8.3 demonstrates the data split used in this work, in which n is the total number of samples that has ground truths in the sequence and k = \lfloor n \times 0.7 \rfloor that is the dividing point (frame no.) for the ordered split.

8.3.1.2 Input configuration

Experiments are carried out with two configurations: Single frame-based and double frame-based with a generic BG model. In the single-frame setting, we employ data augmentation by applying random transformations with rotation within 10 degrees, translation vertically and horizontally with a fraction of 0.1 from the total height and width, and zooming in range of 0.1 inside image samples. These data augmentations
are done on training images and the corresponding ground truths during training. Naturally, this allows the network to learn invariant to such transformations, without a need to see these variant samples in the annotated benchmark datasets. The double-frame setting is exactly the same manner as shown in Chapter 7 Fig. 7.5.

8.3.1.3 Optimizer

The MV-FCN is trained by using Adam-optimizer that minimizes binary cross-entropy loss defined by (8.1), where optimizer takes a base learning rate of 0.0002 with a learning rate scheduler that reduces the learning rate by factor of 0.8 over the training.

\[ E = -\frac{1}{n} \sum_{n=1}^{N} [p_n \log \hat{p}_n + (1 - p_n) \log(1 - \hat{p}_n)], \quad (8.1) \]

where it takes two inputs; first one is the output from the final layer of the network (layer 32 in Table 8.1) with dimension of \( N \times C \times H \times W \), which maps the FG pixel probabilities \( \hat{p}_n = \sigma(x_n) \in [0, 1] \) using Sigmoid non-linearity function \( \sigma(.) \) defined earlier in Eqn. 7.5. And the second one is target \( p_n \in [0, 1] \) with the same dimension as the first one, where \( N, C, H, \) and \( W \) represent the batch size, the number of channels, height, and width respectively of the inputs. In this case, \( p_n \) is the ground truth segmentation images whose pixel values are normalized. The network is trained on each video sequence separately.

8.3.1.4 Transfer learning

<table>
<thead>
<tr>
<th>Model Fine-tuned for</th>
<th>Model Transferred from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>Turnpike_0_5fps</td>
</tr>
<tr>
<td>Office</td>
<td>CopyMachine</td>
</tr>
<tr>
<td>Canoe</td>
<td>Boats</td>
</tr>
<tr>
<td>Boats</td>
<td>Canoe</td>
</tr>
<tr>
<td>Overpass</td>
<td>Pedestrians</td>
</tr>
<tr>
<td>Traffic</td>
<td>Highway</td>
</tr>
<tr>
<td>Boulevard</td>
<td>TwoPositionPTZCam</td>
</tr>
<tr>
<td>CopyMachine</td>
<td>Office</td>
</tr>
<tr>
<td>PeopleInShade</td>
<td>Pedestrians</td>
</tr>
<tr>
<td>TwoPositionPTZCam</td>
<td>Turnpike_0_5fps</td>
</tr>
<tr>
<td>Turnpike_0_5fps</td>
<td>TwoPositionPTZCam</td>
</tr>
</tbody>
</table>

Table 8.3: Transfer learning detail for fine-tuning the proposed MV-FCN.

To improve the network’s learning experience we incorporate intraclass domain transfer. Table 8.3 lists the fine-tuning dataset pairs. For instance, the pre-trained
network with TwoPositionPTZCam is fine-tuned for Turnpike_0.5fps. Here, both the domain have moving vehicles as FG objects. The theoretical and philosophical expositions of transfer learning can be found in [162] and [107].

8.3.1.5 Training environment:

Python with Keras libraries (Tensorflow backend) is used as a software platform for the implementation of the model. The network is then mainly trained on a GeForce GTX 1060-6 GB GPU with Intel(R) Core(TM) i7-4770 CPU @ 3.40 GHz and 32 GB memory (RAM). In average, the training takes about 2 hours on the GPU for each dataset when batch size is 8 and maximum of 30 epochs. The testing is carried out on another GPU - GeForce GTX 1080 Ti-12 GB and it takes about average of 22 ms per sample, i.e., it achieves about 45 fps.

8.3.2 Binary foreground mask

![Binary foreground mask](image)

Figure 8.4: Creating FG mask: Applying an appropriate threshold to the score-map generated from the last classification layer of MV-FCN for a frame taken from the Office dataset.

It is also crucial to create a binary mask that localizes the interested FG region. We apply a threshold to the score-map generated by the trained MV-FCN at frame-level to form a binary FG mask, like shown in Fig. 8.4 where the threshold \( \tau \) is a dataset-specific global parameter set empirically in the range \([0.05, 0.75]\). Then to clean noisy artifacts, we post process the binary image through neighborhood pixel connectivity.
that removes regions with less than 50 pixels. In another approach, We employ the Otsu’s clustering-based model to choose appropriate threshold automatically, since the score-map is a representation of bi-modal image. Otsu’s algorithm iteratively finds a threshold \( \tau \) that lies in between two peaks of the intensity histogram such that the intra-class variances of FG and BG classes are minimum. There, the weighted sum of within-class variances is defined as:

\[
\sigma^2_{\rho}(\tau) = \rho_0(\tau)\sigma^2_0(\tau) + \rho_1(\tau)\sigma^2_1(\tau),
\]

where the weights \( \rho_0 \) and \( \rho_1 \) are the probabilities of BG and FG classes clustered by a threshold \( \tau \), and the variances of these two classes are \( \sigma^2_0 \) and \( \sigma^2_1 \) respectively. An explicit derivation of the method can be found in [104]. An extended derivation is given in Chapter 7. Note that the binarization process is not part of the MV-FCN training procedure, but exclusive for testing stage as the numerical analysis is made on the binary masks.

### 8.4 Experimental setup, results, and discussion

To provide a better understanding of the model’s performance, we select eleven various sequences from the change detection database [158]. The video sequences consist of diversified change and motion, including benchmarks of baseline, dynamic background, camera jitter, shadow, videos shot with PTZ camera, and low frame-rate. A succinct description of the datasets is given in Table 8.2. Hence, the general nature of the datasets as follows.

The **baseline** benchmark represents a mixture of mild challenges, like subtle background motion, isolated shadows, swaying tree branches, and natural illumination changes.

The **dynamic background** category includes scenes with strong (parasitic) BG motion: boats and canoes on shimmering water, or a man walking on a shore of a shimmering water body.

The **camera jitter** datasets contain outdoor videos captured by vibrating cameras due to high wind and unstable mount. The jitter magnitude varies from one video to another.

The **shadow** category comprises indoor video exhibiting strong as well as faint shadows. Here, some shadows are cast by moving objects.

Lastly, in **PTZ camera recordings**, adjustments in camera strongly changes the backgrounds of a recorded video. Such conditions break the assumption of traditional
BG modeling algorithms assume that the recording devices are relatively static or move slowly, and thus it challenges the most algorithms. Note that this category of video sequences are not suitable for the double frame-based experiments since the viewpoint of the sequence is changed time to time as the cameras span around. So the motion detail captured by taking two consecutive frames along with a generalized BG model will be entirely different when the camera pans from one point to another. Thus, this category of video sequences is omitted for the double frame-based FGL.

### 8.4.1 Step-by-step analysis

#### 8.4.1.1 Impact of complimentary feature flows

It is always a good practice to carry out a sanity check to quickly evaluate whether a claim or the result of a calculation can possibly be true. Thus, to validate our approach of multi-view receptive field we conduct experiments with different combination of complimentary feature flows as described in Fig. 8.5. The Office video sequence from the baseline dataset and the Traffic sequence from the camera-jitter category are chosen for the experiments. The results are compared in Table 8.4a and Fig. 8.4b.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Office</td>
</tr>
<tr>
<td>A</td>
<td>87.99</td>
</tr>
<tr>
<td>B</td>
<td>89.02</td>
</tr>
<tr>
<td>C</td>
<td>90.06</td>
</tr>
<tr>
<td>D</td>
<td>91.27</td>
</tr>
</tbody>
</table>

(a) Results: impact of complimentary feature flows.

Table 8.4: Results: impact of complimentary feature flows.
(a) Experiment A: Pivotal feature flow.

