VISUAL OBJECT TRACKING IN VIDEOS FROM LOW-ALTITUDE FLYING UAV'S UNDER OCCLUSION

by

Bahri MARAŞ

B.S., Electrical and Electronics Engineering, Turkish Naval Academy, 2005

M.S., Electrical Engineering, University of Southern California, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2025

ACKNOWLEDGEMENTS

I would like to thank,

My supervisor Prof. Ayşın ERTÜZÜN and my co-advisor Prof. Nafiz ARICA for their guidance, encouragement and support,

My dear wife Seher Ersoy MARAŞ and my precious son Aydın Ersoy MARAŞ for their patience, possession and support,

My beloved family, Aydın MARAŞ, Özden MARAŞ and Berk MARAŞ, for being next to me always,

The Scientific and Technological Research Council of Türkiye (TÜBİTAK) for supporting the study, presented in this thesis, under ARDEB 1001 Program (Support Program for Scientific and Technological Research Projects) with project number 119E596.

ABSTRACT

VISUAL OBJECT TRACKING IN VIDEOS FROM LOW-ALTITUDE FLYING UAV'S UNDER OCCLUSION

Aerial imagery from flying unmanned air vehicles has been very important for many applications such as satellite remote sensing, object detection and tracking. Visual problems in videos from unmanned air vehicles have higher complexity compared to videos taken from ground-based sources.Moreover, if the videos are taken from low-altitude, the visual problems are more severe; thus, processing of such videos is more challenging. This thesis concentrates on this challenging problem and focuses on target tracking under occlusion, which occurs when a target becomes invisible due to blocking by other moving or static objects. First, a new method, named as PFCNN is developed to track vehicles against most visual problems, except occlusion. PFCNN combines a particle filer with a convolutional neural network; and this combination creates a powerful method for tracking against most of the visual problems. Computer simulations verify that PFCNN has superior performance compared to deep learning-based state-of-the-art methods. Then, PFCNNOcc and final PFCNNOcc,two novel methods, are developed to achieve robust target tracking, especially against short- and/or long-term partial and full occlusion. PFCNNOcc architectures also combine convolutional neural network models with particle filters in order to predict optical flow, occlusion maps, segmentation masks, and classification scores, as well as the degree of occlusion. They use the following functions which are proposed in this thesis: 1. Re-Identification, 2. Occlusion Detector and Calculator, 3. Full Occlusion Dense Search, 4. Merged-Target Detection. Computer simulations verify that these functions lead the final PFCNNOcc to the top of the state-of-the-art Siamese network-based methods. A novel synthetic dataset, named S-UAV, is generated to train and to test the proposed architectures, and it fills the gap, arisen for such a dataset in the literature.

ÖZET

ALÇAK İRTİFADA UÇAN İHA VİDEOLARINDA ÖRTÜŞME ALTINDA GÖRSEL NESNE TAKİBİ

Uçan insansız hava araçlarından elde edilen hava görüntüleri; uydu uzaktan algılama, nesne tespiti ve takibi gibi birçok uygulama için önemli olmuştur. İHA'lardan alınan videolarda karşılaşılan görsel problemler, yere sabitlenmiş kaynaklardan alınan videolara kıyasla daha karmaşıktır. Ayrıca, videolar alçak irtifadan çekilmişse, görsel problemler daha da belirginleşmekte ve bu tür videoların işlenmesini daha zor hale getirmektedir. Bu tez, bu zorlu probleme odaklanmakta ve özellikle bir hedefin, hareketli veya sabit nesneler tarafından engellenmesi sonucu görünmez hale gelmesiyle oluşan örtüşme durumlarında hedef takibine yoğunlaşmaktadır. İlk olarak, örtüşme harici görsel problemlere karşı araç takibi yapabilen yeni bir yöntem olan PFCNN geliştirilmiştir. PFCNN, parçacık filtrelerini konvolüsyonel sinir ağlarıyla birleştirerek, görsel problemlere karşı güçlü bir takip yöntemi oluşturmuştur. Bilgisayar simülasyonları, PFCNN'in derin öğrenme tabanlı güncel yöntemlere kıyasla üstün performansa sahip olduğunu doğrulamaktadır. Daha sonra, kısa ve/veya uzun süreli kısmi ya da tam örtüşmelere karşı dayanıklı hedef takibi gerçekleştirmek amacıyla PFCNNOcc ve nihai PFCNNOcc isimli iki yeni yöntem geliştirilmiştir. PFCNNOcc mimarileri, konvolüsyonel sinir ağı modellerini parçacık filtreleriyle birleştirerek optik akışı, örtüşme haritalarını, segmentasyon maskelerini, sınıflandırma skorlarını ve örtüşme derecesini tahmin edebilmektedir. Bu mimariler, tez kapsamında önerilen şu işlevleri kullanmaktadır: 1.Yeniden Tanımlama, 2.Örtüşme Tespit Edici ve Hesaplayıcı, 3.Tam Örtüşme Yoğun Arama, 4.Birleşik Hedef Tespiti. Bilgisayar simülasyonu, bu işlevlerin nihai PFCNNOcc yöntemini, son teknoloji Siyam ağı tabanlı yöntemler arasında en üst seviyeye taşıdığını göstermektedir. Önerilen mimarilerin eğitimi ve testi için S-UAV adlı yeni bir sentetik veri kümesi oluşturularak literatürde bu tür bir veri kümesine duyulan ihtiyaç karşılanmıştır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES xi
LIST OF TABLES xx
LIST OF SYMBOLS xxii
LIST OF ACRONYMS/ABBREVIATIONS xxiv
1. INTRODUCTION 1
1.1. Visual Problems in Object Tracking 1
1.2. Occlusion Problem 3
1.3. Problem Statement 5
1.4. Overview of the Proposed System 8
1.5. The Contributions 10
1.6. Outline of the Thesis 14
2. LITERATURE SURVEY 15
2.1. State-of-the-Art Discriminative Methods Visual Object Tracking 15
2.1.1. Tracking by Handcrafted Feature-Based Methods 15
2.1.2. Tracking by Deep Feature Based Methods 17
2.1.2.1. Tracking-by-Detection Based Methods 17
2.1.2.2. Correlation Filter Based Methods 18
2.1.2.3. Siamese Network-Based Methods 20
2.1.2.4. Reinforcement-Learning Based Methods 22
2.2. Visual Object Tracking Under Occlusion 22
2.2.1. Traditional Optical Flow Methods 24
2.2.1.1. Variational Methods 24
2.2.1.2. Patch Based Methods 24
2.2.1.3. Patch Based Methods with Variational Refinement . . 25
2.2.2. DL-Based Optical Flow Methods 25

2.2.2.1. DL-Based Optical Flow Methods with Supervised Learning . . . 26
2.2.2.2. DL-Based Optical Flow Methods with Unsupervised Learning . . . 29
2.2.2.3. Comparison of DL-Based Optical Flow Methods in Terms of Learning . . . 32
2.2.3. Occlusion Detection with Optical Flow Estimation . . . 34
2.3. Overview . . . 38
3. PFCNN : PARTICLE FILTER WITH CNN METHOD . . . 40
3.1. PFCNN . . . 40
3.1.1. Introduction . . . 40
3.1.2. Theorical Background of PFCNN: SIR Particle Filter . . . 44
3.1.3. Architecture of PFCNN . . . 47
3.1.3.1. Initialization and Particle Sampling . . . 49
3.1.3.2. Particle Rescaling . . . 51
3.1.3.3. CNN Model . . . 54
3.1.3.4. Object State Estimation . . . 55
3.1.3.5. Unification . . . 57
3.1.4. Experiments . . . 64
3.1.4.1. Particle Filter and CNN Training . . . 66
3.1.5. Discussion . . . 68
3.1.6. Conclusion . . . 69
3.2. PFCNN versus APF . . . 70
3.2.1. Introduction . . . 70
3.2.2. Architecture . . . 71
3.2.2.1. Evolution of APF . . . 71
3.2.2.2. Implementation . . . 76
3.2.2.3. Analogy Between APF and SIR PF in the Proposed PFCNN Model . . . 77
3.2.3. Experiments . . . 79
3.2.4. Discussion . . . 80

3.2.5. Conclusion 80
4. A NOVEL APPROACH TO UAV DATASET CREATION 82
4.1. Introduction 82
4.2. A Novel, Synthetic UAV Dataset Creation Approach:S-UAV 82
4.2.1. Unity Motor and Models 84
4.2.1.1. Modeling the Virtual World with the Unity Engine . . 85
4.2.1.2. Unity Vehicle Mask/Segmentation Information 85
4.2.1.3. Unity Vehicle Location Information 86
4.2.2. Creation of Optical Flow, Occlusion and Segmentation as Ground Truth 88
4.2.2.1. Creation of Optical Flow and Occlusion Dataset Using Homography 88
4.2.2.2. Creation of the 250x250 Resolution Dataset 91
4.2.2.3. Creation of Segmentation Information 94
4.3. Creation of the Segmentation Ground Truth for UAV 123 Dataset . . . 95
4.4. Conclusion 97
5. PFCNNOCC : A ROBUST VEHICLE TRACKING METHOD UNDER OCCLUSION USING OPTICAL FLOW AND SEGMENTATION IN LOW-ALTITUDE UAV VIDEOS 99
5.1. Introduction 99
5.2. Design and Training of CNN Models in the PFCNNOcc 103
5.2.1. Architecture 103
5.3. Tracking with PFCNNOcc 104
5.3.1. Architecture 106
5.3.1.1. The First Version of PFCNNOcc Tested in S-UAV Dataset 106
5.3.1.2. The Final Version of PFCNNOcc Tested in Real-Time UAV Dataset, UAV123 131
5.4. Experiments 148
5.4.1. Experiments with the Training of the CNN Models 148
5.4.2. Experiments with the First Version of the PFCNNOcc 150

5.4.3. Experiments with the Final Version of the PFCNNOcc 152
5.4.4. Experiments for the Ablation Study with the Final Version of the PFCNNOcc 155
5.5. Discussion 156
5.6. Conclusion 160
6. SIMULATION RESULTS AND DISCUSSIONS 162
6.1. Simulation Results of PFCNN 162
6.1.1. Overall Comparison 163
6.1.2. Attribute-Based Comparison 164
6.1.3. Scenario-Based Comparison 167
6.2. Simulation Results of Auxiliary Particle Filter with CNN 172
6.2.1. Overall Comparison 172
6.2.2. Attribute-Based Comparison 173
6.2.3. Scenario-Based Comparison 175
6.3. Simulation Results of PFCNNOcc 178
6.3.1. Simulation Results of the First Version of the Tracking Model . 178
6.3.1.1. Scenario-Based Simulation Results of a_a_0_0_g 179
6.3.1.2. Scenario-Based Simulation Results of a_a_1_0_g 183
6.3.1.3. Scenario-Based Simulation Results of a_a_2_0_g 185
6.3.1.4. Scenario-Based Simulation Results of a_a_14_0_g 188
6.3.1.5. Scenario-Based Simulation Results of a_c_1_0_g 190
6.3.1.6. Scenario-Based Simulation Results of c_a_2_0_g 192
6.3.1.7. Scenario-Based Simulation Results of c_a_6_0_g_1 . . . 195
6.3.1.8. Scenario-Based Simulation Results of c_a_6_0_g_2 . . . 198
6.3.2. Simulation Results of the Final Version of the Tracking Model and the State-Of-The-Art Trackers on the UAV123 10fps Dataset 200
6.3.2.1. Overall Comparison 201
6.3.2.2. Attribute-Based Comparison 205
6.3.2.3. Scenario-Based Comparison 215
6.3.3. Simulation Results for the Ablation Study with the Final Version of the Tracking Model on the UAV123 10fps Dataset 225

6.3.3.1. Overall Comparison 226
6.3.3.2. Attribute-Based Comparison 227
6.3.3.3. Scenario-Based Comparison 229
7. CONCLUSIONS AND FUTURE WORK 241
REFERENCES 246
APPENDIX A: THE ALGORITHM TO CREATE THE OPTICAL FLOW DATA 259
APPENDIX B: TRAINING OF CNN MODELS 262
B.1. Training of the CNN Model for Optical Flow and Occlusion 262
B.2. Training of the CNN Model for Segmentation 271
B.3. Training of the CNN Model for Classification 274

LIST OF FIGURES

Figure 1.1. Types of occlusion. 4

Figure 1.2. The general concept of the proposed method using different occlusion detection and state estimation/prediction mechanisms. 9

Figure 1.3. The overview diagram of the PFCNNOcc. 10

Figure 3.1. Overall method of the PFCNN. 43

Figure 3.2. Initialization and particle sampling. 49

Figure 3.3. Particle Rescaling: The images in the rescaling matrix are scaled by 1-sP, 1, and 1+sP, depending on their position. The images on the diagonal are the ones that do not face any aspect ratio variation; thus, they are scaled by the same amount along both directions. However, the images on the off-diagonal positions experience aspect ratio variations; thus, they are scaled by different amounts along each direction. 58

Figure 3.4. Object state estimation. 59

Figure 3.5. Overall unification module. 60

Figure 3.6. Unification module fine-tuner. 61

Figure 3.7. Unification module final scale and state estimator. 63

Figure 3.8. Unification module reference model calculator. 65

Figure 3.9. Performance metrics. 67

Figure 3.10. The pseudocode of APF with evolution of posterior pdf. 78

Figure 3.11. Overall method of APF 79

Figure 4.1. General method to create S-UAV dataset. 84

Figure 4.2. Scenario naming policy. 86

Figure 4.3. Sunny and snowy conditions of urban and rural scenarios. 87

Figure 4.4. Calculation of optical flow vectors. 89

Figure 4.5. Combined image of horizontal and vertical optical flow vectors. . . 90

Figure 4.6. Final optical flow image. 90

Figure 4.7. Visualized flo file. 91

Figure 4.8. Flow diagram for the creation of optical flow, occlusion, and segmentation information with a resolution of 250x250. 92

Figure 4.9. Results of optical flow, occlusion, and segmentation information for video images 398 and 399. 94

Figure 4.10. Distribution graph of the 250x250 resolution dataset labels prepared with a total of 100 scenarios. 95

Figure 4.11. Segmentation annotation with computer vision annotation toolbox. 96

Figure 5.1. General flow diagram of the PFCNNOcc. 101

Figure 5.2. General training flow diagram of CNN models for detection of occlusion with optical flow, segmentation, and classification. 105

Figure 5.3. Transition of occlusion states. 108

Figure 5.4. Overall flow diagram of the first version of PFCNNOcc. 109

Figure 5.5. Re-Identification module. 111

Figure 5.6. The pseudocode of the Re-Identification module. 112

Figure 5.7. Occlusion detector and calculator module. 114

Figure 5.8. Occlusion detector and calculator module-forward processor. . . . 116

Figure 5.9. The pseudocode of the Occlusion Detector-Forward Processor. . . 117

Figure 5.10. Occlusion detector and calculator module-backward processor. . . 118

Figure 5.11. The pseudocode of the Occlusion Detector-Backward Processor. . 119

Figure 5.12. The pseudocode of the Occlusion Calculator. 121

Figure 5.13. Particle filter module. 124

Figure 5.14. Full occlusion ROI dense search module. 127

Figure 5.15. The pseudocode of the Re-Identification Module Full Occlusion. . 129

Figure 5.16. Re-Identification module full occlusion. 130

Figure 5.17. Overview of overall flow diagram of the final version of PFCNNOcc 133

Figure 5.18. The pseudocode of the Re-Identification Module Final PFCNNOcc. 138

Figure 5.19. Occlusion detector and calculator module final PFCNNOcc. 139

Figure 5.20. Re-Identification module final PFCNNOcc. 140

Figure 5.21. The pseudocode of the Occlusion Calculator Final PFCNNOcc. . . 142

Figure 5.22. Merged target detection problem. 143

Figure 5.23. Full occlusion ROI dense search module final PFCNNOcc. 146

Figure 6.1. Overall precision and success plots of the SOTA trackers and the PFCNN. 163

Figure 6.2. Attribute-based precision plots of the SOTA trackers and the PFCNN. 165

Figure 6.3. Attribute-based success plots of the SOTA trackers and the PFCNN. 166

Figure 6.4. Scenario-based precision plots of the SOTA trackers and the PFCNN. 168

Figure 6.5. Scenario-based precision plots of the SOTA trackers and the PFCNN (cont.). 169

Figure 6.6. Scenario-based success plots of the SOTA trackers and the PFCNN. 170

Figure 6.7. Scenario-based success plots of the SOTA trackers and the PFCNN (cont.). 171

Figure 6.8. Overall precision and success Plots: PFCNN versus APF. 172

Figure 6.9. Attribute-Based Precision Plots PFCNN versus APF. 173

Figure 6.10. Attribute-based success plots PFCNN versus APF. 174

Figure 6.11. Scenario-based precision plots PFCNN versus APF. 176

Figure 6.12. Scenario-based precision plots PFCNN versus APF (cont.). 176

Figure 6.13. Scenario-based success plots PFCNN versus APF. 177

Figure 6.14. Scenario-based success plots PFCNN versus APF (cont.). 177

Figure 6.15. A snapshot from scenario a_a_0_0_g during tracking. 180

Figure 6.16. Simulation results of scenario a_a_0_0_g. 181

Figure 6.17. The correlation between the occlusion detection plot and the snapshot image. 182