(b) Experiment B: Combined feature flow of PFF & CFF2.

(c) Experiment C: Combined feature flow of PFF & CFF1.

(d) Experiment D: Combined feature flow of PFF, CFF1 & CFF2.

Figure 8.5: Network configurations for investigating impact of complimentary feature flows.
A frame from Traffic video sequence and its corresponding FG salient map generated by different feature flows are shown in Fig. 8.6. It shows that when all the feature flows are combined differentiation between background and foreground is very strong. On the other hand, when they are not combined we can see that there is a notable fussiness in the salient map that will lead a misleading segmentation of the foreground. It is evident in the f-measure tabulated in Table 8.4a and in Fig. 8.4b.

### 8.4.1.2 Impact of transfer learning

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MV-FCN’s performance with:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G-th(S)</td>
</tr>
<tr>
<td>Highway</td>
<td>0.9207</td>
</tr>
<tr>
<td>Office</td>
<td>0.9127</td>
</tr>
<tr>
<td>Canoe</td>
<td>0.8492</td>
</tr>
<tr>
<td>Boats</td>
<td>0.8493</td>
</tr>
<tr>
<td>Overpass</td>
<td>0.8733</td>
</tr>
<tr>
<td>Traffic</td>
<td>0.8488</td>
</tr>
<tr>
<td>Boulevard</td>
<td>0.7565</td>
</tr>
<tr>
<td>CopyMachine</td>
<td>0.9212</td>
</tr>
<tr>
<td>PeopleInShade</td>
<td>0.9163</td>
</tr>
<tr>
<td>TwoPos.PTZCam.</td>
<td>0.7953</td>
</tr>
<tr>
<td>Turnpike_0_5fps</td>
<td>0.8225</td>
</tr>
<tr>
<td><strong>Over all average</strong></td>
<td><strong>0.8605</strong></td>
</tr>
</tbody>
</table>

Table 8.5: Performance comparison of random vs transfer learning-based model initialization in terms of f-measure: S and P stand for type of training strategy, scratch and fine tuning pre-trained model. Global and Otsu threshold methods are referred by G-th and O-th respectively.

To validate the effectiveness of model initialization using transfer learning-based approach discussed in Section 8.3.1.4, experiments are conducted with single-frame input configuration and the outcome in terms of f-measure is tabulated in Table 8.5.
and Fig. 8.7. It is found that fine tuning the model with pre-trained weights provides consistent performance with overall average improvement of $\sim 4 - 6\%$.

![Graph showing performance vs model initialization in terms of f-measure.](image)

**Sequence Category**

Figure 8.7: Performance vs model initialization in terms of f-measure: S and P stand for type of training strategy, scratch and fine tuning pre-trained model. Global and Otsu threshold methods are referred by G-th and O-th respectively.

![Salient maps of MV-FCN when trained from scratch and fine-tuned.](image)

Figure 8.8: MV-FCN salient map when: (b) trained from scratch, (c) fine-tuned with intra-class transfer learning.

The impact of transfer learning is visualized using one sample from the Office dataset in Fig. 8.8. Respect to the salient maps (i.e., the probability score maps),
we can see that when the network is fine-tuned with pre-trained weights, the model generates the probability scores that have near-zero fussiness between FG and BG. It is apparently evident as the intensity histogram of the salient map produced by the network trained from scratch has multiple peaks, while the intensity histogram of the salient map has well distinguished two peaks; one belongs to FG, and the other belongs to BG. Thus, the transfer learning-based parameter initialization allows the network to produce stronger discrimination of FG regions from BG as the distribution of probability falls around two distinct peaks, generally with the intensity values of 0 (dark as BG) and 255 (bright as FG). Taking the above outcomes as an empirical proof for the multi-view receptive field fully convolutional network (MV-FCN), experiments are carried out on all the datasets listed in Table 8.2 with the two input conditions stated earlier: the single frame-based and the double frame-based with a temporally median filtered BG model.

8.4.1.3 Qualitative analysis

For a visual inspection, three sample results from the single frame-based foreground localization\textsuperscript{1} from a selected video sequence per category from Table 8.2 are shown in Fig. 8.9 - Fig. 8.14. These visual comparisons against the respective ground truths show that the proposed MV-FCN localize the FG objects very closely; however, it has to be quantitatively analyzed for further validation. The following subsection 8.4.1.4 provides the numerical analysis in terms of f-measure.

Figure 8.9: Office dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: training and validation FoM and loss respectively in the top and bottom.

\textsuperscript{1} Rest of the results will be available in the project web page.
8.4.1.4 Quantitative analysis

In FG localization, the standard performance measure used is f-measure or FoM. The FoM measures the similarity between the predicted FG and the ground-truth, and it is defined as a weighted harmonic mean measure of recall and precision, i.e., a region of intersection divided by the union of predicted and actual FG regions. It is also referred as intersection-over-union (IoU) as in (8.3).

\[
FoM = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}
\]  

\[
= \frac{2 \times TP^2}{TP(TP+FP+TP+TN)} = \frac{2 \times TP}{(TP+FP)+(TP+FN)}
\]
Figure 8.12: PeopleInShade dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: Training and validation FoM and loss respectively in the top and bottom.

Figure 8.13: TwoPositionPTZCam dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: Training and validation FoM and loss respectively in the top and bottom.

where recall is the detection rate defined by $TP/(TP + FN)$ and precision is the percentage of correct prediction compared to the total number of detections as positives, given by $TP/(TP + FP)$, where $TP, FN,$ and $FP$ refer true positive, false negative, and false positive respectively. For a given output $X$ from the proposed MV-FCN, i.e., the probabilities over a set of pixels $V = \{1, 2, \cdots , N\}$ in the input image, and $Y \in \{0, 1\}$ the ground-truth assignment for the set $V$, where 0 and 1 refer the BG and FG object pixels respectively, then (8.3) can be formalized as (8.4).

$$FoM = \frac{2 \times I(X)}{U(X)},$$

(8.4)
Figure 8.14: Turnpike_0.5fps dataset. Col. 1-5: Sample input frames, MV-FCN generated score-maps, binary FG masks with empirical and Otsu’s thresholds. Col. 6: Training and validation FoM and loss respectively in the top and bottom.

where $I(X)$ and $U(X)$ can be approximated as follows:

$$I(X) = \sum_{v \in V} X_v \ast Y_v + \epsilon \equiv TP,$$  \hfill (8.5)

$$U(X) = \sum_{v \in V} (X_v + Y_v) + \epsilon \equiv (TP + FP) + (TP + FN),$$  \hfill (8.6)

where $\epsilon$ is a very small value set to $1e-08$.

The Table 8.6 quantitatively compares the performance of our MV-FCN with some of the results recorded in the literature from prior-art and state-of-the-art techniques. These methods include the probabilistic-based approaches as well as neural network (NN)-based learning algorithms in recent years. Figure 8.15 on the other hand, summarizes the results, where SFO-th, SFG-th, DFO-th, DFG-th, BoU, and BoO refer single frame-based model training with Otsu and global threshold, double-frame together with generalized BG as input channels with Otsu and global threshold, best results of our method, and best results of other methods stated in the Table 8.6 respectively. In overall average performance across all the experiments, it is found that using two frames along with a generalized BG the model improves $\sim 6.5\%$ and $\sim 6.0\%$ when global and Otsu threshold are applied, respectively. At the same time, considering the best performances across all the datasets, like shown in Fig. 8.15 and 8.16 our model gains $\sim 8.75\%$ improvements compared to prior- and state-of-the-art results.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>G-th</td>
<td>O-th</td>
<td>G-th</td>
<td>O-th</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Highway</td>
<td></td>
<td>0.9264</td>
<td>0.8790</td>
<td>0.9685</td>
<td>0.9675</td>
<td>0.9330</td>
<td>52</td>
</tr>
<tr>
<td>Office</td>
<td></td>
<td>0.9610</td>
<td>0.9175</td>
<td>0.9708</td>
<td>0.9684</td>
<td>0.5864</td>
<td>149</td>
</tr>
<tr>
<td>Canoe</td>
<td></td>
<td>0.9404</td>
<td>0.9315</td>
<td>0.9563</td>
<td>0.9526</td>
<td>0.7923</td>
<td>137</td>
</tr>
<tr>
<td>Boats</td>
<td></td>
<td>0.8727</td>
<td>0.8600</td>
<td>0.9168</td>
<td>0.8735</td>
<td>0.8324</td>
<td>149</td>
</tr>
<tr>
<td>Overpass</td>
<td></td>
<td>0.8825</td>
<td>0.8707</td>
<td>0.9502</td>
<td>0.9442</td>
<td>0.6924</td>
<td>48</td>
</tr>
<tr>
<td>Traffic</td>
<td></td>
<td>0.8563</td>
<td>0.8054</td>
<td>0.8679</td>
<td>0.8109</td>
<td>0.8302</td>
<td>12</td>
</tr>
<tr>
<td>Boulevard</td>
<td></td>
<td>0.8737</td>
<td>0.8116</td>
<td>0.9413</td>
<td>0.9396</td>
<td>0.8174</td>
<td>52</td>
</tr>
<tr>
<td>CopyMachine</td>
<td></td>
<td>0.9443</td>
<td>0.9349</td>
<td>0.9525</td>
<td>0.9509</td>
<td>0.9289</td>
<td>137</td>
</tr>
<tr>
<td>PeopleInShade</td>
<td></td>
<td>0.9532</td>
<td>0.9396</td>
<td>0.9628</td>
<td>0.9401</td>
<td>0.8986</td>
<td>137</td>
</tr>
<tr>
<td>TwoPos. PTZCam.</td>
<td>0.8411</td>
<td>0.8326</td>
<td>NA</td>
<td>NA</td>
<td>0.8285</td>
<td>137</td>
<td>0.8315</td>
</tr>
<tr>
<td>Turnpike_0.5fps</td>
<td>0.8946</td>
<td>0.9100</td>
<td>NA</td>
<td>NA</td>
<td>0.8967</td>
<td>137</td>
<td>0.9100</td>
</tr>
</tbody>
</table>

Table 8.6: F-measure performance comparison: S- training from scratch, P- pre-trained model fine-tuning, Global and Otsu stand for the two used thresholding methods. Values in red are the best figures while the ones in blue are the second best.
In general, most of the state-of-the-art methodologies use patch-wise processing and multimodal-based algorithms for BG establishment and/or a feedback-based approach as post-processing to refine the primarily detected FG regions. Such setup
ensues complex computations and higher processing time due to the time-consuming iterative pursuit of the low-rank matrix or sparse matrix. On the contrary, the proposed model processes the whole input image as a single entity during inferencing. Then it refines the output by a none iterative post-processing, resulting $\sim 22 ms$ (mean average processing time) per frame, i.e., $\sim 45$ FPS (see Fig. 8.17) on GTX 1080 Ti GPU for FG prediction once the network is trained.

![Inferencing speed of the proposed MV-FCN.](image)

**Figure 8.17**: Inferencing speed of the proposed MV-FCN.

8.5 Conclusion

This chapter put an NN forwards for foreground localization that is inspired by recent innovations in deep learning, such as, ResNet, Inception modules, and Fully convolutional network. The proposed model utilizes a heterogeneous set of convolutions to capture invariant features at different scales.

In the traditional approaches, much time has been spent on sophisticated mathematical modeling to optimize background generation and post-processing to get a few more valid foreground pixels. Besides that, feature engineering and manual parameter tuning of traditional methods become unneeded since the network parameters can be learned from exemplar FG segmentation ground truths during training. For these
reasons, we advocate that the proposed multi-view receptive field FCN is a novel addition to neural-based FG localization systems.

Extensive experiments are conducted to analyze the proposed model’s performance under various conditions:

i. Training with random-state model initialization.

ii. Fine tuning model with transferred pre-trained model parameters.

iii. Input configuration with single frame in RGB color space.

iv. Input configuration that takes gray scale two consecutive frames and a temporarily median filtered BG model.

The qualitative and quantitative performance evaluations of the proposed MV-FCN on various challenging video sequences collected from benchmark datasets demonstrate that the model performs better than or very competitively to the prior- and state-of-the-art methods. However, the limitation of the network comes with a high number of trainable parameters. We leave this for our future direction, where we plan to optimize the network to achieve better results with fewer number parameters. In application point of view, the MV-FCN can be exploited for many other applications, like MRI slice partitioning and path segmentation for autonomous vehicles. Finally, it must be considered that a perfect FG prediction is still an open and intriguing task and a good FG detection system should use the knowledge derived from its ultimate purpose.
Chapter 9

A 3D CNN-LSTM Based Image-to-Image Foreground Segmentation

9.1 Summary

Foreground (FG) segmentation has been widely studied due to its vital role in many applications, including intelligent transportation and video surveillance. Most of the existing algorithms are based on traditional CV techniques that perform pixel-level processing assuming that FG and Background (BG) possess distinct visual characteristics. Recently, the state-of-the-art solutions exploit Deep Learning (DL) models targeted initially for image classification. The major drawbacks of such strategy are the lacking delineation of FG regions and missing temporal information as they achieve FG segmentation based on single frame object detection. To grapple with this issue, we excogitate a 3D Convolutional Neural Network (CNN) with Long Short-Term Memory (LSTM) pipelines that harness seminal ideas, viz. Fully Convolutional Networking (FCN), 3D transpose convolution, and Residual Network (ResNet) connections.

Thence, a FG segmenter is implemented in an Encoder-Decoder (EnDec) fashion and trained on representative FG segments. The model is a Vanilla structure, which fuses the encoded spatiotemporal features with appropriate decoded feature maps for achieving enhanced representation. Finally, the FG is separated by using Nobuyuki Otsu’s method and an empirical global threshold. The analysis of experimental results via standard quantitative metrics on seven benchmark datasets including both indoor and outdoor scenes validates that the proposed 3D CNN-LSTM achieves competitive performance in terms of Figure of Merit (FOM) or F-measure evaluated against prior-
and state-of-the-art methods. We also compare the results with our another model discussed in Chapter 8, the Multi-view receptive field fully convolutional network (MV-FCN).

9.2 Introduction

These days, video-based intelligent systems have become ubiquitous due to a myriad of easily accessible low price camera devices. Such systems face a crucial challenge of processing massive volume of data from multiple feeds at the same time. It is also required for them to tackle with varying environmental factors, like illumination changes, dynamic backgrounds, and so forth [169]. These demands perplex the real-time operation of systems. In the analysis of traffic flow or human activity, the performance of an intelligent system substantially depends on its foreground segmentation robustness. An example of FG detection is shown in Fig. 9.1. Besides being a core unit of video analytic intelligent frameworks, the FG segmentation is also an integral part of various machine-vision problems, for instance, autonomous/intelligent driving [6, 170], object segmentation/retrieval [20, 74, 143], image quality assessment [28], visual tracking [184], and human-robot/machine interaction [46]. The primary objective of FG detection is to place a tight mask, where the appearance of an object, a vehicle or human is monitored. Such FG mask is very informative than bounding box as it allows close localization of objects. It can be achieved by employing several algorithms categorized into five groups: i) Sample-based [11, 62, 137, 144, 146], ii). Probabilistic-based [3, 34, 65, 139, 149, 158], iii). Subspace-based [10, 52, 102], iv). Codebook-based [138, 164, 176], and v). Neural network (NN)-based [8, 41, 122, 177, 181].

Figure 9.1: Traffic flow and its foreground (Input frame, Ground truth, 3D CNN-LSTM score-map, and Predicted FG mask).

The sample-based algorithms create a BG from the past set of $N$ frames, i.e., for each pixel location there are $N$ samples stored. If there are $k$ number of pixels in the BG that have a distance smaller than a threshold $\tau$ to the incoming pixel, then the
Figure 9.2: An overview of the proposed CNN-LSTM image-to-image network: $E(\cdot)$ - binary cross-entropy error.