Figure 6.18. A snapshot from scenario a_a_1_0_g during tracking. 183

Figure 6.19. Simulation results of scenario a_a_1_0_g. 184

Figure 6.20. A snapshot from scenario a_a_2_0_g during tracking. 185

Figure 6.21. Simulation results of scenario a_a_2_0_g. 186

Figure 6.22. A snapshot from scenario a_a_14_0_g during tracking. 188

Figure 6.23. Simulation results of scenario a_a_14_0_g. 189

Figure 6.24. A snapshot from scenario a_c_1_0_g during tracking. 190

Figure 6.25. Simulation results of scenario a_c_1_0_g. 191

Figure 6.26. A snapshot from scenario c_a_2_0_g during tracking. 193

Figure 6.27. Simulation results of scenario c_a_2_0_g. 194

Figure 6.28. A snapshot from scenario c_a_6_0_g_1 during tracking 196

Figure 6.29. Simulation results of scenario c_a_6_0_g_1. 197

Figure 6.30. A snapshot from scenario c_a_6_0_g_2 during tracking. 198

Figure 6.31. Simulation results of scenario c_a_6_0_g_2. 199

Figure 6.32. Overall precision and success plots of the SOTA trackers and the final version of PFCNNOcc. 203

Figure 6.33. Attribute-based precision plots of the SOTA trackers and the final version of PFCNNOcc. 207

Figure 6.34. Attribute-based precision plots of the SOTA trackers and the final version of PFCNNOcc. (cont.) 208

Figure 6.35. Attribute-based success plots of the SOTA trackers and the final version of PFCNNOcc. 212

Figure 6.36. Attribute-based success plots of the state-of-the-art trackers and the final version of PFCNNOcc (cont.). 213

Figure 6.37. Scenario-based precision plots of the SOTA trackers and the final version of PFCNNOcc. 218

Figure 6.38. Scenario-based precision plots of the SOTA trackers and the final version of PFCNNOcc (cont.). 219

Figure 6.39. Scenario-based precision plots of the SOTA trackers and the final version of PFCNNOcc (cont.). 220

Figure 6.40. Scenario-based precision plots of the SOTA trackers and the final version of PFCNNOcc (cont.). 221

Figure 6.41. Scenario-based success plots of the SOTA trackers and the final version of PFCNNOcc. 222

Figure 6.42. Scenario-based success plots of the state-of-the-art trackers and the final version of PFCNNOcc (cont.). 223

Figure 6.43. Scenario-based success plots of the SOTA trackers and the final version of PFCNNOcc (cont.). 224

Figure 6.44. Scenario-based success plots of the SOTA trackers and the final version of PFCNNOcc (cont.). 225

Figure 6.45. Overall precision and success plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers. 228

Figure 6.46. Attribute-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers. 231

Figure 6.47. Attribute-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.). 232

Figure 6.48. Attribute-based success plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers. 233

Figure 6.49. Attribute-based success plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.). 234

Figure 6.50. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers. 235

Figure 6.51. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.). 236

Figure 6.52. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.). 237

Figure 6.53. Scenario-based success plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers. 238

Figure 6.54. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.). 239

Figure 6.55. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.). 240

Figure A.1. Calculation of optical flow vectors. 261

Figure B.1. Pyramidal feature extractor. 263

Figure B.2. Flow model. 265

Figure B.3. Flow estimator. 266

Figure B.4. Flow context network. 267

Figure B.5. Occlusion model. 269

Figure B.6. Segmentation model. 273

Figure B.7. Classifier model. 275

LIST OF TABLES

Table 3.1. Selected UAV123 image sequences for testing. 66

Table 4.1. Content 92

Table 4.2. Quantity of outputs from the 250x250 resolution dataset. 94

Table 5.1. Selected image sequences from UAV123 10 fps dataset to test the final version of PFCNNOcc. 154

Table 6.1. Overall performance of the final version of PFCNNOcc and the SOTA trackers. The best three performances are highlighted in red, green, and blue colors. 202

Table 6.2. Attribute-based precision results of the SOTA trackers and the final version of PFCNNOcc. The best three performances are, respectively highlighted in red, green, and blue colors. 209

Table 6.3. Attribute-based success results of the SOTA trackers and the final version of PFCNNOcc. The best three performances are, respectively highlighted with red, green, and blue colors. 214

Table 6.4. Overall performance of the final version of PFCNNOcc and the ablation versions. The best three performances are highlighted with red, green, and blue colors. 227

Table 6.5. Attribute-based precision results of the final version of PFCNNOcc and ablation versions. The best three performances are highlighted in red, green, and blue colors. 230

Table 6.6. Attribute-based success results of the final version of PFCNNOcc and ablation versions. The best three performances are highlighted in red, green, and blue colors. 230

LIST OF SYMBOLS

$\mathbf{A}$	Transition function/matrix of SIR Particle Filter
$\mathbf{b}_{t,j}$	Auxiliary or parent weight of j the auxiliary particle filter
$bestBoundingBoxes^i_j$	The optimal bounding boxes or particles for jth particle sampled for ith
$\mathbf{C}$	Measurement function/matrix of SIR Particle Filter
dt	Variation in the time or the frame number
dx	Displacement of a pixel along x axes
dy	Displacement of a pixel along y axes
L	The number of the different scale values for x and y axeses to rescale a bounding box at time t
$P^i_{j,k}$	The class score, calculated by CNN, for the kth rescaled bounding box independently for jth particle sampled for ith part
$\mathbf{r}_t$	The noise vector in measurement equation of the particle filter
$RS_{t,l}$	Rescaled bounding box at time t using the scale parameter l.
$\mathbf{S}^i_{j,k}$	The matrix of all bounding boxes rescaled independently kth times for jth particle sampled for ith part
sP	Ultimate scale parameter in PFCNN
$TH1$	Threshold to decide partial occlusion
t	Represents the time or the frame number
$\mathbf{v}_t$	The noise vector in the process equation of the particle filter
$w_{t,j}$	The weight of jth particle at time t in the particle filter
x	Location of a pixel along x axes
$\mathbf{x}_t$	The state variable vector in the process equation of the particle filter
$\mathbf{x}Est^i_t$	The estimated vector for the state variables for each part, i, at time t in the particle filter
$\mathbf{z}_t$	The measurement vector in measurement equation of the particle filter
y	Location of a pixel along y axes

$\beta 1, \beta 2$ $(betas)$	Coefficients used for computing running averages of gradient and its square in the ADAM optimizer
$\delta(.)$	Delta function located at the position of the drawn particle of the particle filter
ϵ	Term added to the denominator to improve numerical stability in the ADAM optimizer
$\Psi()$	Charbonnier penalty function
$\sigma^2_{proc,loc}$	Variance term corresponding to the location state variable of $\mathbf{v}_t$
$\sigma^2_{proc,vel}$	Variance term corresponding to the velocity state variable of $\mathbf{v}_t$
σ^2_{meas}	Variance term of $\mathbf{r}_t$
$\Theta(t, xEstAll^l)$	The rescaling function that rescales a bounding box using $xEstAll^l$ at time t

LIST OF ACRONYMS/ABBREVIATIONS

ACT	Actor-critic Tracking
ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Attentional Aggregation Network
APF	Auxiliary Particle Filter
APN	Anchor Proposal Network
APN-DF	APN Depending On Dual Features
ARV	Aspect Ratio Variation
ATA	Average Tracking Accuracy
AUC	Area Under Curve
BackToBasics	Unsupervised Learning Of Optical Flow via Brightness Constancy and Motion Smoothness
BCA	Brightness Constancy Assumption
CCOT	Continuous Convolutional Operator Tracker
CM	Camera Motion
CNN	Convolutional Neural Network
CSK	The Circulant Structure of Tracking-By-Detection With Kernels
DaSiamRPN	Distractor-aware SiamRPN
DAT	Deep Attentive Tracker
DCF	Deep Correlation Filter
DCNN	Deep Convolutional Neural Network
D2CIP	Deep Convolutional Correlation Iterative Particle Filter
DL	Deep Learning
DSTFlow	Unsupervised deep learning for optical flow estimation
ECO	Efficient Convolution Operators for Tracking
EPE	End-Point-Error
ema	Estimated Moving Average

fps	Frame Per Second
FP	False Positive
FlowNet	Learning Optical Flow with Convolutional Networks
FlowNet2	Evolution of Optical Flow Estimation with Deep Networks
FlowNet3	Occlusions, Motion and Depth Boundaries with a Generic Network for Disparity, Optical Flow or Scene Flow Estimation
FM	Fast Motion
FN	False Negative
FOV	Field Of View
GAN	Generative Adverserial Network
Gappy PCA	Gappy Principal Component Analysis
GC	Graph Cut
GMM	Gaussian Mixture Model
GTA5	Grand Theft Auto 5
HR	High Resolution
HS	Horn and Schunck
IRR	Iterative Residual Refinement
IRR-PWC	Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation
KCF	Kernelized Correlation Filter
LDOF	Large Displacement Optical Flow
LearnByAnalogy	Reliable Supervision from Transformations for Unsupervised Optical Flow Estimation
LiteFlowNet	A Lightweight Convolutional Neural Network for Optical Flow Estimation
LK	Lucas Kanade
LOS	Line of Sight
LR	Low Resolution
LSTM	Long Short Term Memory
MDNet	Multi domain network tracker
MEEM	Robust Tracking With Multiple Experts Using Entropy Minimization

MulFeatOptFlowOcc	Self-attention-based Multiscale Feature Learning Optical Flow with Occlusion Feature Map Prediction
MirrorFlow	Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation
MTL	Multi Task Learning
MVA	Machine Vision and Applications Journal
OccDetProOptPF	Occlusion Detection and Processing Using Optical Flow and Particle Filter
OFIG	Optical Flow Image Generator
OOV	Out-of-view
OPE	One Pass Evaluation
OTP	Object Tracking Precision
PO	Partial Occlusion
pdf	Probability Density Function
PFCNN	A Robust Vehicle Tracking in Low-Altitude UAV Videos
PFCNNOcc	A Robust Vehicle Tracking Under Occlusion Using Optical Flow and Segmentation In Low-Altitude UAV Videos
PHD	Probability Hypothesis Density
PFCNNOccWOM	PFCNNOcc Without Merged Target Detection
PFCNNOccWOOccCalc	PFCNNOcc Without Occlusion Calculator
PMC	Parallel Multiscale Context-Based
PMC-PWC	Parallel Multiscale Context-Based Edge-Preserving Optical Flow Estimation With Occlusion Detection
PWC-Net	CNNs for OpticalFlow Using Pyramid, Warping, and Cost (PWC) Volume
PWC-Net+	Models Matter, So Does Training: An Empirical Study of CNNs for Optical Flow Estimation
PWC	Pyramid, Warping and Cost
PWOC-3D	Deep Occlusion-Aware End-to-End Scene Flow Estimation
RMSProp	Root Mean Squared Propagation
RNN	Recurrent Neural Network
RPN	Region Proposal Network

RLE	Run Length Encoding
ROI	Region Of Interest
SC	Scale Change
SegFlow	A Lightweight Convolutional Neural Network for Optical Flow Estimation
SiamAPN	Siamese APN
SiamFC	Fully Convolutional Siamese Tracker
SiamRPN	Siamese RPN
SIFT	Scale Invariant Feature Transform
SO	Similar Object
SIR	Sequential Importance Resampling
SOTA	State-Of-The-Art
sP	Scale Parameter
SSE	Sum of the squared error
STNs	Spatial Transform Networks
Spynet	Optical Flow Estimation using a Spatial Pyramid Network
SRDCF	Discriminative Correlation Filters
STRCF	Spatial-temporal Regularized Correlation Filter
STRUCK	Structured Output Tracking With Kernels
SymDenseOptFlow	Symmetrical Dense Optical Flow Estimation with Occlusions Detection
TDMPF	Tracking-by-detection With Marginal Particle Filter
TLD	Tracking With Learning Detection
TP	True Positive
UAV	Unmanned Air Vehicle
UDSCNNOcc	Uniform Dense Sampling CNN Occlusion
UnFlow	Unsupervised Learning of Optical Flow with a Bidirectional Census Loss
ULOFPC	Unsupervised Learning of Optical Flow With Patch Consistency and Occlusion Estimation
ULOFNL	Unsupervised Learning of Optical Flow With CNN-Based Non-Local Filtering

UnSupOptFlowOcc	Unsupervised Learning of Multi-Frame Optical Flow with Occlusions
VC	Viewpoint Change

1. INTRODUCTION

1.1. Visual Problems in Object Tracking

In recent years, aerial imagery from flying unmanned air vehicles (UAVs) has been very popular for many applications such as aviation, remote sensing, satellite remote sensing for traffic data collection, traffic congestion analysis, border protection and safety, object detection and tracking.

Because of the higher complexity of the visual problems in UAV videos compared to images taken from ground-based video sources, trackers face performance problems. Specifically, UAVs have a very large field-of-view (FOV) depending on the optics in front of the camera, and the observed objects in the scene are usually very small with low resolution and with less-discriminative features [1]. At ground-based visual systems, the mutual movement of the camera and the object in the scene is very limited, and hence this relative motion does not constitute a serious problem during tracking. But in UAVs, the camera on the UAV, the object on the ground, and the UAV itself can all be in action, and these mutual motion vectors give rise to challenging visual problems such as sudden scale, aspect ratio, and viewpoint variations as well as out-of-view and occlusion problems.

While the effects of these visual problems are increasing in low-altitude UAV videos, compared to those taken from the UAVs that are flying at high altitudes, the difficulty in the target tracking problem escalates in low-altitude UAV videos. In high altitudes, the FOV of the camera on the UAV is very large, and the pixel resolution of the objects on the scene is very low, hence, the possibility of realizing and detecting the effects of these visual problems is very small, and sometimes impossible. This is the main reason why low-altitude aerial images are studied in this thesis.

While the UAV is losing altitude or narrowing down the FOV of its camera, the scale increases and the size of the object gets larger. The effect of the scale variation is felt very drastically in low-altitude flying UAVs, and this variation requires robust model update, sampling and detection mechanisms. Aspect ratio change is the result of the rotation of the object or the variation in the point of view of the camera. The problematic part of the aspect ratio change is the orientation variation that directly affects the lengths of the width and the height of the bounding box. This problem requires a very precise sampling method that is integrated with a well-trained detection algorithm, coupled with a robust model update scheme. When the camera moves, the viewpoint of the tracked object is changed, and its visual model and aspect ratio are also changed with a large displacement that creates a big motion vector. This situation creates a series of visual problems that need a very robust sampling method to be able to cover a large area, depending on the displacement of the object. This module should be supported by a robust detection method, coupled with a well-trained model as well as a high-precision model update mechanism. Out-of-view and occlusion can be considered as the hardest visual problems to tackle, because of the vanishing parts of the objects during tracking. As a part of the object or itself is being vanished from the scene or occluded by another object, its visual model changes and undergoes a serious variation. The drastic change in the visual model requires a very robust detection method, supported by an occlusion detection and a calculation method to be able to understand the existence of the occlusion as well as out-of-view problems. One of the biggest complexities of these problems is detecting the target after its loss and reappearance. Hence, a re-detection mechanism is required to relocate the object without the knowledge of the possible future location. In Section 1.2, the occlusion problem will be elaborated. Other visual problems, such as turbulence or any vibration at UAV, create deformation in UAV videos, and this situation will cause the model drift problem, which makes model update a complex task.

1.2. Occlusion Problem

Occlusion in visual object tracking occurs when an object becomes invisible to the video camera's line of sight (LOS) due to another moving object or a static background object positioning itself between the tracked object and the camera.

Occlusion in UAV videos arises when a tracked object becomes partially or fully invisible to the UAV-mounted camera, often due to obstructions such as bridges, trees, buildings, parking spots, traffic signs and lights, clouds, or proximity to other objects. Detecting the occluder and occluded objects is challenging, as UAV datasets (e.g., UAV123 [2], DARPA [3]) lack depth information and object altitude data.

To manage occlusion during tracking, occlusion types are categorized as "No Occlusion", "Partial Occlusion", and "Full Occlusion", as shown in Figure 1.1 [4]. In this thesis, while "Partial Occlusion" represents the degree of occlusion, between 30% and 100%, "No Occlusion" and "Full Occlusion" are described as 0% and 100%. The occlusion is also classified as short and long-term depending on the time duration that it occurs. Long-term occlusion is harder than the short-term because of the increased uncertainty that the unknown location, kinematics and size of the target bring into the tracking problem.