A pixel is classified as FG. The **probabilistic models** work on the principle of stochastic process, like Gaussian mixture models (GMM) [3, 94] and Conditional Random Field (CRF)-based algorithms [187]. The **subspace-based** approaches perform a transformation of data to a subspace, such as Eigenspace or Principal Component Analysis (PCA)-based subspace. Then, they form a BG model using the subspace and estimate the FG. The **Codebook** generates a dictionary that consists of color, intensity, temporal features, or similar representations. Same properties of a new pixel are compared with the dictionary values to determine it’s status. The **NN-based** approaches are kind of models that generate a classifier through training to handle the segmentation task. The trained weights of a NN serve as BG model and can be updated to reflect the changes occurred in the scene. Here, a learning systems, which formulates FG segmentation as a structured input-output matching problem. Such models have gained their reputation after a breakthrough performances in the ImageNet-Large-Scale Visual Recognition Challenge (ILSVRC).

The NN-based techniques have been exploited for image semantics and labeling task [130, 178], medical image partitioning [109, 119], and recently for video segmentation [169], as well. The main challenges in CNN-based FG detection is that dealing with time-dependent motion and the dithering effect at bordering pixels of FG objects. We address these issues, by excogitating a 3D EnDec CNN that utilizes ResNet [56]-like residual connections for lost feature recovery and LSTM units to handle spatiotemporal motion of FG objects. To facilitate the training process, we take advantage of intra-domain transfer learning.

In summary, the key contribution of this chapter is a novel 3D deep FCN for FG segmentation that deals with the time-dependent video data via LSTM cells in an EnDec architecture. The proposed model harnesses seminal strategies, like 3D convolutions, residual connections, and 3D transpose convolutions introduced in
the recent literature for precise BG-FG representation. The rest of this chapter is
organized as follows: Section 9.3 elaborates the architectural information. Section 9.4
describes the experimental setup, analyses the performance, and highlights some key
characteristics of the compared existing methods. Finally, Section 9.5 concludes the
chapter with future directions.

9.3 The CNN-LSTM based foreground segmenter

This section provides a soup-to-nuts description of the proposed model. The proposed
3D CNN-LSTM model abstracted by Fig. 9.2 has the following variations from the a
basic image-to-image CNN like U-net discussed in Chapter 1:

i. The max-pooling operations achieve invariant features but has a toll on object
localization accuracy [21]. To circumvent this, we perform subsampling process
by zero padded conv with a kernel size of 3 and stride of 2.

ii. Our model entirely uses 3D conv layers embedded with LSTM modules for long-
short term temporal feature considerations both in encoding and decoding sub-
nets.

iii. The proposed model uses 3D convolutions in the encoding stage, so it employs
3D transpose conv in decoding path in contrast to 2D standard upsampling op-
erations.

The LSTM is an advanced version of Recurrent Neural Networks (RNNs) [59].
The Constant Error Carousel (CEC) cells in LSTM use an identity activation func-
tion and have self-routed connections to themselves with a constant weight of 1.0. So,
errors backpropagated through the LSTMs cannot explode or vanish [127]. It is con-
sidered to be biologically plausible structure, to a certain level and has been proved
to solve previously unlearnable DL tasks involving temporal data. There are many
variations of LSTMs, such as decoupled extended Kalman filter LSTM-RNN [110],
bi-directional [45], and Connectionist temporal classification (CTC) [44]. LSTM is
applicable to several real-world tasks, like handwriting recognition [111], speech/lan-
guage identification [45], robot control/localization [151], and driver distraction detec-
tion [163]. Thus, our model also harnesses the LSTM to capture temporal connections
between consecutive frames to detect FG objects.

A simplified version of the proposed model with single stage of encoding and
decoding is shown in Fig. 9.4. It has only one level of subsampling (convolutional
Figure 9.3: A layer-wise schematic of the proposed 3D CNN-LSTM. It exploits 3D conv embedded with 2D LSTM both in the encoding and decoding phases, and residual layer concatenation and 3D transpose conv layers only in the decoding subnetwork. However, the actual network has five levels of encoding and four levels of up-sampling stages. Where, encoding stage subsumes six layers while decoding stage consists of eighteen layers, including 3D-convT (4) residual feature concatenation (4), and final classification layers (BN, 3D-Conv). Thus, Fig. 9.3 overviews the entire 3D CNN-LSTM structure via a lucid schematic while layer configurations and connectivity pattern are tabulated in Table 9.1. The network integrates three major components: encoder, decoder, and classifier. It maintains a constant number of filters (16) at each layer (except the penultimate layer, that produces 20 feature maps) and the kernel size, $k = 3$. Hence, the spatial dimension of feature maps is linearly reduced by half through striding the kernel at a rate of 2 in the encoding phase. Thus, the last layer of the encoder generates feature maps that have spatial dimension of $15 \times 20$ as the network’s input layer accepts frames with spatial dimension of $240 \times 360$ (layer ID 1 - 6). To achieve precisely decoded feature maps there are four mini-decoder blocks sequentially networked (layer ID 7 - 22). Where, each block subsumes a 3D transpose conv, a 3D conv+2D LSTM, a concatenation, and again a 3D conv+2D LSTM layers connected serially. Hence, the final layer of the decoder produces feature maps with same spatial dimension as the network’s input. Every stage in the decoder receives residual cues from encoding stage via shortcuts as shown in Fig. 9.3. The final classifier module consists of a Batch Normalization (BN) and 3D conv layers with Sigmoid function as classifier. Thus, output of the 3D CNN-LSTM is the probability map of the current frame estimated based on the observed $n$ frames including the present one (layer ID 23 - 24). Except the last layer all other conv related layers employ ReLU activator. At this juncture, it is vital to discuss about the intricacies of the core components and the functions utilized in the proposed network. To this end, the following
Figure 9.4: Simplified 3D CNN-LSTM model with single stage EnDec.
<table>
<thead>
<tr>
<th>Layer ID</th>
<th>Layer type ((k, s))</th>
<th>Output Shape</th>
<th>Input (layer ID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input Layer</td>
<td>((b, n, 240, 320, 3))</td>
<td>mini-batch</td>
</tr>
<tr>
<td>2</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 240, 320, 16))</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Conv3D+LSTM2D ((3, 2))</td>
<td>((b, n, 120, 160, 16))</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Conv3D+LSTM2D ((3, 2))</td>
<td>((b, n, 60, 80, 16))</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Conv3D+LSTM2D ((3, 2))</td>
<td>((b, n, 30, 40, 16))</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Conv3D+LSTM2D ((3, 2))</td>
<td>((b, n, 15, 20, 16))</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Conv3D ((3, 2))</td>
<td>((b, n, 30, 40, 16))</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 30, 40, 16))</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>Concatenation</td>
<td>((b, n, 30, 40, 32))</td>
<td>5, 8</td>
</tr>
<tr>
<td>10</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 30, 40, 16))</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>Conv3DT ((3, 2))</td>
<td>((b, n, 60, 80, 16))</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 60, 80, 16))</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>Concatenation</td>
<td>((b, n, 60, 80, 32))</td>
<td>4, 12</td>
</tr>
<tr>
<td>14</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 60, 80, 16))</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>Conv3DT ((3, 2))</td>
<td>((b, n, 120, 160, 16))</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 120, 160, 16))</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>Concatenation</td>
<td>((b, n, 120, 160, 32))</td>
<td>3, 16</td>
</tr>
<tr>
<td>18</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 120, 160, 16))</td>
<td>17</td>
</tr>
<tr>
<td>19</td>
<td>Conv3DT ((3, 2))</td>
<td>((b, n, 240, 320, 16))</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 240, 320, 16))</td>
<td>19</td>
</tr>
<tr>
<td>21</td>
<td>Concatenation</td>
<td>((b, n, 240, 320, 32))</td>
<td>2, 20</td>
</tr>
<tr>
<td>22</td>
<td>Conv3D+LSTM2D ((3, 1))</td>
<td>((b, n, 240, 320, 20))</td>
<td>21</td>
</tr>
<tr>
<td>23</td>
<td>BatchNorm</td>
<td>((b, n, 240, 320, 20))</td>
<td>22</td>
</tr>
<tr>
<td>Top</td>
<td>Conv3D ((3, 1))</td>
<td>((b, n, 240, 320, 1))</td>
<td>23</td>
</tr>
</tbody>
</table>

Total number of trainable parameters \(298,529\)

\(k\) - kernel size, \(s\) - stride rate, \(b\) - mini-batch size,
\(n\) - number of frames taken for representing the temporal domain.

Table 9.1: Layer detail of the proposed 3D CNN-LSTM.

Subsections provide the fundamentals.

### 9.3.1 ConvLSTM layers

The 3D conv is pertinent to spatiotemporal representation learning. It performs convolutional operations spatiotemporally unlike 2D conv layer that does only spatially. Thus, a 3D conv extracts short-term temporal features resulting in an output volume that is received by the embedded LSTM units to retain long-term temporal connectivity cues between consecutive frames. The conv operation is determined by its filter weights that are updated through training. All the filter weights are fixed like a system memory; some literature refer it as anchor vectors since they serve as reference visual patterns in the testing phase.
Figure 9.5: A standard LSTM module with three gates.

An output feature map of a conv $C$ w.r.t. kernel $\omega$, bias $b$, and an input image/patch $x$ is computed as

$$C(m, n) = b + \sum_{k=0}^{K-1} \sum_{l=0}^{K-1} \omega(k, l) * x(m + k, n + l), \quad (9.1)$$

where $\ast$, $K$, $\{m, n\}$, and $\{k, l\}$ represent the conv operation, size of the kernel, first coordinate or origin of the image/patch, and element index of the kernel respectively. Hence, feature map dimension of the conv layer is given by $(I_s - K_s + 2 \times P)/S + 1$, where $I_s, K_s, P$, and $S$ denotes size of input image/path, filter size, number of zero-padded pixels, and stride rate respectively.

The conventional 1D LSTMs do not take spatial dependency into consideration; however, in this work, the 2D LSTMs cover the spatiotemporal relationships as they are integrated with 3D conv. Figure 9.5 describes a standard LSTM unit, where $X_1, \ldots, X_t$ are the inputs, $C_t$ is the cell state, $H_t$ is the hidden state, and $i_t, f_t, o_t$ are the gates of a ConvLSTM block. If '$\ast$' and 'o' denote the conv operator and Hadamard product, then computation of the ConvLSTM block can be derived as:
\[i_t = \sigma(W_{xi} \ast X_t + W_{hi} \ast H_{t-1} + b_i),\]  
\[(9.2)\]

\[f_t = \sigma(W_{xf} \ast X_t + W_{hf} \ast H_{t-1} + b_f),\]  
\[(9.3)\]

\[o_t = \sigma(W_{xo} \ast X_t + W_{ho} \ast H_{t-1} + b_o),\]  
\[(9.4)\]

\[C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} \ast X_t + W_{hc} \ast H_{t-1} + b_c),\]  
\[(9.5)\]

\[H_t = o_t \circ \tanh(C_t),\]  
\[(9.6)\]

where \(\sigma\) is the recurrent activator, \(W_{x}\) and \(W_{h}\) are the spatial dimension of conv kernels. In this case, \(\sigma\) is a hard sigmoid function.

### 9.3.2 Transpose convolution

The 3D transpose conv layers perform upsampling of 3D conv such that spatial dimension of the output feature maps become as twice as the input without losing the connectivity pattern. In contrast to spatial resizing (extrapolation), the transpose layer has trainable parameters. It is done by inserting zeros between consecutive neurons in the input receptive field, then sliding the conv kernel with unit strides \[33\].

### 9.3.3 Activation functions

This section outlines the activation functions used in the proposed model. Additional information on activation functions can be referred to Chapter 2, Section 2.10. The activation functions improve NN’s representation ability by introducing nonlinear factors, since the linear representation of conv operation faces its limits when it comes to deep architectures. The ReLU can be formally defined as (9.7) when taken a case with \(K\) number of anchor vectors, denoted by \(w_k \in \mathbb{R}^N, k = 1, 2, \ldots, K\). For a given input \(x\), the correlations with \(a_k\) and \(k = 1, 2, \ldots, K\), defines a nonlinear rectification to an output \(y = (y_1, \ldots, y_K)^T\), where

\[y_k(x, a_k) = \max(0, a_k^T x) \equiv ReLU(a_k^T x),\]  
\[(9.7)\]

i.e., it clips negative values to zero while keeping positives intact. The benefit of ReLU is sparsity, overcoming vanishing gradient issue, and efficient computation than other activations. Sigmoid, on the other hand, has output in the range \([0, 1]\) for an input \(x\) and it is defined by

\[f(x) = \frac{1}{1 + \exp(-x)}.\]  
\[(9.8)\]
It befits a binary classifier, as used in this work and linear regression problems. **Hard-Sigmoid** is a linear piece-wise function that approximates the outputs as a linear interpolation between pair of cut-points. It is computationally very fast [73].

**The batch normalization** operation can be mathematically formulated as follows. Let the output of a layer \( X \in \mathbb{R}^{N,D} \), where \( N \) is the number of samples in the mini-batch and \( D \) is the number of hidden neurons, then normalized matrix \( \hat{X} \) is given as

\[
\hat{X} = \frac{X - \mu_B}{\sqrt{\sigma^2_B + \epsilon}}, \tag{9.9}
\]

where \( \mu_B, \sigma^2_B \), and \( \epsilon \) refer to the mean and variance of the mini-batch, and a small value of 0.001 to prevent division by zero, respectively. Then, the layer maintains its representational strength by testing the identity transform as

\[
y = \gamma \hat{X} + \beta, \tag{9.10}
\]

where, \( \beta \) and \( \gamma \) are trainable parameters that are initialized with \( \beta = 0 \) and \( \gamma = 1 \), in this work. Note that, when \( \beta = \mu_B \) and \( \gamma = \sqrt{\sigma^2_B + \epsilon} \) it returns the previous layer’s activation map. Employing BN has multifaceted benefits, including training process with much higher learning rates without much attention to initialization [63].

### 9.3.4 Training strategy

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Frame size ((W \times H))</th>
<th>Nature</th>
<th>( N ) frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>320 × 240</td>
<td>Baseline</td>
<td>1229</td>
</tr>
<tr>
<td>Office</td>
<td>360 × 240</td>
<td></td>
<td>1447</td>
</tr>
<tr>
<td>Canoe</td>
<td>320 × 240</td>
<td>Dynamic background</td>
<td>342</td>
</tr>
<tr>
<td>Boats</td>
<td>320 × 240</td>
<td></td>
<td>6026</td>
</tr>
<tr>
<td>Overpass</td>
<td>320 × 240</td>
<td>Camera jitter</td>
<td>440</td>
</tr>
<tr>
<td>Boulevard</td>
<td>320 × 240</td>
<td>Shadow</td>
<td>1004</td>
</tr>
<tr>
<td>CopyMachine</td>
<td>720 × 480</td>
<td>Shadow</td>
<td>1401</td>
</tr>
</tbody>
</table>

**Table 9.2**: Dataset summary.

**Exclusive sets**: Experiments are carried on widely accepted video sequences from change detection 2014 [158] benchmark database. Table 9.2 briefs the properties of the datasets. To form exclusive sets of training and test, the available samples with ground truths are divided such a way the training set takes first 70% of frames and the
test set takes the rest. This approach is more appropriate than a random selection of frames used in [8] for video FG segmentation. Because, an arbitrary choice of samples may pick a frame, $I_t$ for training set while picking a temporally closest frame, like $I_{t+1}$ or $I_{t-1}$ for test set. There can be many such instances in random selection resulting in mere exclusiveness of training and test sets. Note that, in [169], 90% of the samples are selected for training and only 20 samples from the rest are considered for testing from each dataset. To meet the input layer requirement of the proposed model, the training and test datasets, have to be rearranged to form a 5D data sequence as shown in Fig. 9.6, where $t, k$ refer the number of frames taken to represent the temporal domain and total number of sequential samples in the particular dataset. Accordingly, the same arrangement is done for the corresponding ground truths as well.