Occlusion is considered one of the most difficult challenges in visual tracking and is distinct from other visual problems due to its effect on the object's appearance model update mechanism. For other visual problems, a very robust appearance update method is required to keep track of the visual model of the object. The absence of an adaptive appearance model will lead to a model drift. However, in the presence of a full occlusion state, the visual model update scheme should be stopped to keep the last known visual model of the object in the memory. If a partial occlusion is going on, a different state prediction equation and model update mechanism should be utilized to keep track of the target. So the presence of the occlusion and the degree of it bring another question into our minds; How to detect occlusion? The answer to this

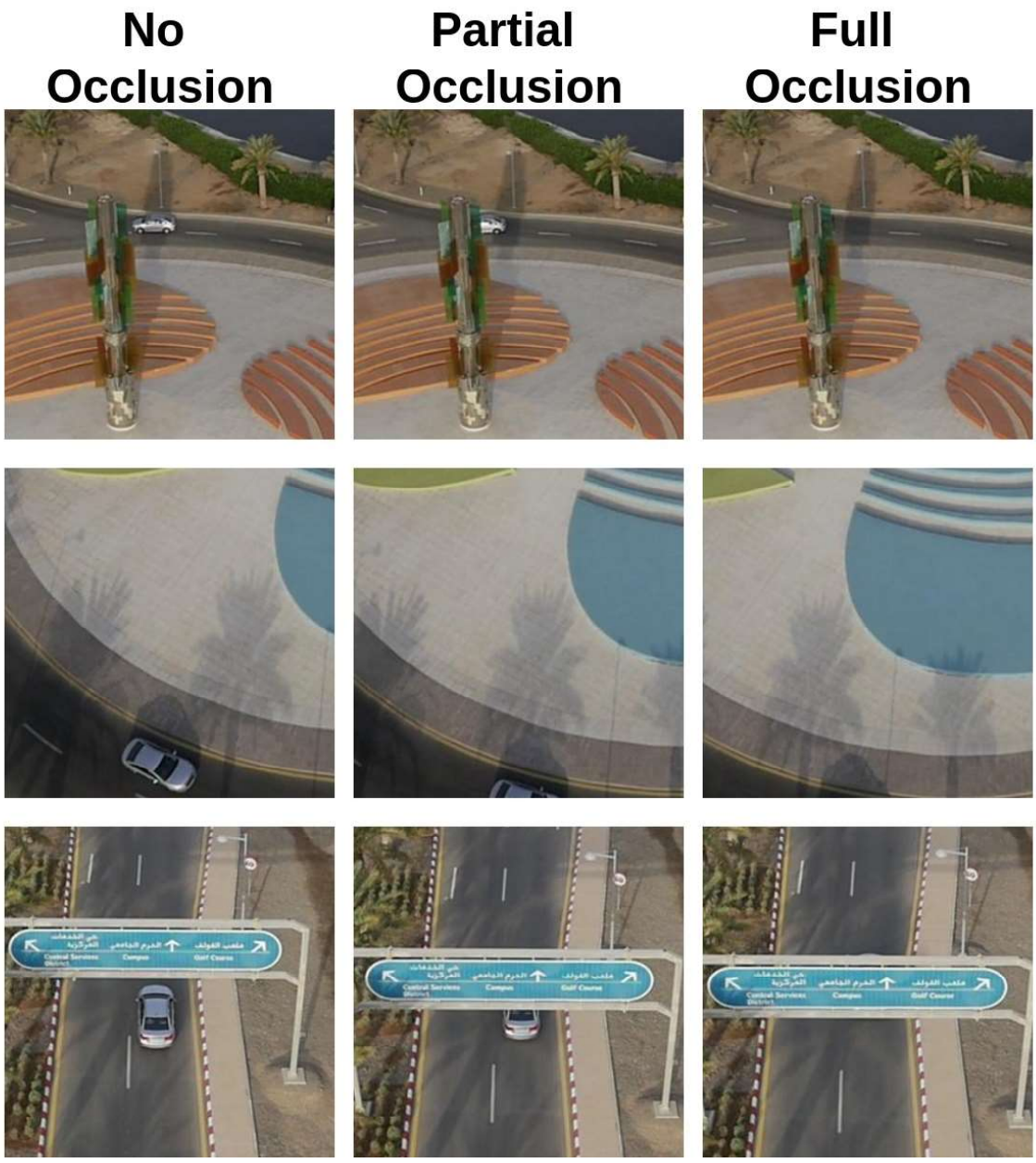


Figure 1.1. Types of occlusion.

question will determine how to approach the occlusion problem. The re-detection algorithm should be developed to foresee the possibility of occlusion. Another complex task is re-detecting and re-tracking the object after the occlusion state vanishes, as the uncertainty region where the target may reappear can vary in size depending on the object's last known dynamic motion prior to occlusion. All these challenges make occlusion the most difficult problem in visual tracking applications.

1.3. Problem Statement

The combination of the other visual problems, mentioned in Section 1.1, with occlusion makes tracking problems in UAV videos a very complicated task. In literature, the occlusion problem has been approached with different methodologies, compared to the ones that are used to solve other visual problems [4, 5]. However, the occlusion problem is still an open issue to be solved in computer vision area.

In this thesis, our goal is to design a method that can track a vehicle against all visual problems, especially occlusion, in low-altitude flying UAV videos. The complexity of the method differs based on the number of visual issues combined in a video. To create a robust method, we draw a road map with two steps.

First of all, we designed and realized a base algorithm to track vehicles against visual problems except occlusion. This method is very robust against standard visual problems; such as scale and aspect ratio variations, high dynamic motion, low resolution, and is generated via a combination of sequential importance resampling (SIR) particle filter with a convolutional neural network (CNN) model, utilized to extract features and calculate class scores of sampled objects. We call it PFCNN [6], and the robust vehicle tracking is achieved by using this structure. This is an important step in understanding the nature of visual problems before trying to solve occlusion problems. During rotation and fast motion, the aspect ratio of the vehicle changes abruptly while the motion vectors of the front and back parts of it show different directions. To be able to keep track of the diverse movement of the vehicle, PFCNN is based on multi-part tracking, and these parts represent the equally divided sections of the vehicle. In this thesis, the number of parts is accepted as 2 because of the average size of vehicles and their resolution after the division, as well as the algorithmic simplicity. Through complete integration of the particle filter with the CNN model, the weight update for each particle is supported by the class probability of the CNN model; as a result, the integrated structure resembles the APF [7–9]. To improve tracking performance against rotation and fast motion, this process is individually applied to different parts.

Variations in scale, aspect ratio, and visual model are other problems that need an online visual model update mechanism. We suggest a reliable technique that rescales every particle while sampling across the suggested region. The particle linked to the bounding box with the highest class probability is chosen based on the class probabilities of the sampled bounding boxes, which are determined by the CNN model. Then, its weight is updated and the state estimation (location, velocity, and dimension) process of the particle filter is carried out. Finally, the unification module updates the reference feature for both parts, determines the ultimate scale parameter (sP) along both axes, and unifies the final estimates of the parts.

The second step is to design a method, called the PFCNNOcc, which can detect the presence of occlusion, and the degree of it, to understand whether partial or full occlusion takes place, besides other visual problems. Since the combination of occlusion with other visual problems exponentially boosts the complexity of tracking. The proposed method requires a robust algorithm to model both vehicles and background objects to be able to discriminate them at the pixel level. To separate the vehicle from the background at the pixel level and to track the vehicle of interest under occlusion, CNN models, which are designed to find optical flow and occlusion maps, segmentation masks, and classification scores/features are re-designed, trained, and integrated with a novel algorithm, proposed in this thesis. While CNN model is trained to find optical flow vectors and occluded pixels along the forward and backward directions between consecutive UAV video frames, the segmentation method is taught to segment the pixels of a vehicle from background pixels for successive frames.

Optical flow vectors and segmentation masks play the key role in marking the occluded pixels as well as the pixels that will be seen in the next frame. They provide the exact location and the motion vectors of the vehicles, which are vital for high precision and successful tracking in UAV videos. Depending on the degree of occlusion that is detected by a novel occlusion calculator function, the proposed method switches between different tracking regimes that utilize different state estimation/prediction methods, visual model update mechanisms, and re-detection approaches as shown in

Figure 1.2. Overall degree of occlusion is named as degOfOccAll, which is tailored to decide which regime to follow during tracking. A threshold, called $TH1$, determined empirically, is tailored to decide if there is a partial occlusion or not. Below this threshold, it is assumed that no occlusion is present, and the target model is being updated. The state variables are estimated using a particle filter at the final step.

If degOfOccAll is higher than the so-called threshold, the visual model update mechanism is stopped, and the state variables are predicted using a random walk model without particle filter estimation. Depending on the predicted states, main processes, such as tracking and occlusion detection, are achieved.

If degOfOccALl is calculated as 100 %, meaning that the vehicle is fully occluded, the visual model update should be stopped because of the disappearance of the target from the scene. Under this condition, it is vital to sample the region of interest (ROIs) around the predicted location of the vehicle, calculated using a random walk model based on the state variables from the previous frame. For each ROI, it is aimed at finding the best match for the vehicle that is fully occluded or vanished in the previous frame or frames. The duration between frames where the vehicle is lost is very important to find the correct target after it emerges again. As long as the duration increases after it vanishes, the model, the location, and the speed of it can vary drastically, which can increase the burden for the re-detection or re-identification schemes. Switching from the full occlusion to the other occlusion regimes (no occlusion and partial occlusion) requires a very robust re-identification method. The occlusion regime that is switched from full occlusion depends on the reference features, especially the number of pixels and the size of the bounding box of the target, updated during the no occlusion regime before the full occlusion occurs. If a scale variation occurs during full occlusion, the transition from full occlusion to partial occlusion is also possible. During full occlusion, the re-identification module initially filters out the detected contours from the ROI, and does not satisfy the reference features of the target, updated at the end of each frame if the no occlusion regime is valid. At the final step, it calculates the confusion scores for each pair of the selected contours from

image pairs by calculating the multiplication of the cosine distance between the RGB and HSV color features with the Euclidean distance between the estimated state from the previous iteration and the state estimated/predicted in the present iteration. The contour pair with the maximum confusion score is the best contour that represents the tracked target after the occlusion ends.

Hence, deciding the presence and the degree of the occlusion as partial or full occlusion during vehicle tracking, and switching between these regimes, is very complicated, depending on its duration, and this issue constitutes the main problem in this thesis.

1.4. Overview of the Proposed System

The proposed system, PFCNNOcc, is designed to achieve a robust target tracking on low-altitude flying UAVs against all visual problems, especially against short or long-term partial and full occlusion coupled with aspect ratio change, background clutter, camera motion, fast motion, illumination variation, low resolution, out-view, similar object, scale variation, viewpoint change. The PFCNNOcc combines CNN models, trained with the novel synthetic S-UAV dataset [10, 11] and the UAV123 dataset [2] to predict the optical flow and occlusion maps, segmentation masks, and classification scores. The overview diagram of the PFCNNOcc is shown in Figure 1.3, and its novel functions are named as;

- Re-Identification Module Final PFCNNOcc,
- Occlusion Detector and Calculator Module Final PFCNNOcc,
- Full Occlusion Dense Search Module Final PFCNNOcc,
- Merged Target Detection.

These are the contributions of the PFCNNOcc to the literature. As far as is known that the PFCNNOcc is the first method in literature that predicts and utilizes the optical flow and segmentation to estimate the state variables of the target as well as the

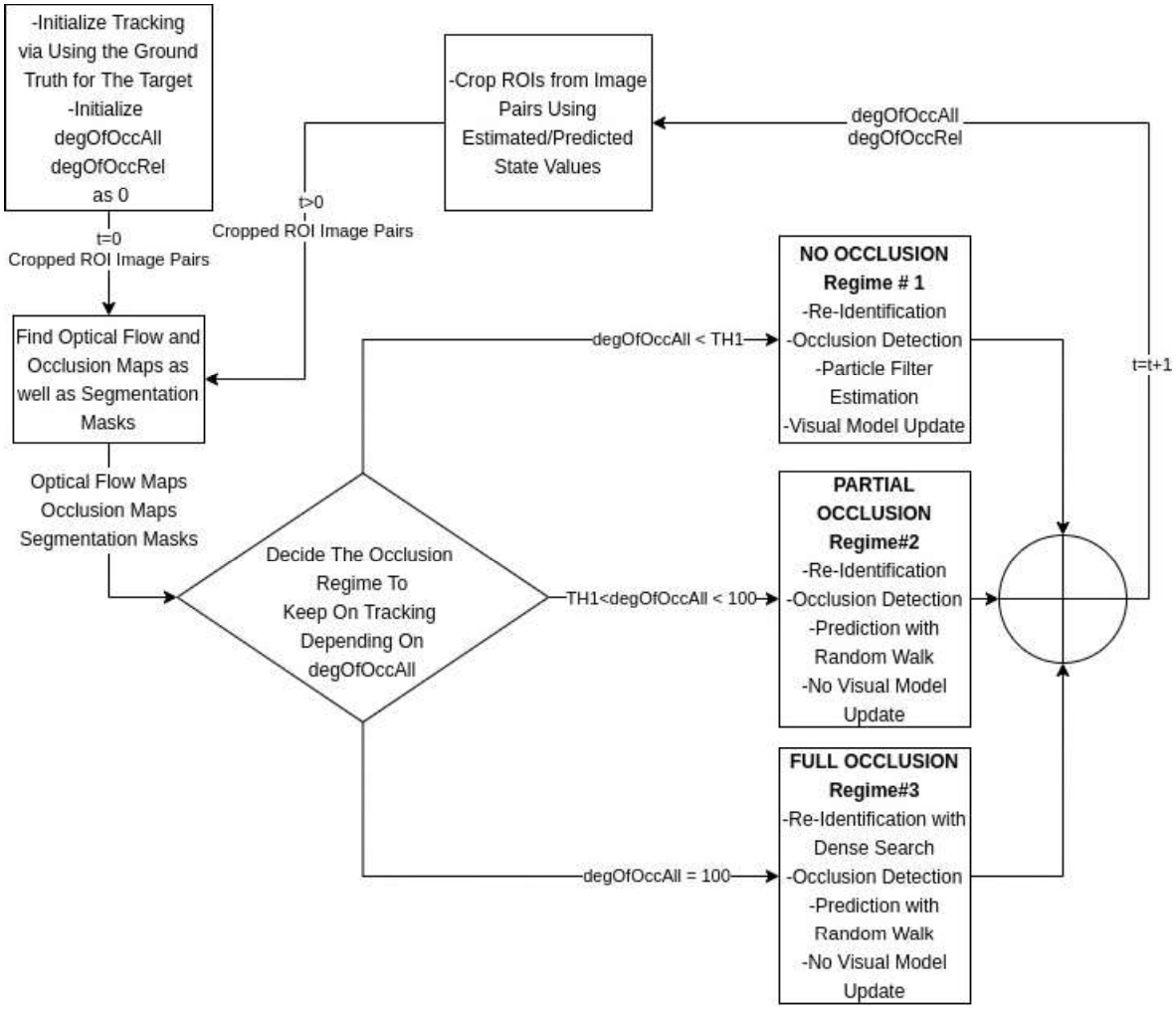


Figure 1.2. The general concept of the proposed method using different occlusion detection and state estimation/prediction mechanisms.

occlusion state and the degree of it during tracking in UAV videos. "Re-Identification Module Final PFCNNOcc" utilizes the forward and optical flow predictions, and the segmentation masks of the consecutive images with a novel matching algorithm to re-detect the target at the beginning of each image pair and to localize its location and size at pixel-level with high precision. In the meantime, the average motion vector for the target is also calculated.

The location, size, and motion vector, calculated with high accuracy pave the way to boost the tracking performance of the final version of the PFCNNOcc. "Occlusion Detector and Calculator Final PFCNNOcc" is very vital to decide whether a pixel is non-occluded, will-be-occluded, and already-occluded, and to calculate the degree of

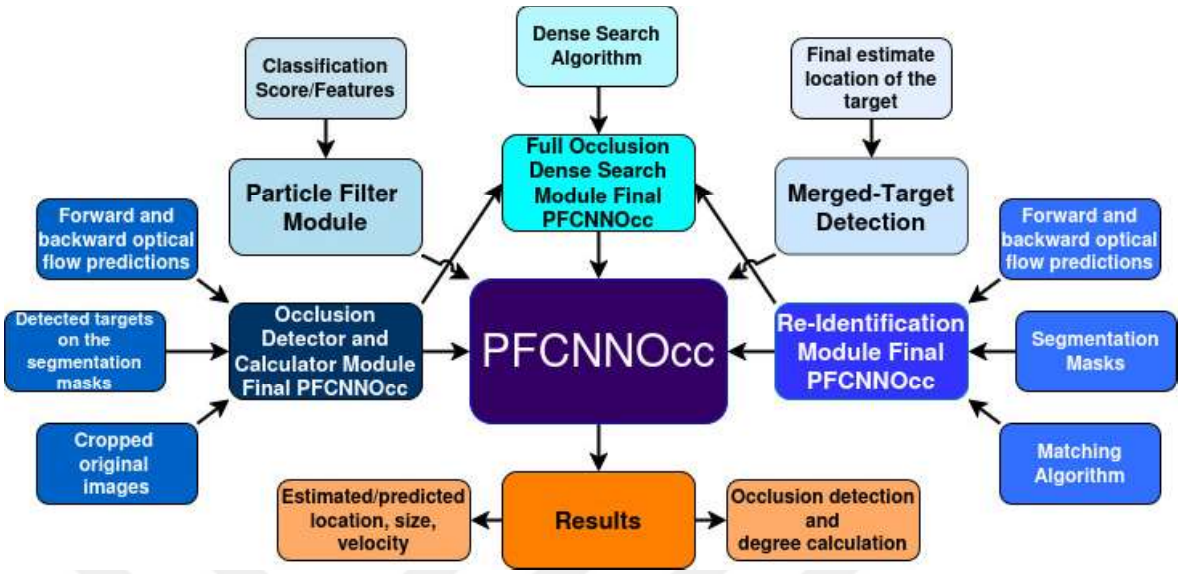


Figure 1.3. The overview diagram of the PFCNNOcc.

occlusion via considering the possible scale and aspect ratio variations. This novel function increases the robustness of the method against partial and full occlusion, and hence, it makes a huge difference compared to the state-of-the-art (SOTA) trackers. "Full Occlusion Dense Search Module Final PFCNNOcc" is a combination of the "Re-Identification Module Final PFCNNOcc" and "Occlusion Detector and Calculator Final PFCNNOcc" with a dense-search algorithm to re-detect the target after a short or long-term occlusion. Merged target detection function at the end of iteration detects whether a target is occluded by another vehicle during tracking or not. After the occlusion or merged-target condition ends, the discrimination capability of the function, separating the target from other vehicles on the run, helps the PFCNNOcc to keep on tracking precisely.