**Training:** The 3D CNN-LSTM is trained specifically to each dataset with Adam-optimizer that minimizes binary cross-entropy loss defined by Eqn. (9.11), where the base learning is set to 0.0002 with a scheduler that reduces the learning rate by factor of 0.8.

$$E = \frac{-1}{n} \sum_{n=1}^{N} \left[ p_n \log \hat{p}_n + (1 - p_n) \log(1 - \hat{p}_n) \right],$$  \hspace{1cm} (9.11)$$

where it takes two inputs; first one is the output from the final layer of the network with dimension of $N \times t \times C \times H \times W$, which maps the FG pixel probabilities $\hat{p}_n = \sigma(x_n) \in [0, 1]$ using Sigmoid classifier, $\sigma(.)$ defined earlier in Eqn. 9.8. And the second one is target $p_n \in [0, 1]$ with the same dimension as the first one, where $N, t, C, H,$
and $W$ represent the batch size, the number of frames in temporal axis and channels, height, and width respectively. In this case, $p_n$ is the normalized segmentation ground truth images.

**Transfer learning:** To improve the network’s trainability in short-span of epochs, it is necessary to have proper weight initialization. It can be achieved by transfer learning, where the model learns new task efficiently by using already learned parameters or knowledge [2]. To this end, we incorporate intraclass domain transfer and fine-tuning following the dataset pairs given in Table 9.3. For instance, the model is pre-trained on *TwoPositionPTZCam* then fine-tuned for *Boulevard*.

<table>
<thead>
<tr>
<th>Model fine-tuned to</th>
<th>Model transferred from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>Turnpike.0.5fps</td>
</tr>
<tr>
<td>Office</td>
<td>CopyMachine</td>
</tr>
<tr>
<td>Canoe</td>
<td>Boats</td>
</tr>
<tr>
<td>Boats</td>
<td>Canoe</td>
</tr>
<tr>
<td>Overpass</td>
<td>Pedestrians</td>
</tr>
<tr>
<td>Boulevard</td>
<td>TwoPositionPTZCam</td>
</tr>
<tr>
<td>CopyMachine</td>
<td>Pedestrians</td>
</tr>
</tbody>
</table>

Table 9.3: The dataset pairs used for model fine-tuning.

**Environment:** Python with Keras (Tensorflow backend) is used as a software paradigm. The network is trained on a GTX 1080Ti 12GiB with Intel(R) Core(TM) i7-6850K CPU @ 3.60 GHz, 64 GiB memory, and Ubuntu 64-bit OS. In average, the training takes about 1.5 to 2 hours depends on dataset when batch size and max number of epochs are set to 8 and 30 respectively, and the testing takes 68.66 ms per sample, i.e., $\sim 15$ fps.

### 9.3.5 Binary foreground mask

It is also crucial to create a binary mask that segments FG region from BG. We apply an empirical dataset-specific global threshold value ([0.05, 0.75]) to transform score-maps generated during inferencing. Then to clean noisy artifacts, a neighborhood connectivity-based post-processing is carried out removing regions of 50 pixels or less. As the score-map represents a bi-modal grayscale image, we employ the Nobuyuki Otsu’s clustering algorithm with and without Gaussian smoothing to choose an appropriate threshold adaptively. Otsu iteratively computes a threshold value, $\tau$ that lies in-between two peaks of the intensity histogram of a bi-model image, whereby
intraclass variances are minimum \cite{104}. The weighted sum of within-class variance is defined as

\[ \sigma^2_{\rho}(\tau) = \rho_0(\tau)\sigma^2_0(\tau) + \rho_1(\tau)\sigma^2_1(\tau), \]  

(9.12)

where the weights \( \rho_0 \) and \( \rho_1 \) are the probabilities of BG and FG clustered by a threshold \( \tau \), and the variances of these two classes are \( \sigma^2_0 \) and \( \sigma^2_1 \) respectively. An explicit derivation of the Otsu method can be found in Chapter \cite{7} Section \cite{7.3.4} or the reader can refer to \cite{104}. This binarization process is part of testing stage only as the numerical analysis is made on the binary FG mask.

### 9.4 Experimental setup, results, and discussion

This section examines the proposed model through comparisons to existing methods, including classical approaches and recent NN-based ones. The highlights of the compared methods are also provided on-the-fly. Comparison is also made against our another network, the single frame-based MV-FCN in Chapter \cite{8}. The MV-FCN is also an EnDec-like architecture but differs from 3D CNN-LSTM in the following manners:

i. It uses 2D conv in place of 3D conv.

ii. It does not utilize LSTM layers since it just considers the current frame itself for predicting FG, like an object detector.

iii. It uses three different kernels that convolve with the same input and generate three feature flow paths. The generated feature maps are, then, merged along the EnDec path.

Layer detail and connectivity pattern of the MV-FCN are given in Table \cite{8.1}. The model is evaluated on seven video sequences from the benchmark change detection database \cite{158} that consists of both indoor and outdoor scenes. A succinct description of the datasets is given in Table \cite{9.2}. General nature of the datasets as follows: the baseline represents a mixture of mild challenges, like subtle background motion, isolated shadows, swaying tree branches, and natural illumination changes; the dynamic background includes scenes with strong (parasitic) BG motion, and shimmering water; the camera jitter contains outdoor videos captured by vibrating cameras due to high wind; and the shadow category comprises indoor video exhibiting strong as well as faint shadows, where the shadows are even cast by moving objects.
9.4.1 Qualitative analysis

(a) Sample results of the Highway dataset. Col. 1-6: input frames, ground truths, 3D CNN-LSTM score-maps, binary FG masks with empirical, Otsu and Gaussian smoothing + Otsu thresholds.

(b) F-measure on Highway dataset.

Figure 9.7: Qualitative analysis on Highway video sequence.

(a) Sample results of the Office dataset. Col. 1-6: input frames, ground truths, 3D CNN-LSTM score-maps, binary FG masks with empirical, Otsu and Gaussian smoothing + Otsu thresholds.

(b) F-measure on Office dataset.

Figure 9.8: Qualitative analysis on Office video sequence.

(a) Sample results of the Canoe dataset. Col. 1-6: input frames, ground truths, 3D CNN-LSTM score-maps, binary FG masks with empirical, Otsu and Gaussian smoothing + Otsu thresholds.

(b) F-measure on Canoe dataset.

Figure 9.9: Qualitative analysis on Canoe video sequence.

A visual inspection is carried out by comparing the predicted FG regions with the ground truths. We limit the qualitative presentation with two samples per data sequence as shown in Fig. 9.7 - Fig. 9.13 due to space constraints. A bar chart on the right of each visual sample (Gl-th - Global threshold, Ot-th - Otsu’s threshold,
(a) Sample results of the Boats dataset. Col. 1-6: input frames, ground truths, 3D CNN-LSTM score-maps, binary FG masks with empirical, Otsu and Gaussian smoothing + Otsu thresholds.

(b) F-measure on Boats dataset.

Figure 9.10: Qualitative analysis on Boats video sequence.

(a) Sample results of the Overpass dataset. Col. 1-6: input frames, ground truths, 3D CNN-LSTM score-maps, binary FG masks with empirical, Otsu and Gaussian smoothing + Otsu thresholds.

(b) F-measure on Overpass dataset.

Figure 9.11: Qualitative analysis on Overpass video sequence.

(a) Sample results of the Boulevard dataset. Col. 1-6: input frames, ground truths, 3D CNN-LSTM score-maps, binary FG masks with empirical, Otsu and Gaussian smoothing + Otsu thresholds.

(b) F-measure on Boulevard dataset.

Figure 9.12: Qualitative analysis on Boulevard video sequence.

Ot-Ga-th - Otsu’s Gaussian smoothed Otsu’s threshold) compares the average performance our 3D CNN-LSTM model with our MV-FCN. Full segmentation results will be available on the project site.
9.4.2 Quantitative analysis

The standard performance measure in terms of f-measure (refer Chapter 1, Section 1.3) is tabulated in Table 9.4. While the plots in Fig. 9.14 and box plots in Fig. 9.15 analyze the performance of the proposed network in comparison to the single frame-based MV-FCN, existing traditional approaches, and NN-based algorithms from recent years. Figure 9.14 compares the results wrt datasets, where the best FoM of the proposed model is compared with the best results of our MV-FCN and all other methods listed in Table 9.4. On the other hand, Figure 9.15 summarizes all the results of individual methods across all datasets. From these graphical analysis it is found that the NN-based models have the potential for precise FG segmentation, while our proposed 3D CNN-LSTM outperforms all the methods including our MV-FCN. Hence, It is noticed that using Gaussian smoothing with Otsu’s algorithm for 3D CNN-LSTM does not have significant impact. It proves that the network generates very strong FG probability map that has ignorable noise.