1.5. The Contributions

The novelty of the methods proposed in this thesis and the contributions are presented in the following paragraphs.

In this thesis, we developed two novel methods called the PFCNN and the PFCNNOcc. While the PFCNN focuses on resolving abrupt changes in scale, aspect ra-

tio, visual model, target motion, and rotation except occlusion for vehicle tracking in low-altitude UAV videos, PFCNNOcc is capable of tracking a vehicle under all visual problems, especially partial and full occlusion, which is not addressed by the PFCNN.

The contributions of the PFCNN [6] are threefold:

(i) Particle rescaling mechanism against scale and aspect ratio variation: By introducing a particle rescaling mechanism where each particle is resized with different aspect ratios, tracking under scale and aspect ratio variations is improved.

(ii) A new structure, obtained by fully integrating the CNN model with the particle filter: By this new structure, APF [7–9] is mimicked. By supporting the weight update process of the particle filter with class probability scores of the particles improves the state estimation.

(iii) The unified multi-part vehicle tracking framework: Forming such a tracking mechanism enables several tracking agents to act independently and to provide estimates during the particle sampling phase. The novel unification module improves the standard particle filtering performance against variations in scale, aspect ratio, target motion, and visual model, and puts the PFCNN into the top-ranking SOTA trackers [1], [4], and [12, 13].

The PFCNNOcc method, designed in the second step of the thesis, outperforms all SOTA trackers in the UAV123 10fps (frames per second) dataset [2]. It combines CNN models, trained with the novel synthetic S-UAV dataset [10], [11] and the UAV123 dataset [2] to predict the optical flow and occlusion maps, segmentation masks, and classification scores. The novel functions, named as;

- Re-Identification Module Final PFCNNOcc: It utilizes the forward and backward optical flow predictions, and the segmentation masks of the consecutive images with a novel matching algorithm to re-detect the target at the beginning of each image pair and to localize its location and size at the pixel level with high precision. In the meantime, the average motion vector for the target is also calculated.

The location, size, and motion vector, calculated with high accuracy, pave the way to boost the tracking performance of the final version of the PFCNNOcc.

- Occlusion Detector and Calculator Final PFCNNOcc: It is very vital to decide whether a pixel is non-occluded, will-be-occluded, and already-occluded, and to calculate the degree of occlusion by considering the possible scale and aspect ratio variations. This novel function increases the robustness of the method against partial and full occlusion, and hence, it makes a huge difference compared to the SOTA trackers.
- Full Occlusion Dense Search Module Final PFCNNOcc: It is a combination of the "Re-Identification Module Final PFCNNOcc" and "Occlusion Detector and Calculator Final PFCNNOcc" with a dense-search algorithm to re-detect the target after a short or long-term occlusion. The overall performance of most of the SOTA trackers decreases because of the full occlusion condition, creating a huge uncertainty in terms of the location and size after the target vanishes. The amount of this uncertainty is a cumbersome for the trackers to handle after the target is appeared on the scene again. Hence, "Full Occlusion Dense Search Module Final PFCNNOcc" has the key contribution in the best performance of the PFCNNOcc.
- Merged Target Detection: It detects whether or not a target is occluded by another vehicle during tracking at the end of each iteration. After the occlusion or merged-target condition ends, the discrimination capability of the function, separating the target from other vehicles on the run, helps the PFCNNOcc to keep on tracking precisely.

These are the contributions of the PFCNNOcc to the literature, and will be explained in detail in Chapter 5. As far as is known that the PFCNNOcc is the first method in literature that predicts and utilizes the optical flow and segmentation to estimate the state variables of the target as well as the occlusion state and the degree of it during tracking in UAV videos.

Another contribution of this thesis is the creation of the S-UAV dataset [10], [11] to train CNN models, used to predict the optical flow maps in correlation with the occlusion maps, as well as segmentation masks and classification scores. The vital importance of this synthetic dataset published in [10], is to construct the ground truth for the optical flow and occlusion data that are almost impossible to extract for real-time scenarios. To the best of our knowledge, S-UAV dataset [10] is the first and only UAV dataset that has these ground truth data at the same time, especially the optical flow and occlusion. While these ground truth data in the S-UAV dataset boost the performance of the PFCNNOcc method to outperform the SOTA trackers, the ground truth data for the segmentation and classification are also utilized to train the SOTA detection algorithms in [11] to enhance their performances.

One major contribution is presented through the development of a tracking-by detection framework using the Marginal Particle Filter (TDMPF) [20], aimed at validating the effectiveness of particle filtering for visual object tracking. This method introduces a technique for tracking objects exhibiting nonlinear dynamic motion. Moreover, the variance of the noise in the process equation of the particle filter is recursively estimated, enhancing the system's robustness against sudden dynamic movements of the object and reducing tracking errors in videos with continuously changing backgrounds.

To constitute the basis of the novel PFCNN [6] and PFCNNOcc methods, one major contribution is presented through the development of a tracking-by-detection framework using the Marginal Particle Filter (TDMPF) [14], aimed at validating the effectiveness of particle filtering for visual object tracking. This method introduces a technique for tracking objects exhibiting nonlinear dynamic motion. Moreover, the variance of the noise in the process equation of the particle filter is recursively estimated, enhancing the robustness of the system against sudden dynamic movements of the object and reducing tracking errors in videos with continuously changing backgrounds.

1.6. Outline of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, the visual object tracking methods and those designed to solve the occlusion problem in the literature are discussed. In Chapter 3, the PFCNN is elaborated, and is compared with APF. In Chapter 4, the creation of the novel synthetic UAV dataset, called S-UAV [10], and the segmentation ground truth for the UAV123 dataset [2] is explained in detail. In Chapter 5, the proposed method, PFCNNOcc, is explicated in detail. In Chapter 6, a comprehensive analysis is carried out separately for the PFCNN, the APF, and the PFCNNOcc as well as it ablation versions to find the best-performing method among the SOTA trackers. The analysis consists of scenario-based, attribute-based, and overall comparisons and summarizes the pros and cons of the methods. Finally, in Chapter 7, concluding remarks based on the analysis covering the SOTA methods are presented, and the possible areas of future work are summarized.

The figures created within the scope of this thesis and copyrighted to the publisher are used in the thesis in accordance with the “the editorial policy of the author on the reuse of self-produced text and figures”on the website of the publisher.

2. LITERATURE SURVEY

This chapter examines discriminative approaches in visual object tracking techniques. Additionally, deep learning (DL) techniques are categorized among these approaches based on the structure of their CNN models and their application. To provide insight into how optical flow information and segmentation are customized to identify occlusion, a survey of the most advanced optical flow and occlusion detection techniques created with DL models is conducted. For better understanding, the literature survey will be presented in two different sections. While the first section explains the SOTA discriminative methods in visual object tracking, the second one concentrates on the techniques, especially focusing on visual object tracking under occlusion using optical flow, segmentation, and other techniques. In [1], both generative and discriminative target tracking algorithms are investigated extensively, and their respective advantages and disadvantages are outlined through comparative analysis. Additionally, it details the databases and evaluation metrics commonly used in the literature for assessing performance.

2.1. State-of-the-Art Discriminative Methods Visual Object Tracking

2.1.1. Tracking by Handcrafted Feature-Based Methods

To identify the discriminative function that learns both the target and the background, the trackers in this class monitor the target while labeling the candidate bounding boxes that are sampled using techniques like Adaboost and positive-negative learning [1], [15].

To accomplish visual target tracking, tracking with learning detection (TLD) [16] combines a binary-classifier learning approach with the Lucas Kanade tracking algorithm [17]. The relationship between discrete Fourier transform and circulant matrices is taken into consideration in order to establish a high-speed tracking-by-detection

process while utilizing the circulant structure of tracking-by-detection with kernels (CSK) [18] and kernelized correlation filter (KCF) [19].

Correlation filters for potential translation and scale variation are computed separately in the discriminative scale space tracker (DSST) [20]. To address the pitfalls resulting from periodic assumptions, a spatial regularization parameter is incorporated in the discriminative correlation filter (DCF) formulation in spatially regularized discriminative correlation filters (SRDCF) [21]. The model updating procedures and the sampling technique of the target, surrounding the ROI, are the drawbacks of the methods. An abrupt shift in visual appearance is accompanied by changes in the scale/aspect ratio of the target, sudden camera movements, or strong dynamic motion with extreme maneuvers.

In TDMPF [14], an algorithm that utilizes a tracking-by-detection approach to monitor objects exhibiting non-linear dynamic motion is designed. This method incorporates a tracking-by-detection technique based on the Gauss kernel, which leverages circulant matrix theory and the Fourier transform, in conjunction with the marginal particle filter method. The marginal particle filter applies scores obtained from the Gauss kernel during the measurement update phase to assign weights to particles that are propagated around the tracked target. During the state variable updates by the marginal particle filter, estimates are made for the coordinates of the objects, their associated correction values, and the dimensions of the image window surrounding the object. This study constitutes the basis for the PFCNN [6] and the PFCNNOcc which are the main contributions of this thesis.

There are methods, designed based on the updating rule of the classification function about changes in the tracking environment. The tracking is achieved by using support vector machines to train the target model and the background. Instead of sampling and labeling independently, the kernelized and structured support vector machines in structured output tracking with kernels (STRUCK) [22, 23] are designed to achieve tracking by detecting transformation between consecutive images.

Only 2D motion variables are estimated in [22], but scaling is also computed in [23]. Robust tracking with multiple experts using entropy minimization (MEEM) [24] method mitigates the drift or model deviation issue during tracking. These techniques have trouble tracking if the target exhibits rapid dynamic motion and abrupt visual model changes. According to this scenario, they need a more sophisticated sampling technique and a more robust model update structure.

2.1.2. Tracking by Deep Feature Based Methods

Only the most advanced tracking techniques [4] that use DL to achieve visual object tracking are described in this section.

2.1.2.1. Tracking-by-Detection Based Methods. Visual tracking is seen as a target and background classification problem by tracking-by-detection techniques. By sampling numerous bounding boxes surrounding the potential target area in the image, the deep networks used in this class of algorithms to learn and to detect the target in the proposal region.

A CNN design is suggested in the multi-domain network tracker (MDNet) [25] to learn the shared properties of potential targets from several annotated image sequences, or "domains." Additionally, the network contains a number of domain-specific layers that function as distinct branches and are utilized to learn features for binary classification of various image sequences. The weights learned from each image sequence in the preceding layers for generic representation learning are shared by these levels. Additionally, during the testing phase, the model is coupled with an online learning system. To do this, all domain-specific categorization branches are eliminated, leaving only one branch to be used for testing score estimation. To acclimate to the new domain, the fully-connected layers within shared layers and the new categorization branch undergo online training.

A reciprocative learning method that customizes attention maps is suggested in deep attentive tracking (DAT) [26], where MDNet [25] is used as a baseline tracking-by-detection framework. The classification score is determined in the forward pass when an input image is fed into the deep CNN (DCNN) architecture. The reciprocative function is applied to the classification score in order to determine its derivative with respect to the input image along the path from the last fully-connected layer towards the first convolutional layer, prior to applying the traditional backward propagation to the loss function during training. After that, the output of the first layer of the network—the attention map—is extracted. No network parameter is changed while the attention map is being calculated. The traditional backward propagation is implemented to update the network parameters once these attention mappings are applied as a regularization term to the loss function. Thus this technique learns to focus on the desired areas and remove background pixels.

2.1.2.2. Correlation Filter Based Methods. Finding the target's location within a patch of an image by learning the DCF is the main goal of correlation filter-based techniques. Typically, an ROI that is larger than the bounding box of the target is cropped, and any feature extraction approach is used to compute the features. In order to create a Gaussian response map in relation to an ideal map, these features are then customized to learn several correlation filters. The resulting response map is then used to estimate the bounding box of the tracked object in the last phase.

Due to the lack of a target model update mechanism, these approaches suffer from significant tracking issues if the monitored object experiences partial or complete occlusions and deformations. Consequently, it is crucial to include a regularization factor in the loss function used for filter learning.

Using multi-resolution deep feature maps, the continuous convolutional operator tracker (CCOT) [27] learns the convolutional filters customized in [18, 19] in the continuous spatial domain. While it formulates the learning issue in the continuous spatial domain using an implicit interpolation approach, it has the ability to localize

sub-pixels, which is essential for the task of precise feature point tracking. Efficient convolution operators for tracking (ECO) [28] evolves from CCOT [27] by altering the model updating mechanism and integrating a factorized convolution operator to reduce the number of convolutional filter parameters. To achieve so, an operator for factorized convolution, which significantly lowers the model's parameter values, and a conservative model update technique characterized by enhanced resilience and less complexity are developed.

By including temporal regularization, the spatial-temporal regularized correlation filter (STRCF) [29] enhances SRDCF [21]. SRDCF [21] integrates an extra spatial penalty on convolutional filter coefficients to address the boundary effect issues encountered in [18, 19]. The tracking performance is enhanced by this technique, but the computational load is also increased in return. STRCF, which is produced by adding temporal regularization to SRDCF [21], can be regarded as a logical approach to SRDCF with multiple training images. However, depending on the level of visual fluctuation in the scene, STRCF [29] provides a more reliable appearance model than SRDCF.

A research that is comparable to proposed PFCNN [6] is the deep convolutional correlation iterative particle filter (D2CIP) [30]. D2CIP is an extension of its prior visual trackers [31–34]. It combines Sequential Monte Carlo procedures with correlation-convolution approaches. It incorporates several particles as inputs to a CNN and uses the correlation filter from the ECO tracker [28] to build correlation maps for each particle. The new location of the target is determined by analyzing the correlation between the target model from previous frames and the convolutional features retrieved from the current frame. If the distance between the peak of the correlation response map and the particle position exceeds a given threshold, a new response map is generated using the picture patch centered at the peak. As this process repeats, most particles end up in a limited number of places. D2CIP iteratively updates particle likelihoods to calculate the posterior distribution over the proper support points in particle-correlation trackers. However, there are several significant variations be-

tween the PFCNN and D2CIP. Our approach customizes parts-based tracking, particle rescaling, and updating the weights via mimicking APF and unification algorithms for optimal performance, whereas D2CIP uses only correlation filters to refine the sampled particles. While D2CIP only uses the output of the correlation maps to fine-tune the location of the particles, the PFCNN updates the weights of the particles using the likelihoods and the class probability scores of these particles. The PFCNN also fine-tunes the location and size of the bounding boxes of the particles in particle rescaling and unification functions. Hence, the PCFNN evolves not only the estimates for the location but also the size of the bounding boxes for each particle. The primary issues with these techniques arise when there is rapid target motion combined with variations in scale, aspect ratio, and visual model.

2.1.2.3. Siamese Network-Based Methods. Using a similarity learning strategy, Siamese networks, which are made up of two identical components with shared weights, are utilized to learn the common representation of the input images. These trackers have garnered a lot of interest because of their promising performance [35]. The trackers have two main approaches to sample the region: 1. Anchor-based, 2. Anchor-free. While anchor-based methods tailor the region proposal network (RPN), anchor-free ones rely on the four sides of the bounding box for sampling and training.

On the other hand, a DCNN is trained offline to determine the similarity between a target image and a larger search image in the fully convolutional Siamese tracker (SiamFC) [36]. The target image is tracked within the search image via the Siamese architecture of DCNN. The fully-convolutional structure of this Siamese network allows end-to-end tracking, and a bilinear layer specifically designed for the cross-correlation of the target and search images performs the sliding-window function.