The key aspects of the compared existing methods as follows (detailed literature review is given in Chapter 3). The [66] and [149] are pixel-based probabilistic models. The [177] is a Stacked Denoising Auto-Encoder (SDAE) learning module along with a binary scene modeling based on density analysis. Similarly, [181] also takes advantage of NNs with a stacked multilayer Self-Organizing Map (SOM) to model the BG. [41] extends the basic SOM model of [181] with a self-balancing multi-layered SOM that tracks a long time pixel dynamics for better FG detection. Besides, [137], [48], [5] employ local features, such as, Local Ternary Pattern (LTP) and Local Binary Patterns (LBP) to form background models then to detect FG. On the other hand, [8], [169], and [12] exploit CNN-based models for FG segmentation.
<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>3D CNN-LSTM</th>
<th>Single Fr. MV-FCN</th>
<th>Prob. Models</th>
<th>Others</th>
<th>NN Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Highway</td>
<td>95.20</td>
<td>95.58</td>
<td>92.64</td>
<td>87.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Office</td>
<td>95.19</td>
<td>95.14</td>
<td>96.10</td>
<td>91.75</td>
<td>92.64</td>
</tr>
<tr>
<td></td>
<td>Canoe</td>
<td>93.83</td>
<td>88.23</td>
<td>94.04</td>
<td>93.15</td>
<td>79.23</td>
</tr>
<tr>
<td></td>
<td>Boats</td>
<td>90.88</td>
<td>91.14</td>
<td>87.27</td>
<td>86.00</td>
<td>83.24</td>
</tr>
<tr>
<td></td>
<td>Overpass</td>
<td>90.21</td>
<td>88.02</td>
<td>88.25</td>
<td>87.07</td>
<td>69.24</td>
</tr>
<tr>
<td></td>
<td>Boulevard</td>
<td>87.07</td>
<td>81.35</td>
<td>87.37</td>
<td>80.70</td>
<td>81.74</td>
</tr>
<tr>
<td>CopyMachine</td>
<td></td>
<td>95.53</td>
<td>94.58</td>
<td>94.43</td>
<td>93.49</td>
<td>92.89</td>
</tr>
</tbody>
</table>

Table 9.4: Performance Comparison in terms of FoM: Global-th and Otsu-th stand for the two thresholding methods applied. Values in red are the best FoM while the ones in blue are the second best.
In overall analytic observation, the proposed model performs consistently regardless of the challenging conditions. The model mainly outperforms the existing approaches when the background has dynamic nature. It shows that considering the long-short term temporal dependencies is crucial for an accurate foreground localization. When an average performance is taken across all the video sequences, the CNN-LSTM based EnDec network, gains $\sim 11\%$ and $\sim 6.5\%$ compared to conventional statistical approaches and modern NN-based learning systems respectively. Besides, the computationally intensive nature of LSTM results $\approx 69ms$ mean average processing time per frame, i.e., $\approx 14.5FPS$ predication speed on GeForce GTX 1080 Ti when batch size is set to 8.

![Performance vs dataset](image)

Figure 9.14: Performance vs dataset.

### 9.5 Conclusion

This work excogitates an encoder-decoder deep learning model for video foreground segmentation that harnesses 3D convolution/transpose convolution, LSTM modules, and residual connections. It captures short-long term spatiotemporal features collectively from a set of $n$ frames before predicting the FG region of the current frame. In contrast to conventional approaches, DL models do not require feature engineering
and manual parameter tuning as the network parameters are learned from FG segmentation exemplars during training. Therefore, it is reckoned that the proposed 3D CNN-LSTM is a new addition to the state-of-the-art FG segmentation/localization algorithms.

The qualitative and quantitative analysis with seven benchmark video sequences demonstrates that the network is a performant model when dealing with FG detection challenges involving lighting variations, cast shadow, dynamic backgrounds in indoor and outdoor environments. The results also show that our model superiorly performs most of the cases when compared with traditional and modern NN-based foreground segmentation methods. However, improvement can be still made, for instance, (i). The network can be optimized for less number of trainable parameters. We leave this as our future direction, and (ii). The input data can include a generic background model either a precomputed one or computed in run-time. So, it is expected to capture highly temporal cues of moving objects resulting high delineated segmentation. The proposed 3D CNN-LSTM model is applicable to many computer vision-based intelligent systems not just limited to path segmentation for autonomous vehicles and crowd segmentation for video surveillance. Finally, it is understood that developing a robust FG segmentation model is still an open-end problem.
Chapter 10
Conclusion

The dissertation is an effort to provide a detailed overview of video foreground localization, followed by proposing a number of enhancements to traditional methods and modern deep learning-based approaches. The video foreground localization is often considered as one of the most core and challenging task in computer vision. It has many applications in a variety of areas from video surveillance to autonomous driving. Through this work, we have tried to explore and exploit new methods that can help precise motion detection followed by a full mask of the moving object extraction towards video foreground localization.

In Chapter 1 the fundamental aspects and rationale of video foreground localization have been discussed followed by a discussion on the various applications and standardized evaluation method to quantify the quality of localization. It is also important to highlight the exploited methodologies and the dissertation’s expected outcomes. Thus, Chapter 1 serves that purpose too. As a preparation to the reader, the Chapter 2 lays a strong foundation through providing amble backgrounds from definitions to a depth of essential topics involved in this dissertation. It is then extended, whereby Chapter 3 is dedicated to providing a well-round literature review on existing methods for foreground localization, their types, and advantages.

With the foundation ready, the dissertation elaborates the proposed enhancements of traditional approaches for foreground localization in Chapters 4, 5, and 6. Later, it introduces improvements to deep learning-based encoder-decoder foreground detection models with spatiotemporal features in Chapters 7, 8, and 9. Each of the proposed works inherits some contributions and limitations, as summarized next.

10.1 Contributions and limitations

The contributions of the proposed models can be divided in two broad categories:
1. Traditional methods: Simple sample-based and improving foreground detection of GMM.


The categories and their characteristics are discussed in the following subsections followed by a summary.

### 10.1.1 Traditional methods

Although there have been many sophisticated algorithms introduced, they are generally very high complexity ones and are not necessary for specific surveillance purposes, such as monitoring an ATM in a shopping complex or bank. Because in such conditions, the surveillance camera is fixed at a place and the background environment is known prior to the actual monitoring operation. In such cases, it is recommended to employ simplistic models to detect the foregrounds, i.e., the moving objects in the given environment. To this end, Chapter 4 presents two simplistic algorithms: a probabilistic based model with non-supervised threshold computation and a 3D-color space model using distance vector for background suppression. The empirical study is carried out with various color spaces, like \(RGB\), \(YCbCr\), \(YIQ\), and \(YUV\). It is found that using a pixel-level background modeled by PMF in temporal domain and 3D-color space improves the FG localization performance \(\sim 15\%\) compared to the traditional GMM-based model, in average.

The main limitation of the proposed approach is that demand for a high number of prior samples and sensitiveness to illumination and shadow. Thus, as for future directions, the following can be considered.

i A weight parameter can be introduced to control the prior probabilities of each intensity level at each pixel. For instance, if an intensity value is classified as FG in the current frame, it will have less probable to be in BG at the same pixel coordinate, so its weight in the BG prior probability can be set to lower than its initial value. By doing so, the model will have the ability to adopt a new BG intensity level and to remove an old intensity value which becomes least probable as the scene evolve.

ii The Euclidean distance-based algorithm can be improved by taking variance information of each channel into the distance calculation like in Mahalanobis distance so that it will be able to adopt rapid scene changes.
Hence, the traditional GMM-based background subtraction methods usually perform well when the background is stationary. However, they require parameter tuning to deal with dynamic backgrounds, whose background pixel values change over the time. Notably, the threshold which determines the pixels associated with moving objects from the resultant of BGS. Considering that, the Chapters 5 and 6 intent to present a novel idea to update the threshold of GMM-based BGS with respect to color distortion, similarity and illumination measures in pixel-level. These cues are interesting ones as the color similarity and distortion have not been used for foreground localization by the CV community so far.