Following the successes of SiamFC [36], SiamRPN [37] offers a region proposal network and resolves the tracking issue as a one-shot local detection procedure. SiamRPN is the first method that applies the RPN to visual tracking problems via tailoring predefined anchors [35]. It comprises a Siamese subnetwork for feature extraction and

a region proposal subnetwork, which includes classification and regression branches. One of the primary issues with Siamese trackers is their inability to distinguish the target from purely non-semantic background, or background with a different textural pattern from the target, under distractor-aware Siamese region proposal networks (DaSiamRPN) [38]. When the background is similar to the target, it is referred to as a distractor. In such cases, the tracking process becomes more complex. The lack of an online update method for the target model is another issue with contemporary Siamese trackers. DaSiamRPN [38] suggests a distractor-aware Siamese network for precise and long-term tracking in order to address these issues with current Siamese trackers. Positive picture pairs that resemble the goal are exploited during the training process, but negative image pairs that resemble the semantic backdrop (distractors) are also produced. By including these distractions in the training process, DaSiamRPN is able to block them while tracking in real time.

In SiamAPN [35], an anchor proposal network (APN) is proposed in an adaptive fashion as anchor-free method. Instead of using pre-defined anchors, the utilization of adaptive anchors increases the probability of including objects with different sizes inside the bounding box. This feature decreases the unbalance of the images from the object and background. The hybrid dimension features from APN and feature extraction networks are integrated into a clean feature fusion network. A complex multi-classification framework is suggested to take into account fully classifiable elements to improve SiamAPN's discriminability [35].

To solve the semantic information variation problem in SiamAPN [35] via using the attention mechanism, an attentional Siamese tracker, called as SiamAPN++ [39], was proposed for real-time UAV tracking. SiamAPN++ is designed with the integration of the attentional aggregation network (ANN) and anchor proposal network depending on dual features (APN-DF). In order to adaptively aggregate the cross-interdependencies from various feature maps and the self-semantic interdependencies of a single feature map through two methods, a unique AAN is presented. By implementing the dual features in APN-DF, the suggested anchors' anti-interference and

resilience in the face of significant scale fluctuation are enhanced. The methods in this class, especially DaSiamRPN [38], SiamAPN [35], and SiamAPN++ [39], achieve the best scores or performances in the SOTA [4], [35], [38,39] for the tracking on UAV20L, UAV123@30fps, and UAV123@10fps from the UAV123 Benchmark Dataset [2].

2.1.2.4. Reinforcement-Learning Based Methods. In reinforcement learning based methods, the tracker learns the most feasible translational parameters between the search samples and the target. They exploit the feedback to calculate these parameters. After each observation is input to these methods in discrete time steps, they select the best action among several feasible alternatives. This iteration is repeated until convergence is achieved [4]. In actor-critic tracking (ACT) [40] method, the actor-critic framework is taken as a reference to design a learning-based approach that uses uninterrupted actions. At each iteration, one action is taken by the "actor" network to predict the possible location of the target in the image. Then, the reinforcement learning idea forces the method to acquire the best optimal action for the best target estimate during training. "Critic" network acts as a confirmatory agent during both offline training and online tracking [40].

2.2. Visual Object Tracking Under Occlusion

The tracked target becoming partially or fully invisible on the LOS of the video camera because of other moving objects or background objects in the environment is known as the occlusion problem.

In this section, the methods tailored to detect occlusion will be summarized. And optical flow is one of the best methods, commonly used in the literature for detecting and determining the degree of occlusion. It is commonly known that two-dimensional image coordinates are also affected by the relative movements of targets in three-dimensional space.

In order to calculate the magnitude and direction of the vectorial changes displayed by the targets, techniques have been developed in the literature to identify brightness variations between the two images. This allows for the creation of two-dimensional displacement field vectors, also referred to as optical flow vectors [41].

The optical flow field is generally best described as a dense field of displacement vectors that translates every point in the first image onto its matching location in the second image. Numerous computer vision domains, including visual surveillance, robot navigation, image interpolation, and super-resolution, use optical flow [41]. Being able to identify and follow an object is essential for visual surveillance, as is the ability to do depth ordering to comprehend the potential for occlusion. A crucial tool for completing this work is optical flow, which extracts the scene's motion vectors.

Another extremely helpful feature of optical flow is its application in robot navigation, which it is used to determine the speed and direction of the robot and to address issues with obstacle recognition and collision avoidance systems.

The application of optical flow in picture interpolation and super-pixel resolution is another crucial topic. The Super-Pixel resolution technique is designed to improve the resolution of low-resolution (LR) photos to high-resolution (HR) images, while image interpolation is used to raise the frame rates and resolution of films as post-processing. Additionally, optical flow is essential for aligning two pictures or video frames with differing resolutions.

The earliest optical flow approach, known as the Brightness Constancy Assumption (BCA) was created by the classical work of Horn & Schunck (HS) [41], and relies on the supposition that brightness remains constant for a brief period between two frames, and it is illustrated as

$$f(x, y, t) = f(x + dx, y + dy, t + dt) \tag{2.1}$$

where x and y stand for the location of the pixel along x and y axeses in the frame, f, at time t, dx, dy, and dt present the displacements for the location and the time.

After the classical work of HS [41], many different optical flow methods have been developed in the literature, and these methods can be categorized into two broad categories: 1. Traditional Methods, 2. DL Based Methods [42,43].

2.2.1. Traditional Optical Flow Methods

In this section, the phylosophy behind the traditional optical flow methods will be explained under three different subcategories: 1. Variational Methods, 2. Patch-Based Methods, 3. Patch-Based Methods with Variational Refinement.

2.2.1.1. Variational Methods. All the methods [1], [12,13], [15] in this category calculate the optical flow vectors by minimizing an energy function or cost function. One of the most effective and earliest variational methods is HS [41], which exploits the BCA, thus considering that the horizontal and vertical displacements are small. The BCA equation, given in Equation (2.1), is linearised using Taylor series expansion. All variational methods aim to minimize an energy function similar to the one in HS, but they can exploit different data types, such as gradient constancy (Scale Invariant Feature Transform (SIFT) features) in [44], higher order derivatives, and color model.

2.2.1.2. Patch Based Methods. Most of the patch-based methods [1], [12,13], [15] rely on Lucas Kanade (LK) [45] approach, which works on the discrete domain. Two consecutive frames are divided into fixed-size patches selectable by the user, and these patches are matched by minimizing the intensity gradient of the image. LK is robust against small motions between two consecutive images but is prone to error if a large displacement occurs in the scene. To solve this problem, a coarse-to-fine strategy is integrated into the LK algorithm, which uses the image pyramids, downsampled from high-resolution images. At the top, there is the coarsest resolution image.

2.2.1.3. Patch Based Methods with Variational Refinement. The difference between these methods from classical patch-based algorithms is how they refine the energy function, which is used to minimize a gradient, color model difference, etc. to find the optical flow. For example one of the first methods of this kind is Large Displacement Optical Flow (LDOF) [44], which puts an additional energy term depending on the SIFT features and color into the total energy equation as another constancy. Lots of methods based on LDOF have a feature extractor, matching, and variational refinement stages. Another very effective patch-matching algorithm from LDOF is Deep flow [46]. This is a variational optical flow method with an additional loss function based on a deep matching algorithm with a classical variational framework that allows integration of feature descriptors and matching. Deep Flow is a non-parametric algorithm based on a fine-to-coarse approach, and its most innovative part is its deep matching algorithm, which is used to extract descriptors and match them in deep. The deep matching algorithm is tailored by other very high-performance methods with different variational refinement algorithms.

2.2.2. DL-Based Optical Flow Methods

Deep flow [46] is one of the best-performing hand-crafted optical flow approaches, similar to DL methods because of its convolutional structure but with no learned parameters. Hence, deep flow opened a new era in which DL methods are being applied to find optical flow in different ways.

As a consequence, in recent years, CNN-based methods have been widely used to find optical flow. In some of these methods, CNNs are used for extracting deep features for image comparison, while in others they directly generate dense optical flow vectors end-to-end.

CNN-based methods, tailored for optical flow, are fundamentally divided into two main classes: 1. Supervised Learning, 2. Unsupervised Learning.

2.2.2.1. DL-Based Optical Flow Methods with Supervised Learning. A sizable optical flow training dataset is necessary for the CNN model to learn the optical flow in supervised learning-based techniques to provide an accurate reference or ground truth. The effectiveness of the supervised learning-based approach is directly impacted by the diversity and accuracy of the training dataset. Nevertheless, it is extremely challenging to label optical flow in real-time recorded movies, and nearly all of the techniques used in the literature are trained on artificially created datasets that include labeled optical flow. The following lists the relevant synthetic datasets. Flying Chairs [47] is a synthetic database with 22,000 image pairs and corresponding ground truth optical flow data. ChairsSDHom [47] is a synthetic database consisting of image pairs and corresponding ground truth optical flow data. FlyingThings3D [48] is a synthetic database with image pairs and corresponding ground truth optical flow data. Sintel [49] contains 1,000 image pairs taken from open-source videos. It is preferred for fine-tuning and evaluation of images with dense optical flow vectors. Monkaa [48] consists of 8,500 image pairs extracted from a cartoon and is similar to the Sintel database. KITTI2012 [50] is constructed with 200 frames of stereo videos of road scenes from the real world. Some of the synthetic datasets found in the literature [51–54] are derived from the Grand Theft Auto 5 (GTA5) game. However, the optical flow results presented in these studies are not provided from the perspective of a drone, and the number of labeled images and vehicle diversity are insufficient. This is because it is not possible to access the mesh information from a certain distance from the camera in the GTA5 world. The datasets [55] and [56] were created using the Unity engine. The Crowdflow [55] dataset includes optical flow information along with a human crowd, and it has been captured from the perspective of a drone.

The success of DL leads to the development of novel optical flow models. [46] computes optical flow using patch matching by extracting discriminative features from deep nets. [57] adds more semantic information to the patch matching-based approaches, extending them even further. A robust thresholded hinge loss is later proposed in [58] for Siamese networks to learn CNN-based patch matching features. [59] produces optical flow results quickly and accurately while speeding up the processing of the patch

matching cost volume. FlowNet [47] employs an end-to-end learning strategy to forecast optical flow via training a network to predict the x–y flow fields straight from the images using a dataset of image pairs and ground truth flows. Stacking both input images together and passing them through a very general network is a straightforward option that allows the network to determine how to process the image pair in order to extract the motion information. This architecture is built only with convolutional layers, and is called "FlowNetSimple(S)". A less generic and hand-designed model is created to guarantee learning with better performance without creating a larger model. The network is limited by this architecture so as to first generate meaningful representations of the two images independently before combining them at a higher level via introducing a "correlation layer". The usual error metric for optical flow estimation, end-point-error (EPE), is employed as the training loss. It is the average of Euclidean distances of all pixels between the ground truth and the anticipated flow vector. Hence, FlowNet [47] signified a change in perspective for estimating optical flow. The notion of directly learning the concept of optical flow from data using a basic CNN architecture is entirely at odds with all of the accepted methods. This work demonstrates that a generic U-Net CNN architecture may be used to directly estimate optical flow from raw images [60].

Introducing FlowNet2 [61] fixes issues with noisy artifacts and tiny displacements in estimated flow fields by FlowNet [47]. This results in a significant boost in performance for practical uses like motion segmentation and action detection. FlowNet2 also presents a warping operation and demonstrates how employing it to stack several networks can greatly enhance the outcomes. But, its subnetworks must be trained one after the other since large models are more likely to experience the over-fitting issue. Because of its size, required by the memory during training, it is not a good fit for embedded and mobile devices.

SpyNet [62] uses DL in conjunction with a traditional spatial-pyramid formulation to learn how to compute optical flow. By calculating an update to the flow and warping one image of a pair at each pyramid level by the current flow estimate, big motions

are computed in a coarse-to-fine strategy. An advantage of the pyramid is that, unlike the more current FlowNet [47] technique, the networks do not have to handle massive motions. In order to estimate the motion from these two images, SpyNet just needs a small network. Hence, SpyNet demonstrates how classical principles and CNNs can be combined so that a small network can do the motion estimation. EPE loss is used during training.

LiteFlowNet [63] is made up of two small sub-networks that specialize in optical flow estimates and pyramidal feature extraction. Any pair of images can be converted into two pyramids of multi-scale, high-dimensional characteristics with a pyramidal feature extractor, called NetC. A flow estimator, such as NetE, is composed of regularization and cascaded flow inference modules to estimate coarse-to-fine flow fields. This network innovates the practical aspects of traditional techniques. While image warping is accomplished via the feature warping function of CNN, and brightness constraint in data fidelity is considered by the pyramidal features in CNN. This compact network outperforms FlowNet [47] and FlowNet2 [61] while having a smaller model size and faster iteration time. Each network at a different level has independent CNN parameters to train without sharing with each other. Therefore, this increases the number of trainable parameters to decrease the computation load during training. EPE loss is tailored during training as applied in FlowNet [47], [61].

SegFlow [64], an end-to-end trainable network, is designed to predict optical flow and pixel-wise object segmentation in consecutive images or videos at the same time. The two branches of the method allow for the bidirectional propagation of important information about object segmentation and optical flow inside a single, cohesive framework. The optical flow branch utilizes the FlowNet model, whereas the segmentation branch is based on a fully convolutional network, which has demonstrated efficacy in image segmentation tasks. To learn a general concept, the unified framework is iteratively trained offline. It is then refined online for particular objects. The loss function is constructed with the combination of EPE loss for the optical flow network and a categorical cross-entropy loss for segmentation network.

PWC-Net [65] and PWC-Net+ [66] close the gap between the traditional optical flow methods and CNN. Initially, traditional optical flow techniques frequently preprocess raw images to extract features that remain invariant to shadows or illumination variations. Moreover, in the specific context of matching, a cost volume serves as a more discerning depiction of disparity (1D flow) compared to unprocessed images or features. Instead of applying cost volume over the full frame, a partial search area at each pyramid level decreases time consumption for the process. PWC-Net is developed to benefit all these basic and well-designed realities, such as pyramidal processing, warping, and cost volume usage, to improve the optical flow accuracy in real-time scenarios. PWC-Net warps the CNN features of the second image using the current optical flow estimate, which is cast in a learnable feature pyramid. After that, it creates a cost volume using the features of the initial image and warped features, which CNN processes to estimate the optical flow. A very important point in its architecture is that, rather than using the same parameters, the CNN estimators, at various levels, have their own. This method uses the same multiscale training loss, tailored in FlowNet [47].

Each of these approaches requires a sizable optical flow dataset for training, and their CNN models differ. Furthermore, their error functions vary concerning the elements that are intended to be improved during CNN training. Photometric or smoothness terms, or a combination of both, make up the error functions. The development of CNN-based optical flow methods starts with the design of FlowNet [47]. Then, the other CNN-based methods are developed depending on the baseline CNN architecture in FlowNet [47] via improving the accuracy and computational speed with the addition of pyramidal processing, warping, and cost volume usage, weight sharing between the levels of the pyramid, etc. These new concepts boost the performance of the CNN-based optical flow estimators while keeping the basic principles from the traditional methods.

2.2.2.2. DL-Based Optical Flow Methods with Unsupervised Learning. The primary drawback of DL techniques is their voracious appetite for labeled data because supervised learning-based optical flow estimator models need dense labels with per-pixel

motion between two consecutive frames. At this point, unsupervised learning-based CNN techniques are being developed to reduce the dependence of supervised learning-based methods on labeled optical flow datasets. In unsupervised learning-based CNN methods, the photometric error function between the warped image and the first image is iteratively optimized after the second image is warped to the coordinate space of the first image using the optical flow data that the CNN has gathered. At this point, the unsupervised DL approaches differ from the supervised DL methods, using precise references to train the network.

There are advanced unsupervised learning methods in the literature. DST-Flow [67] begins with a localization net utilizing the FlowNet structure [47] given two input images. Pixel-level translation estimation is the result of the localization net. The spatially warping feature map is created by feeding this into a bilinear sampling net. Lastly, the objective function, which is commonly employed in learning-free variational approaches, measures the loss as the photometric inaccuracy between the warped feature map from the source image and the target image. Originally intended for object level transformation modeling, this network can be thought of as a pixel-level implementation of the Spatial Transformer Networks (STNs) [68], specifically tuned for optical flow. Another key distinction is that this approach uses a loss in a completely unsupervised scenario, whereas STN uses object level supervision to learn the transformation in the context of recognition. The similar multi-scale EPE loss function of FlowNet [47] is also tailored.

Using a typical variational loss, the BackToBasics [69] technique (i.e., without utilizing ground truth flow) trains a CNN to predict optical flow between two images. It employs only the images for training instead of the associated ground truth flow and imagery. Specifically, we use a loss function that combines a spatial term that simulates the predicted change of flow throughout the image with a data term that assesses photometric constancy over time. Based on the network's expected optical flow, the photometric loss calculates the difference between the initial input image and the (inversely warped) subsequent image. The difference between spatially adjacent

flow predictions is measured by the smoothness loss. These two losses together create a stand-in for losses determined by the ground truth flow.

An end-to-end unsupervised method that shows how well unsupervised learning works for optical flow is presented by UnFlow [70]. Building on current optical flow CNNs [47], [61], it substitutes an unsupervised photometric loss akin to that in [69] for the supervision from synthetic data. By executing a second pass with the two input images switched, it calculates bidirectional optical flow (i.e., both in forward and backward direction). After that, it uses the census transform to improve robustness on real images and creates a loss function that explicitly considers occlusion by utilizing bidirectional flow. It builds upon the earlier FlowNetS-based BackToBasics [69] and extends it in three key ways: iterative refinement by stacking multiple FlowNet networks [61], training FlowNetC with its comprehensive unsupervised loss to estimate bidirectional flow, and designing a symmetric, occlusion-aware loss based on bidirectional (i.e., forward and backward) optical flow estimates. In order to determine whether unsupervised procedures are a good substitute for or supplement to supervised learning, this work explores potential methods for increasing their accuracy.

LearningByAnalogy [71] suggests a framework that uses augmentations to self-supervise and learn optical flow from unlabeled image sequences. The fundamental learning framework is modified by adding a forward pass for transformed images, where the supervision comes from the transformed prediction of original images, to prevent the goal of view synthesis from being unreliable on transformed data. The architecture enables the network to learn by analogy with the original samples, progressively learning the capacity to handle tough samples. The self-supervision from transformations prevents the unsupervised aim from being confusing in challenging circumstances.

To improve the matching accuracy, UnSupOptFlowWithOcc [72] suggests a patch-based census constancy loss by patch-based warping. It uses a local patch as the representation of the center point to find its correspondence rather than just taking into account one point. The census transform, which is not affected by changes in illumi-

nation, is used to compare the differences between patches. Additionally, it develops an additional branch to concurrently estimate a soft occlusion mask in an unsupervised manner. To lessen the impact of the occlusion zone, the estimated occlusion mask is trained unsupervised using pixels with a high patch constancy error and poor forward-backward flow consistency. Among unsupervised learning techniques utilizing the FlowNet-like structures [47], [61], [73], this approach achieves the SOTA results among unsupervised-based methods.

A CNN-based non-local term, in ULOFNL [74], is created to address the issue of over-smoothing beyond motion bounds. Because a non-local term is incorporated into the DCNN, its weights can be automatically learned, making it appropriate for denoising a variety of images while maintaining motion boundary features. Second, the network is designed to extract occlusion maps and combine them to create a loss function in order to address issues caused by occluded regions, which typically yield inaccurate information. In particular, bi-directional occlusion maps are produced by estimating the forward and backward optical flows. More specifically, two consecutive frames are used as inputs during the training phase. Both forward and backward optical flows are computed by switching the image order. The calculated optical flow is then refined using the CNN-based non-local term. Subsequently, bi-directional occlusion maps are obtained by forward and backward consistency checks using the revised bidirectional optical flow. A loss function is constructed based on the estimated occlusion maps and bi-directional optical flow.

2.2.2.3. Comparison of DL-Based Optical Flow Methods in Terms of Learning. The performance difference between supervised and unsupervised learning-based approaches is still present. Supervised methods, trained with ground truth optical flow datasets using the combination of a bunch of loss functions, outperform unsupervised methods. Even though the performance differences, the frameworks of these methods are very similar except the way the networks are trained. Especially the loss functions, tailored during training, are almost the same, and itemized as,

- Photometric Loss: This loss term , $Loss_p$, is tailored to measure the difference between the target image and the image, warped using the optical flow vector that is predicted by a CNN model over the non-occluded regions [5], and defined as

$$Loss_p^1 = [\sum_{x,y} \Psi(\tilde{f}(x,y) - f(x,y)).O(x,y)]/[\sum O(x,y)] \quad (2.2)$$

$$Loss_p^2 = [\sum_{x,y} \Psi(\nabla\tilde{f}(x,y) - \nabla f(x,y)).O(x,y)]/[\sum O(x,y)] \quad (2.3)$$

where $\tilde{f}(x,y)$ stands for the image, warped from the target image $f(x,y)$ using the optical flow vector over the non-occluded region, $O(x,y)$. While $Loss_p^1$ is derived with image brightness, $Loss_p^2$ is calculated using the gradient of the image. $Loss_p$ is computed using the Charbonnier penalty function $\Psi(s) = \sqrt{s2 + 0.0012})$ as the combination of $Loss_p^1$ and $Loss_p^2$.

- Smoothness Loss: This loss term, $Loss_s$, is designed to provide a smooth optical flow result, especially on the motion boundaries that frequently coincide with the image boundaries. While the occluded region has no photometric loss, the existence of the estimated optical flow vector in the occluded region will be determined by $Loss_s$ [5]. $Loss_s$ is elaborated as

$$Loss_s^1 = \sum_{x,y} \sum_{d \in x,y} \Psi(\mid \partial_d PF_{fwd} f(x,y) \mid e^{-\alpha|\partial_d f(x,y)|}) \quad (2.4)$$

$$Loss_s^2 = \sum_{x,y} \sum_{d \in x,y} \Psi(\mid \partial_d^2 PF_{fwd} f(x,y) \mid e^{-\alpha|\partial_d f(x,y)|}) \quad (2.5)$$

where ∂_d and ∂_d^2 stand for the first-order and second-order derivatives of the predicted forward optical flow, $PF_{fwd}(x,y)$, by the CNN, and α controls the weight of edges, and d indexes over partial derivative on x and y directions over the image, f. $Loss_s$ is computed with the combination of $Loss_s^1$ and $Loss_s^2$.

- Forward-Backward Consistency Loss: This loss term, $Loss_c$, is designed to measure the bi-directional motion consistency between two consecutive images [75]. The flow vectors of pixels, visible in both images, must be the inverse of each other. If one of the pixels is not seen in any of the images, it means that the unseen pixel is occluded. $Loss_c$ computes the Euclidean distance between these

pixels as

$$Loss_c^f = \overline{o}_{f(x,y)}^t \overline{o}_{f(x,y)'}^{t+1} (\| f(x,y) - PF_{bwd} f(x,y)' \|) \tag{2.6}$$

$$Loss_c^b = \overline{o}_{f(x,y)}^{t+1} \overline{o}_{f(x,y)''}^{t} (\| f(x,y) - PF_{fwd} f(x,y)'' \|) \tag{2.7}$$

$$Loss_c = \sum_{x,y \in f(.)^t} Loss_c^f + \sum_{x,y \in f(.)^{t+1}} Loss_c^b \tag{2.8}$$

where $f(x,y)' = PF_{fwd} f(x,y)$ stands for the pixel that is forward-projected into the image coordinates at time $t+1$, and $f(x,y)'' = PF_{bwd} f(x,y)$ represents the pixel, back-projected into the image coordinates at time t. $\overline{o}_{f(x,y)}$ is 1 if the pixel, $f(x,y)$, is non-occluded.

- Occlusion-Disocclusion Symmetry Loss: Occlusions and disocclusions demonstrate a symmetry relationship. While already-occluded pixels in $f(x,y)$ at time t will correspond to the disocclusions at $f(x,y)'$ at time $t+1$ in the forward direction, will-be-occluded pixels $f(x,y)$ at time $t+1$ will correspond to the disocclusions at $f(x,y)'$ at time t in the backward direction. This loss term, $Loss_{sym}$, penalizes cases in which the occlusion-disocclusion symmetry relationship is not valid [75].

2.2.3. Occlusion Detection with Optical Flow Estimation

The effect of occlusion on optical flow estimates is one of the most important issues that drags the performance to lower levels. It is difficult to establish pixel-level correspondences while reconstructing the distorted image following the warping process in optical flow computation.

Occlusion results in pixel occlusion, which impairs the error function and makes optical flow estimation challenging. To avoid occluded pixels negatively affecting the non-occluded areas, precise information about occluded areas is essential for accurate optical flow calculation. Thus, in occlusion detection, the calculation of bidirectional optical flow is combined with the inclusion of obstructed pixels in the error function. The mutual reliance of optical flow and occlusion calls for a cooperative strategy and

contends that this has not been carried out as effectively as it may be. This strategy has led to the development of numerous techniques in the literature in recent years that estimate the optical flow and the occlusion at the same time. To achieve this, methods can be broadly categorized into three groups [5]:

- Methods that estimate pixels in occluded regions as constant values and treat the occlusion as an outlier.
- Methods that ignore the error function in occluded regions and take advantage of the symmetric nature of the optical flow.
- Intricate structures that can simulate image layering or depth.

The advanced methods, classified as a second group member in the literature, and explained in the following paragraphs, can simultaneously predict optical flow and occlusion.

MirrorFlow [75] discusses the occlusion map estimation and optical flow as a chicken-and-egg problem, and suggests a combined energy formulation and optimization technique. Through the estimation of forward and backward flow as well as occlusion maps, both directions collectively, the method directly makes use of their relationship and enables them to benefit from one another. It takes advantage of the forward-backward flow consistency and occlusion-disocclusion symmetry, two important symmetry characteristics of the optical flow field and the occlusion map within the two consecutive images. These symmetry characteristics enable a greater exploitation of the geometric and temporal information in the two sequential images, in addition to coupling optical flow with occlusion.

Even though CNN-based models have demonstrated excellent accuracy and robustness in optical flow computation based on a number of publicly available optical flow databases, such as the MPI-Sintel [49] and KITTI [50] benchmarks, the majority of CNN-based optical flow methods still struggle with edge-blurring brought on by motion occlusions. In Parallel Multiscale Context-Based Edge-Preserving Optical Flow

Estimation With Occlusion Detection (PMC-PWC) method [76], a parallel multiscale context-based pyramid, warping, and cost (PWC) volume network with occlusion detection for edge-preserving optical flow estimation is presented to address the problem of edge-blurring induced by motion occlusions. PMC-PWC approach yields improved optical flow with the noteworthy advantage of edge-preserving, particularly in occluded locations. For occlusion identification, a parallel multiscale context network is built. The suggested PMC network refines the occlusion boundaries by extracting multiscale context information. An occlusion detection module is created by combining the PMC network with a context network. The occlusion module is then integrated into a pyramid, warping, and cost volume network to create an edge-preserving optical flow model. The combination of edge loss, EPE-based loss, and binary cross-entropy loss, can supervise the network, and estimate the flow field and occlusions at the same time. Using a deep network for disparity or optical flow estimate based on FlowNet2 [61], FlowNet3 [73] elegantly integrates occlusion estimation with depth or motion boundary estimation, and provides these values clearly as output.

Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation (IRR-PWC) [77], which draws inspiration from both residual networks and classical energy minimization techniques, suggests an iterative residual refinement (IRR) scheme based on weight sharing that can be integrated with multiple backbone networks [47], [65]. It either increases accuracy, decreases the number of parameters, or even does both. Furthermore, it demonstrates how the accuracy may be increased by incorporating bi-directional flow estimate and occlusion prediction into the IRR method. This method is the inspiration for the PFCNNOcc. The designed CNN model, which predicts forward and backward optical flow and occlusion using a pyramidal approach from the input phase to the output, with a novel weight sharing policy, common for all levels inside the pyramid.

PWOC-3D [78] has specialized on particular issues in a standard scene flow pipeline. First, it uses a spatial pyramid and warp image features at each pyramid level using the intermediate flow estimate from the previous level to handle large mo-

tion. This method is based on the work of PWC-Net [65] for optical flow. To enhance scene flow estimations, it also suggests a self-supervised method for predicting dense occlusion maps from images without the need for tagged occlusion data. By reducing the computational load, this approach is among the first to reason about occlusion with a single flow prediction and no occlusion ground truth.

To jointly learn optical flow and occlusions, UnSupOptFlowOcc [79] models temporal relationships over many frames. It does this by adding more frames to the two-frame design suggested in [65]. Then, within a three-frame temporal window, it estimates optical flow in both past and future directions together with an occlusion map. The estimated flow fields and occlusion map are used by the unsupervised loss to assess the distorted images from the past and the future. The technique adds a constant velocity restriction inside the temporal frame in addition to the standard spatial smoothness constraints. This makes it possible to use temporal information to anticipate optical flow in occluded regions more precisely while reasoning about occlusions in a rational way.

Large displacements and motion occlusions continue to be problems for the majority of learning-based models, despite the impressive accuracy and efficiency of optical flow techniques based on CNNs. MulFeatOptFlowOcc [80] suggests a self-attention-based multiscale feature learning optical flow computation method with occlusion map detection, based on PWC-Net [65], to increase the precision and resilience of optical flow estimation under large displacements and occlusions. In order to estimate optical flow, it uses the self-attention method to build a multiscale feature learning network. This model can compensate for the high displacement optical flow by capturing long-range dependencies.

The normalized occlusion feature map is created by taking advantage of a self-learning occlusion prediction module. The optical flow in occluded areas is then corrected using the resulting occlusion feature map. A hybrid loss function that combines the photometric and smoothness losses with the traditional EPE loss is used to guar-

antee the good performance of the method. The suggested optical flow network is made more precise and resilient to significant displacements, occlusions, and image noise thanks to this hybrid loss.

In order to detect and manage occlusion, OccDetProOptPF [81], integrates particle filtering and optical flow. By extracting the spatial-temporal information, it maintains the robustness of the optical flow. The ambiguity around the occluded item is simultaneously addressed by the probabilistic character of the particles. The detection of occlusion is achieved via controlling an error function, called the sum of the squared error (SSE), in a local window around the target. Occlusion decision is given when SSE is bigger than a pre-determined threshold. This is an ad-hoc method that does not use any CNN model to extract optical flow, occlusion, and segmentation. PFCNNOcc, the proposed method in this thesis, on the other hand, uses CNN models to perform all these tasks. Moreover, the states of the occlusion problem are not explained, and there is no function to find the states of occlusion and their transition to each other [81]. The method is not compared with the SOTA trackers in the UAV benchmark datasets [2], [82, 83]. That is why the performance of this method is not well understood. Hence, the way that PFCNNOcc approaches the occlusion problem is very different from OccDetProOptPF [81] both theoretically and philosophically, and the PFCNNOcc outperforms the SOTA trackers in the UAV123@10fps dataset while the performance of the OccDetProOptPF [81] is unknown.

2.3. Overview

The SOTA visual tracking algorithms in recent years are notably represented by the highly effective trackers, explained in [1], [4], [12, 13].

According to the assertion made in [12], SRDCF [21] was the best technique in terms of accuracy and success rates on the UAV123 dataset, which is largely due to its high fidelity scale adaptation. The performances of MUSTER [84], DSST [20], STRUCK [22, 23] are comparable.

Therefore, the best trackers created with correlation filters in [12] are SRDCF [21], MUSTER [84], and DSST [20].

The DL-based techniques in [4] differ from those in [12] in terms of accuracy and success rates. DaSiamRPN [38], SiamRPN [37], SiamAPN [35] and SiamAPN++ [39] are the most promising methods, and achieve very high results during tracking compared to other SOTA trackers.

Based on the literature review on optical flow and occlusion detection, it is observed that the most advanced methods encountered are those capable of simultaneously detecting optical flow and occlusion with high precision and accuracy. These methods are designed to leverage the symmetry property of optical flow, based on supervised learning. Particularly, algorithms based on the process of improving the total error, resulting from predicting optical flow and occlusion at each pyramid level of the newly resized images, and refining these predictions in the DCNN model, have been found to achieve high success rates. The bidirectional estimation of optical flow (forward and backward flow) further enhances the performance of these methods. In the PWC-Net method [65, 66], the approach mentioned above is applied successfully for optical flow estimation, utilizing the same DCNN model with different model parameters learned at each pyramid level. However, this leads to a significant increase in the number of parameters, posing challenges for real-time implementations.

Despite the high number of parameters, considering the accuracy of the optical flow simultaneously performed end-to-end, the PWC [65, 66] has served as a reference for many occlusion detection methods based on optical flow. In this context, one of the most prominent methods in the literature is the Iterative Residual Refinement PWC (IRR-PWC) approach [77]. As the name suggests, it integrates PWC [65, 66] approaches with iterative residual refinement. The most significant difference between the IRR-PWC [77] method and the PWC approach [65, 66] is that all DCNN model parameters used for optical flow and occlusion are shared across all pyramid levels. This reduces the number of parameters and enables real-time processing.

3. PFCNN : PARTICLE FILTER WITH CNN METHOD

The first and baseline visual object tracking algorithm in this thesis, the PFCNN, is designed with the combination of a particle filter [85] and a CNN classifier [6], and this novel method is elaborated in this chapter. To demonstrate the effectiveness and the superiority of the PFCNN, the APF [7–9] is simulated for tracking purposes, and the results are compared with those, obtained by the PFCNN to prove that the class probabilities from CNN, used to update the weights of the particle filter in the PFCNN, mimic the behavior of auxiliary weights in the APF.

3.1. PFCNN

3.1.1. Introduction

Here, we present a novel approach that combines the particle filter with a CNN model to achieve robust tracking against changes in scale, aspect ratio, target motion, visual model, and rotation. This approach is inspired by the concept of the tracking-by-detection based DL methods [4], the particle filter and the combination of the CNNs with the well-designed dense sampling algorithms. These concepts constitute the main idea that gives rise to the design of the PFCNN.

In the PFCNN, particle filtering produces particles as bounding box samples of the parts of the vehicle, dispersed in a proposed region surrounding the last known location of the target. The part of the vehicle stands for the section of it that the tracking process is based on. By calculating the distance between the characteristics of these bounding boxes and the reference bounding boxes, which fully describe the parts of the tracked object in the image, the probabilities of these particles are assessed.