Thus, Chapter 5 empirically derives a unified model using the measures above that adaptively computes an appropriate threshold to extract the FG region from the resultant of BGS. The conducted experiments demonstrate the effectiveness of the proposed unified model. In comparison to some of the long-familiar GMM-based BGS methods in the literature, the model gains $\approx 20\%$ improvement in terms of f-measure. However, the proposed method does not attempt to provide a real-time performance, rather it investigates a potential utilization of the aforementioned measures to set a threshold automatically to detect moving objects in video sequences. Thus, it lacks processing speed as it gets less than a frame per second while the standard GMM achieves higher than a frame per second in average considering the processing time across all the experiments.

Similarly, Chapter 6 extends the idea of exploiting the color similarity, distortion, and illumination measures through a fusion strategy. It addresses the difficulty in setting an adaptive threshold in the multi-model Gaussian-based BG-FG separation through a novel FG enhancement strategy by assimilating color and illumination measures. It formulates the problem mathematically by using a histogram of a fused feature of color and illumination measures resulting improvement of the FGL by introducing the following contributions:

i A new distance measure to check if a pixel matches a Gaussian distribution.

ii A new strategy to enhance the primary resultant of traditional background subtraction with fusion of color and illumination measures.

iii A computationally speedy histogram-based methodology to find an appropriate threshold adaptively to separate BG and FG.

iv A FG validation process through probability estimation of multivariate Gaussian model distribution.
The experimental study demonstrates that the proposed approach works well in challenging conditions, at the same time, it performs competitively against state-of-the-art GMM-based algorithms and some other traditional methods as well. Although the model performs better than other compared GMM-based algorithms, it does not have a strategy to tackle with the cast shadows of the moving objects and sudden changes in region-level. For instance, when the model has an overall average performance of 0.8639 f-measure across all the tested baseline data sequences in Table 6.1, it falls to 0.7103 f-measure due to intermittent moving shadows in the video sequence, the BusStation from the CD-net benchmark database.

Thus, the proposed models in Chapters 5 and 6 expose to the drawback of higher processing time due to the layered FG validation process and having no mechanism to counterbalance the effect of moving object’s cast shadows. It prompts to have a future work dedicated to fixing the processing time from either the framework perspective or the programming perspective, for instance, utilizing GPUs and handling the shadows jointly with BG modeling. Also, thoughts can be given to exploiting color and illumination features in the local-temporal level instead of global-temporal level to enhance foreground features and to extract the foreground region robustly.

10.1.2 Deep learning-based models

The deep learning networks have become a state-of-the-art solution in computer vision and been successfully applied to big data for knowledge discovery, knowledge application, and knowledge-based prediction. As a result of that, the deep CNN has become a cornerstone of the modern era autonomous driving, video surveillance, drug and food inspection, and so forth. Thus, this dissertation extends its work on utilizing DL for FGL. It harnesses the power of CNN-based visual semantic segmentation strategies for video foreground localization. It improves the basic encoder-decoder CNN through innovative approaches, viz.:

1. Slow encoder-decoder CNN with micro auto-encoder blocks, batch normalization, and channel-wise residual feature concatenation (Chapter 7).

2. Multi-view receptive field to capture scale-invariant features of FG objects (Chapter 8).

3. CNN-LSTM model to exploit spatiotemporal cues for a delineate FGL (Chapter 9).
The Chapter 9 harnesses the ability of LSTM modules in handling time series data, like speech signal. It implements a 3D CNN-LSTM network that takes four frames at a time to predict the FG region in the current frame. Thus, the intuition of utilizing the LSTM units instead of pure 3D convolution is that capturing not just temporal features but long-short term spatiotemporal cues. This LSTM-based spatiotemporal model gains $\sim 11\%$ and $\sim 6.5\%$ in an overall average performance when compared to conventional statistical approaches and modern NN-based learning systems respectively. However, it faces low fps with $\sim 15$ due to the massive computational overhead of stacked LSTM cells both in the encoding and decoding subnetworks. It is then addressed by taking only two consecutive frames in grayscale stacked depthwise along with a generalized BG model without LSTM modules in Chapter 7 and 8.

The Chapter 8 introduces akin Inception modules with a multi-view receptive field to capture scale invariant FG clues. Moreover, to capture the spatiotemporal features, it uses a temporally median filtered BG model stacked as the third channel of input data that takes two consecutive frames as the first two channels. It achieves an average of $\sim 8.75\%$ improvements compared to the state-of-the-art and a high processing speed of 45 fps on a GTX 1080 Ti.

The Chapter 7, on the other hand, proposes two elegant ideas to improve the learning ability of a basic image-to-image CNN network through micro-auto-encoder blocks in the subsampling subnetwork and slow decoding blocks in the upsampling subnetwork. That chapter also presents empirical grounds for how well the proposed architectural changes improve the FGL and carries out rigorous experiments on various challenging benchmark video sequences. This proposed model also takes the same input configuration as in Chapter 8 to extract spatiotemporal information of moving objects. Besides the complexity of the structure, it records very competitive performance it achieves a higher frame rate of 49 fps on a GTX 1080 Ti.

In terms of f-measure, both multi-view CNN and 3D CNN-LSTM models with spatiotemporal cues perform quite the same with a mean average of $\sim 92\%$, while the slow-encoder slow-decoder model achieves a mean average f-measure of $\sim 93\%$ with even higher FPS than the other two architectures. However, we believe that optimizing the 3D CNN-LSTM model for the number of filters and number of layers, including the input layer configuration such a way to account a generic BG model will improve its performance (higher f-measure and higher fps). We leave this for future direction.
10.2 Applications

The effectiveness of most of the VCA-based high-level tasks depend on the robustness of FGL model. Therefore, this work can be extended towards multi-object FG detection that can be applicable to the following applications: multi-tasking event/activity and action detection/recognition, simultaneous localization and mapping (SLAM), intrusion detection, video surveillance, object counting and tracking, obstacle avoidance, path finding, human machine interaction, image quality assessment, selective data compression, autonomous driving and traffic safety, and so forth.

10.3 Dissemination

One of the objectives set at the beginning of this Ph.D. is disseminating the findings at every stage through internationally recognized conferences and journals. The following statistic summarizes the publication, wherein I am the author or co-author of the work.

<table>
<thead>
<tr>
<th>Dissemination type</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Journal</td>
<td></td>
</tr>
<tr>
<td>Published</td>
<td>2</td>
</tr>
<tr>
<td>Under review</td>
<td>3</td>
</tr>
<tr>
<td>Conference</td>
<td></td>
</tr>
<tr>
<td>Published</td>
<td>15</td>
</tr>
<tr>
<td>Submitted</td>
<td>2</td>
</tr>
<tr>
<td>Archive</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 10.1: Publication summary since 2015.
Bibliography


154


Appendix A

IEEE Permission to Reprint

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Windsors products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to https://www.ieee.org/publications_standards/publications/rights/permissions_faq.pdf to learn how to obtain a License from RightsLink.
Appendix B

Elsevier Permission to Reprint

In reference to Elsevier copyrighted material which is used with permission in this thesis, the Elsevier does not endorse any of University of Windsors products or services. Internal or personal use of this material is permitted. If interested in reprinting/re-publishing Elsevier copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please visit: https://www.elsevier.com/about/our-business/policies/copyright#Author-rights
Appendix C

Code Samples

The code samples will be provided upon request. You are welcome to reach the author at thangara@uwindsor.ca.
Vita Auctoris

Name : Thangarajah Akilan

Education :
2018 Doctor of Philosophy
Electrical and Computer Engineering
University of Windsor, Canada

2013 Master of Engineering (Thesis)
Microelectronics and Embedded Systems
Asian Institute of Technology, Thailand

2011 Bachelor of Engineering
Computer Systems Engineering
Curtin University, Australia