The “Tracker and Estimator” and the “Feature Extractor” are the two primary modules that make up the PFCNN as shown in Figure 3.1. The modules for par-

ticle sampling, particle scaling, object state estimation, and unification make up the "Tracker and Estimator" module. The "Feature Extractor" module consists of a CNN that is used to extract features and to determine class probabilities from the sampled vehicle images, and it is trained using a low-altitude UAV dataset. As understood from the overall method, shown in Figure 3.1, "CNN Model For Class Probability " module is utilized to compute the class probabilities of the sampled bounding boxes.

The PFCNN makes the following three contributions:

- A mechanism for rescaling particles against changes in scale and aspect ratio is designed. Tracking under scale and aspect ratio variations is enhanced by implementing a particle rescaling technique in which each particle is resized with a different aspect ratio.
- A new structure that mimics the APF [7–9] is obtained by completely integrating the CNN model with the particle filter. State variable estimation is unquestionably improved by supporting the weight update process of particle filter with the class probability scores of the particles.
- The unified multi-part vehicle tracking framework is created where several tracking agents can function independently, and produce estimates during the particle sampling phase. During the estimation stage of particle filtering, the individual estimates for the parts are aggregated to get a final bounding box estimate for the whole vehicle.

The first fully connected layer of the CNN model is where the features of the cropped images corresponding to each sampled bounding box are extracted from. Through complete integration of the particle filter with the CNN model, the weight update of each particle is supported by the class probability of the CNN model; as a result, the integrated structure resembles the APF [7–9]. Thus, a potential error in the particle likelihood probabilities is corrected. During rotation and fast motion, the aspect ratio of the vehicle changes abruptly while the motion vectors of the front and back parts of it show different directions. To be able to keep track of the diverse

movement of the vehicle and to improve tracking performance against rotation and fast motion issues, this process is individually applied to different parts of the vehicle during tracking.

Variations in scale, aspect ratio, and visual model are other problems that require an online visual model update mechanism. We suggest a reliable technique that rescales every particle while sampling across the suggested region. As a result, the width and height of each bounding box vary, reflecting the many scale and aspect ratio modifications that the target is exposed to.

The particle, linked to the bounding box with the highest class probability, is chosen based on the class probabilities of the sampled bounding boxes, which are determined by the CNN model. Then, the weight of the particle is updated, and the state estimation is achieved in the particle filtering. Finally, the unification module updates the reference feature for both parts, determines the ultimate scale parameter, sP, along both axes, and unifies the final estimates of the parts.

The tracking performance against size, aspect ratio, rotation, and visual model fluctuations is enhanced by the particle rescaling and unification techniques. In the UAV123@30fps dataset [2], the experiments are conducted on vehicle image sequences that include significant differences in scale, aspect ratio, viewpoint, target motion, and rotation.

Depending on the scenario or attribute-based evaluation results, given in Section 6.1, the experimental results demonstrate that the PFCNN outperforms all or most of the SOTA trackers and achieves good precision and success scores. The superiority in performance is due to the particle rescaling scheme, unified multi-part tracking, and the support of the class probability from CNN in the weight update process by mimicking the APF. To refine the estimate of the box size consistently, the final bounding box is rescaled into several candidate boxes with various scale values at that stage. The CNN model determines the class scores and features of each candidate box.

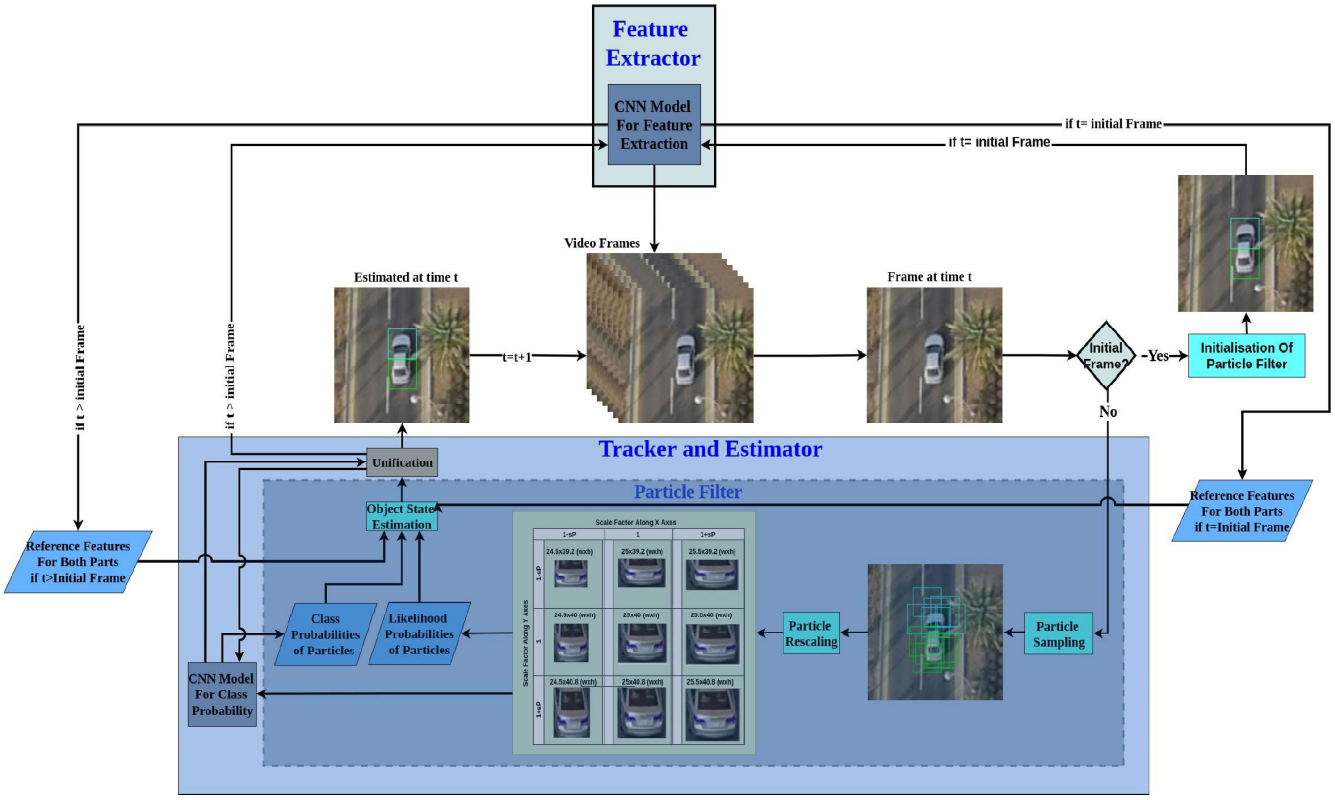


Figure 3.1. Overall method of the PFCNN.

The class score of each box is then used to weight the similarity distance between the reference box feature and the recent box features in a way that resembles the APF [7]. The "Unification" module places the PFCNN among the best SOTA trackers and enhances the standard particle filtering performance against changes in scale, aspect ratio, target motion, and visual model.

In Section 3.1.2, SIR particle filter is explained in detail. The architecture of PFCNN is elaborated in Section 3.1.3. The PFCNN is tested for a subset of the UAV123@30fps dataset [2] which does not include the image sequences with occlusion and out-of-view problems, and compared with the SOTA tracking methods in terms of precision and success rate in Section 6.1. In Section 3.1.5, a comprehensive analysis is carried out to find the best performing method between the SOTA trackers. The analysis consists of scenario-based, attribute-based and overall comparisons, and summarizes the pros and cons of each method. Finally, in Section 3.1.6, concluding remarks based on the analysis covering the SOTA methods are presented and the possible areas of future work are summarized.

3.1.2. Theorical Background of PFCNN: SIR Particle Filter

In this section, theoretical background of PFCNN is elaborated. PFCNN was created based on the Sequential Importance Resampling (SIR) particle filter [85, 86]. To fully comprehend the contributions of PFCNN against conventional SIR filter, the SIR filter derivations will be summarized.

SIR particle filter presents a recursive solution to estimate the state variables of a system under non-linear and/or non-Gaussian environments. To utilize the SIR particle filter, the process and measurement equations of the state-space model for the specific system should be defined as

$$\mathbf{x_t} = \mathbf{A}\mathbf{x_{t-1}} + \mathbf{v_t} \tag{3.1}$$

$$\mathbf{z_t} = \mathbf{C}\mathbf{x_t} + \mathbf{r_t} \tag{3.2}$$

$$\mathbf{v_t} \sim noisePDF_{proc} \tag{3.3}$$

$$\mathbf{r_t} \sim noisePDF_{meas} \tag{3.4}$$

where $\mathbf{x}_t$ and $\mathbf{z}_t$ are unobservable state variables and measurement vectors at time t. The process equation is used to predict the state variable at time t, and $\mathbf{A}$ is the transition function/matrix, and the measurements are updated using the measurement equation as explained in Equations (3.1-3.4) by utilizing the measurement matrix $\mathbf{C}$. $\mathbf{v_t}$ and $\mathbf{r_t}$ are the vectors of process and measurement noise processes, respectively, sampled from noise probability distribution functions (pdfs), $noisePDF_{proc}$ and $noisePDF_{meas}$. For the systems that can be modeled with linear equations as well as Gaussian distribution, the SIR particle filter boils down to the classical Kalman Filter [85]. As stated in [85], the aim is to estimate joint posterior probability density function (pdf), $(p(x_{t=0:t}|z_{t=0:t}))$, or marginal posterior pdf, $(p(x_t|z_{t=0:t}))$. Here $t = 0, 1, 2, ..., T$ is the discrete time index. State variables are estimated using the measurements, which are noisy versions of the state variables. During the estimation process, the posterior pdf of the state variables at time t is approximated as the weighted sum of delta functions located at the position of drawn particles, representing each sampled state variable, as [85]

$$\hat{p}(x_{0:t-1}|z_{0:t-1}) \approx \sum_{j=1}^{M} w_{t-1,j}\delta(x_{0:t-1} - x_{0:t-1,j}) \tag{3.5}$$

where j is the particle number, and M is the total number of particles. The weights, $w_{t-1,j}$, for each particle at time $t-1$ are computed using the joint posterior evaluated at the jth particle, $p(x_{0:t,j}|z_{0:t})$, and the joint importance density, $q(x_{0:t,j}|z_{0:t})$, as

$$w_{t,j} = \frac{p(x_{0:t,j}|z_{0:t})}{q(x_{0:t,j}|z_{0:t})}. \tag{3.6}$$

In Equations (3.5,3.6), $\delta(x_{0:t-1} - x_{0:t-1,j})$ shows the delta function at the position of the jth particle up to time t, and $w_{t-1,j}$ and $w_{t,j}$ are the weights of jth particle at time $t-1$ and t, respectively. The weights for all particles should sum to one.

To understand the significance of the importance density selection, the Bayesian theorem [85] is used to derive the joint pdf recursively as given below

$$p(x, z) = p(x|z)p(z) = p(z|x)p(x). \tag{3.7}$$

To establish a recursive weight update equation, it is required to factorize both pdfs into a product of other pdfs. Hence, the joint posterior can be calculated by Bayes theorem using the posterior from the previous time instant $t-1$ as [85]

$$\begin{aligned} p(x_{0:t}|z_{0:t}) &= \frac{p(z_t|x_t)p(x_t|x_{t-1})}{p(z_t|z_{0:t-1})} p(x_{0:t-1}|z_{0:t-1}) \\ &\propto p(z_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|z_{0:t-1}) \end{aligned} \tag{3.8}$$

where, $p(z_t|x_t)$ is the likelihood, $p(x_t|x_{t-1})$ is the transition density, and $p(x_{0:t-1}|z_{0:t-1})$ is the joint posterior pdf at time t-1.

By making an assumption similar to the Markov property. Joint importance density can be represented as

$$\begin{aligned} q(x_{0:t}|z_{0:t}) &= q(x_t|x_{0:t-1}, z_{0:t})q(x_{0:t-1}|z_{0:t}) \\ &= q(x_t|x_{0:t-1}, z_{0:t})q(x_{0:t}|z_{0:t-1}) \end{aligned} \tag{3.9}$$

where $q(x_{0:t}|z_{0:t})$ is the importance density at time t, and $q(x_{0:t-1}|z_{0:t})$ is the distribution whose state variables from previous time is conditioned on the measurements, updated up to present time.

Then, as in [85], the weight of the jth particle, $w_{t,j}$, at time t can be written in a recursive form as

$$\begin{aligned} w_{t,j} &= \frac{p(z_t|x_{t,j})p(x_{t,j}|x_{t-1,j})p(x_{0:t-1,j}|z_{0:t-1})}{q(x_{t,j}|x_{t-1,j})q(x_{0:t-1,j}|z_{0:t-1})} \\ &= w_{t-1,j}\frac{p(z_t|x_{t,j})p(x_{t,j}|x_{t-1,j})}{q(x_{t,j}|x_{t-1,j})}. \end{aligned} \tag{3.10}$$

One possible choice for the importance density, $q(x_{t,j}|x_{t-1,j})$, is the prior density,

$p(x_{t,j}|x_{t-1,j})$, in this case, the weight update equation simplifies to [85]

$$w_{t,j} = w_{t-1,j}p(z_t|x_{t,j}). \tag{3.11}$$

But it is not a perfect choice for the importance density because the prior density does not utilize the observations. At this point, it should be clarified that the PFCNN is designed based on the SIR particle filter, but some different features separate the PFCNN from the SIR particle filter, since PFCNN is a hybrid method that combines CNN with the SIR particle filter. The details about the architecture of the PFCNN will be elaborated in the next section.

3.1.3. Architecture of PFCNN

The goal of this method is to track a vehicle in low-altitude UAV videos while dealing with extreme visual issues such as fast motion, sudden rotation, size variation, aspect ratio shift, and visual model modification. A method whose high-level design is displayed in Figure 3.1 is offered to accomplish the successful target tracking against challenging scenarios. The "Tracker and Estimator" and the "Feature Extractor" are the two primary modules that make up the PFCNN. The modules for particle sampling, particle scaling, object state estimation, and unification make up the "Tracker and Estimator" module. The CNN used to extract features and to determine class probabilities from the sampled vehicle images is called the "Feature Extractor" and was trained using a low-altitude UAV dataset. Class probabilities are incorporated into the weight update of the particles to give a more effective sampling process and, thus, more accurate state estimates. The class probabilities can be interpreted as the auxiliary weights, utilized for the performance enhancement of the APF [7–9].

The ground truth values from the UAV dataset are used to individually initialize the state variables for N distinct vehicle parts at the first iteration of the particle filter, as shown in Figure 3.1. Following initialization, the "Feature Extractor" receives the bounding boxes that the sampled particles represent. This generates and saves reference features that are specifically designed to determine the likelihood probabilities of the particles in the next iteration.

Target tracking is handled by the "Tracker and Estimator" after initialization, which makes extensive use of the sampling strategy of the particle filter at the beginning of each iteration. As seen by the various colors in Figure 3.1, the bounding boxes surrounding the vehicle are made up of several particles that are independently sampled around their last known or estimated location for each part of the vehicle. Due to the challenge of visualizing N parts in Figure 3.1, the two-part processing of the PFCNN is depicted without loss of generality.

The best bounding box for each part is then determined by rescaling each proposal to several bounding boxes using a scale parameter, sP, in the "Particle Rescaling " module. Once the class probabilities are computed by CNN, the proposal with the highest class probability is selected as the best bounding box. Finally, using the likelihood and the class probabilities, the weight of each proposal is updated. The introduction of the class probabilities in the weight update improves the performance of the particle filter, as does the introduction of auxiliary weights in the APF [7–9].

The object state estimation module performs the independent estimation of the state variables of the different parts. Updating the target model with the running average of the current and past values of the previously stored reference features is another important feature of the object state estimation module. The vehicle is estimated as a whole object by unifying the estimated bounding boxes of various parts in the innovative unification module, which comes after object state estimation. To fine-tune the projected box size consistently, these boxes are rescaled into several candidate boxes with different scale factors.

The ability of the particle filter to sample multiple bounding boxes surrounding the last known or estimated location of the target makes it an immanent region proposal structure. Thus, its robustness against non-linear target motions is not the only reason to use it as an estimator in the PFCNN.

3.1.3.1. Initialization and Particle Sampling. The objective of this module is to properly configure the particle state variables and accomplish appropriate sampling by the particle filter for every part of the object. The process and measurement equations of the particle filter are given in Equations (3.1-3.4), where $\mathbf{x}_t^i$, $i = 1, 2, 3, ..., N$, is the state vector at time t, and of size N that is the number of parts. $\mathbf{x}_t^i$ includes the location, velocity, and the dimension of the bounding boxes along x and y coordinates. $\mathbf{z}_t^i$ is the measurement vector at time t. In the proposed method, $\mathbf{A}$ and $\mathbf{C}$ are accepted as identity matrices, and the noise terms, $\mathbf{v_t} \sim N(0, \sigma_{proc}^2)$ and $\mathbf{r_t} \sim N(0, \sigma_{meas}^2)$, are, respectively, sampled from i.i.d. normally distributed probability distribution functions (pdfs), shown in Equations (3.1-3.4).

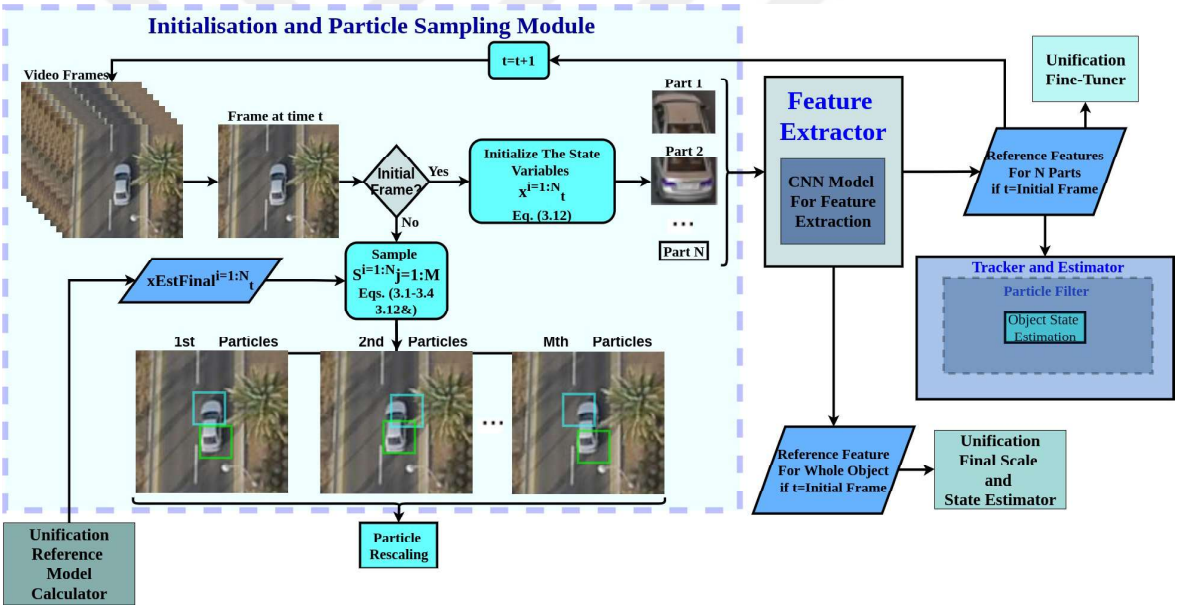


Figure 3.2. Initialization and particle sampling.

The random walk model provides the basis for the formulation of the process equation given by Equation (3.1). Figure 3.2 makes it clear that the particle state variables for different parts of the vehicle are initialized separately. Vehicles can be divided into N parts, and multi-part processing of the vehicle helps keep track of aspect ratio, abrupt rotational changes, and fast motion.

Thus, the state vector and noise vectors are defined as

$$\mathbf{x}_t^i = \begin{bmatrix} xLoc_1 \\ yLoc_1 \\ xVel_1 \\ yVel_1 \\ xBestBoxDim_1 \\ yBestBoxDim_1 \\ xLoc_2 \\ yLoc_2 \\ xVel_2 \\ yVel_2 \\ xBestBoxDim_2 \\ yBestBoxDim_2 \\ . \\ . \\ . \\ xLoc_N \\ yLoc_N \\ xVel_N \\ yVel_N \\ xBestBoxDim_N \\ yBestBoxDim_N \end{bmatrix} \quad \mathbf{v}_t^i = \begin{bmatrix} n_xLoc_1 \\ n_yLoc_1 \\ n_xVel_1 \\ n_yVel_1 \\ 0 \\ 0 \\ n_xLoc_2 \\ n_yLoc_2 \\ n_xVel_2 \\ n_yVel_2 \\ 0 \\ 0 \\ . \\ . \\ . \\ n_xLoc_N \\ n_xLoc_N \\ n_xVel_N \\ n_yVel_N \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{r}_t^i = \begin{bmatrix} n_xLoc_1 \\ n_yLoc_1 \\ n_xVel_1 \\ n_yVel_1 \\ 0 \\ 0 \\ n_xLoc_2 \\ n_yLoc_2 \\ n_xVel_2 \\ n_yVel_2 \\ 0 \\ 0 \\ . \\ . \\ . \\ n_xLoc_N \\ n_yLoc_N \\ n_xVel_N \\ n_yVel_N \\ 0 \\ 0 \end{bmatrix} . \tag{3.12}$$

The user initializes state variables in the first iteration, and the ground truth table is used to read the location and dimension data of the entire vehicle. These values are then computed for each part of the vehicle based on its orientation in the scene (i.e., the magnitude of its height and width). If N, the number of parts, is chosen as two, the box is split into two parts from the side with the larger magnitude.

After setting the initial values for the state variables for N parts, particles are drawn using the process equations given in Equations (3.1-3.4) and (3.12), and are sent to the "Particle Rescaling" module. In the meantime, the reference features for each part are extracted using the CNN model, as will be elaborated in Section 3.1.3.3, and will be used to calculate the likelihood of particles in the next iteration.

Without knowing the next maneuver of the vehicle, it might be challenging to identify the visual changes brought on by abrupt changes in scale, aspect ratio, fast motion, and rotation when viewing the vehicle as a whole. However, by processing N parts separately, it will be simple to adapt to aspect ratio variation, abrupt rotational changes, and fast motion; therefore, the final estimates for the entire vehicle will be based on the location, velocity, and dimensions of the bounding boxes of N parts.

3.1.3.2. Particle Rescaling. Despite the special design of the particle filter to work in nonlinear environments, it is difficult to accurately track the complex uncertainties introduced by sudden scale changes brought about by video camera zooming, aspect ratio variations brought by rotation, and fast motion. Additionally, the possibility to determine the optimal bounding box size of each particle for each part is increased by rescaling each particle by using a scaling parameter that varies incrementally with several times, defined before the iteration.

Sampled particles by rescaling function are found as

$$\mathbf{S} = \mathbf{S}^{i}_{j,k} \quad = \quad \begin{bmatrix} [S^1_{11}, S^1_{12}, ..., S^1_{1K}] \\ [S^1_{21}, S^1_{22}, ..., S^1_{2K}] \\ ... \\ ... \\ ... \\ [S^1_{M1}, S^1_{M2}, ..., S^1_{MK}] \\ [S^2_{11}, S^2_{12}, ..., S^2_{1K}] \\ [S^2_{21}, S^2_{22}, ..., S^2_{2K}] \\ ... \\ ... \\ ... \\ [S^2_{M1}, S^2_{M2}, ..., S^2_{MK}] \\ ... \\ ... \\ ... \\ [S^N_{11}, S^N_{12}, ..., S^N_{1K}] \\ [S^N_{21}, S^N_{22}, ..., S^N_{2K}] \\ ... \\ ... \\ ... \\ [S^N_{M1}, S^N_{M2}, ..., S^N_{MK}] \end{bmatrix} \tag{3.13}$$

where $S^{i=1:N}_{j=1:M,k=1:K}$, $i = 1, 2, ..., N$, $j = 1, 2, ..., M$, $k = 1, 2, ..., K$ refers to all bounding boxes sampled independently for all particles sampled for each part, and thus, is a matrix of dimension $NxMxK$. M stands for the total number of particles, N is the total number of parts, and K is the number of rescaled bounding boxes for a particle, sampled for each part as shown in Figure 3.3. Following the application of M sampled particles for each part to the particle rescaling function, class scores of these particles

for the specific parts, $P(features_{S^i_{j,k}} \mid class_i)$ or $P^i_{j,k}$, are calculated by CNN as

$$\begin{aligned} P(features_{S^i_{j,k}} \mid class_i) =& CNN(S^i_{j,k}) \\ P^i_{j,k} =& P(features_{S^i_{j,k}} \mid class_i) \end{aligned} \tag{3.14}$$

where $class_i$ stands for reference class ID of i'th part of the vehicle. The optimal bounding boxes or particles for sampling M particles, $S^{i=1:N}_{j=1:M}$, are then found as

$$\begin{aligned} bestBoundingBoxes^i_j =& idx(P^i_{j,k} = max(P^i_{j,k})) \\ =& \begin{bmatrix} bestBoundingBox^1_1 \\ bestBoundingBox^1_2 \\ \dots \\ \dots \\ \dots \\ bestBoundingBox^1_M \\ bestBoundingBox^2_1 \\ bestBoundingBox^2_2 \\ \dots \\ \dots \\ \dots \\ bestBoundingBox^2_M \\ determine... \\ \dots \\ \dots \\ bestBoundingBox^N_1 \\ bestBoundingBox^N_2 \\ \dots \\ \dots \\ \dots \\ bestBoundingBox^N_M \end{bmatrix} \end{aligned} \tag{3.15}$$

based on class scores or so-called class probabilities, which indicate how much these particles belong to those particular parts as illustrated in Figure 3.3. The best, rescaled particles are then delivered to the "CNN Model For Class Probability" module in order to determine the class probabilities of these particles to be input to the "Object State Estimation" module as

$$\begin{aligned} P(features_{S_{j=1:M}^{i=1:N}} \mid class_i) = & \quad CNN(S_{j=1:M}^{i=1:N}) \\ P_{j=1:M}^{i=1:N} = & \quad P(features_{S_{j=1:M}^{i=1:N}} \mid class_i) \end{aligned} \tag{3.16}$$

which is elaborated in Section 3.1.3.3. It should be stated that there is a trade-off between the computation cost and the estimation accuracy when deciding on K and N. The likelihood of estimating the sP correctly increases as K and N get higher values at the expense of higher computation cost.

To seamlessly estimate the scale or aspect ratio adjustments, sPs at the particle rescaling function, as depicted in Figure 3.3, are established using increasing or decreasing steps. Small-scale and aspect ratio changes in the target are overlooked during tracking for the high sP values. The degree of severity of the scale or aspect ratio fluctuations greatly influences the value of sP. The particles can therefore determine whether there is a scale or aspect ratio fluctuation and the extent of these variations along both axes by using a scaling mechanism, as suggested above. The procedure is repeated until the required number of iterations is completed.

3.1.3.3. CNN Model. The CNN model is utilized in the PFCNN for feature extraction and class probability computation. $P(features_{S_{j,k}^{i}} \mid class_i)$ or $P_{j,k}^{i}$ is utilized by particle rescaling in Equation (3.14), and $P(features_{S_{j=1:M}^{i=1:N}} \mid class_i)$ or $P_{j=1:M}^{i=1:N}$ in Equation (3.16) is customized by the unification module and object state estimation module, which is controlled by the particle filter.

The VGG16 CNN model [87] is tailored as the CNN model in PFCNN. It has already been trained using the Imagenet dataset [88], and refined by transfer learning

using the UAV123 dataset [2] for PFCNN. Only the final two fully connected layers, which store the high-level properties of vehicles, are updated throughout training; the weights of the shallow layers of the VGG16 are fixed. The output layer is changed to categorize the samples into only five different classes, such as 1. boat, 2. car, 3. half boat, 4. half car, and 5. background. The reason to add the background class is to be able to classify the sampled bounding boxes that do not include a car, a bot, and half of them correctly. If the background class is not added to the training dataset, the CNN model will not learn the background and will classify the samples into just four classes, even if the sampled bounding box includes only the scene from the background. Hence, this situation affects the correctness of class scores that are utilized to update the weight of particles in the particle filter as well as the reliability estimation of the state variables of the tracked target negatively.

To extract a reference feature and estimate a class probability for each distinct part at the end of each iteration, the CNN is trained using half of a boat and half of a car. The weight update mechanism of the particle filter uses the reference features as measurements. Additionally, the CNN model is designed to determine the class probabilities of each rescaled particle in the "Particle Rescaling "and "Unification "modules in order to determine the optimal bounding box that best represents the particular particle.

The CNN output for a sampled particle is a vector containing probabilities for four object classes, and the sum of these class probabilities equals 1. The weight update process of the particle filter is evolved using these class probabilities as auxiliary weights, which will be elaborated in Section 3.1.3.4.

3.1.3.4. Object State Estimation. The foundation of the PFCNN is object state estimation. This module simulates the APF [7–9] by combining a CNN model with an SIR particle filter [85, 86]. The CNN model at the final step of Figure 3.3 computes the class probabilities, namely, $P(features_{S_{j=1:M}^{i=1:N}} \mid class_i)$ or $P_{j=1:M}^{i=1:N}$. During the weight update process, the particles are generated in the regions with high like-

lihood, reducing large variances in weights. Consequently, the state estimates are more accurate and the sampling process is more effective. The likelihood probabilities, $p(feature_t(S_{j=1:M}^{i=1:N}) \mid refFeat_t(part)^i)$, for each particle of each part are calculated as

$$\begin{aligned} &p(feature_t(S_{j=1:M}^{i=1:N}) \mid refFeat_t(part)^i) = \\ &N(refFeat_t(part)^{i=1:N} - feature_t(S_{j=1:M}^{i=1:N}), \sigma_{meas}^2) \end{aligned} \tag{3.17}$$

where these likelihoods indicate how similar the extracted features $feature_t(S_{j=1:M}^{i=1:N})$ are to the reference feature for each part, $refFeat_t(part)^{i=1:N}$, which was extracted by the "Feature Extractor-CNN Model For Feature Extraction" module as shown in Figure 3.4.

The state estimation is carried out as follows: $P(features_{S_{j=1:M}^{i=1:N}} \mid class_i)$ or $P_{j=1:M}^{i=1:N}$ are fed to the "Object State Estimation" module as shown in Figure 3.4 from "CNN Model For Class Probability"at "Particle Rescaling" module in Figure 3.3. Then, the weight update equation is given as

$$\begin{aligned} w_{t,j=1:M}^{i=1:N} =& w_{t-1,j=1:M}^{i=1:N} * p(feature_t(S_{j=1:M}^{i=1:N}) \mid refFeat_t(part)^i) \\ &* P(feature_t(S_{j=1:M}^{i=1:N}) \mid class_i). \end{aligned} \tag{3.18}$$

Here, the weights at time t, $w_{t,j=1:M}^{i=1:N}$, are updated for all particles, j, for each part, i, using the class probabilities and likelihood. Figure 3.4 illustrates how the estimated values or vector for the state variables for each part, i, at time t, $\mathbf{x}Est_t^{i=1:N}$, are computed and passed to the "Unification" module for the unification process following the weight resampling.

Thus, at this point, the proposed PFCNN method differs from the conventional SIR PF. The additional term, namely, $p(z_t|x_{t,j}, class)$ or $P(feature_t(S_{j=1:M}^{i=1:N}) \mid class_i)$ is referred to as the class probability, and is calculated by CNN. It quantifies how much the object inside the sampled bounding box resembles the class of reference observation, z_t or $feature_t(S_{j=1:M}^{i=1:N})$ at time t. By including $p(z_t|x_{t,j}, class)$, in other words, the class

score of jth particle, the likelihood, $p(z_t|x_{t,j})$ or $p(feature_t(S_{j=1:M}^{i=1:N}) \mid refFeat_t(part)^i)$, is supported. As a result of this, the meaningful particles are resampled more, and they are used to sample more particles around the area where the presence of the target is much more probable in the following iteration. Hence, this additional term acts like the auxiliary particle of the APF [7–9], and is used to reduce the imprecise estimation, caused by the significant variances that the particle weights encounter.

3.1.3.5. Unification. The "Unification" module, depicted in Figure 3.5, is intended to amalgamate the part estimates, computed by the "Object State Estimation" module, as illustrated in Figure 3.4. The three distinct sub-modules that make up the unification process are illustrated in Figures 3.6, 3.7, and 3.8 are referred to as "Fine-tuner", "Final Scale and State Estimator", and "Reference Module Calculator", respectively. The estimated bounding boxes for each independent part, i, are resampled into f, $f = 1, 2, 3, ..., F$, number of bounding boxes as detailed below

$$\begin{aligned} \mathbf{x}Est_{t,f=1:F}^{i=1:N} &= \mathbf{x}Est_t^{i=1:N} + r_{t,f=1:F}^{i=1:N} \\ r_{t,f}^i &= N(0, \sigma_{meas}^2) \end{aligned} \tag{3.19}$$

to fine-tune each estimate for each part, i, at time t in the "Fine-Tuner" shown in Figure 3.6.

As seen in Figure 3.6, the bounding boxes are then input into the CNN to determine their class probabilities, $P(feature_{xEst_{t,f=1:F}^{i=1:N}} \mid class_i)$, and their features, $feature_{xEst_{t,f=1:F}^{i=1:N}}$. The bounding box scores for each part, $score_{t,f=1:F}^{i=1:N}$, are determined by multiplying their class probabilities by the similarity score, $P(feature_{xEst_{t,f=1:F}^{i=1:N}} \mid refFeat_t(part)^{i=1:N})$, which is computed between their features and their reference features as

$$\begin{aligned} score_{t,f=1:F}^{i=1:N} = &P(feature_{xEst_{t,f=1:F}^{i=1:N}} \mid refFeat_t(part)^{i=1:N}) * \\ &P(feature_{xEst_{t,f=1:F}^{i=1:N}} \mid class_i). \end{aligned} \tag{3.20}$$

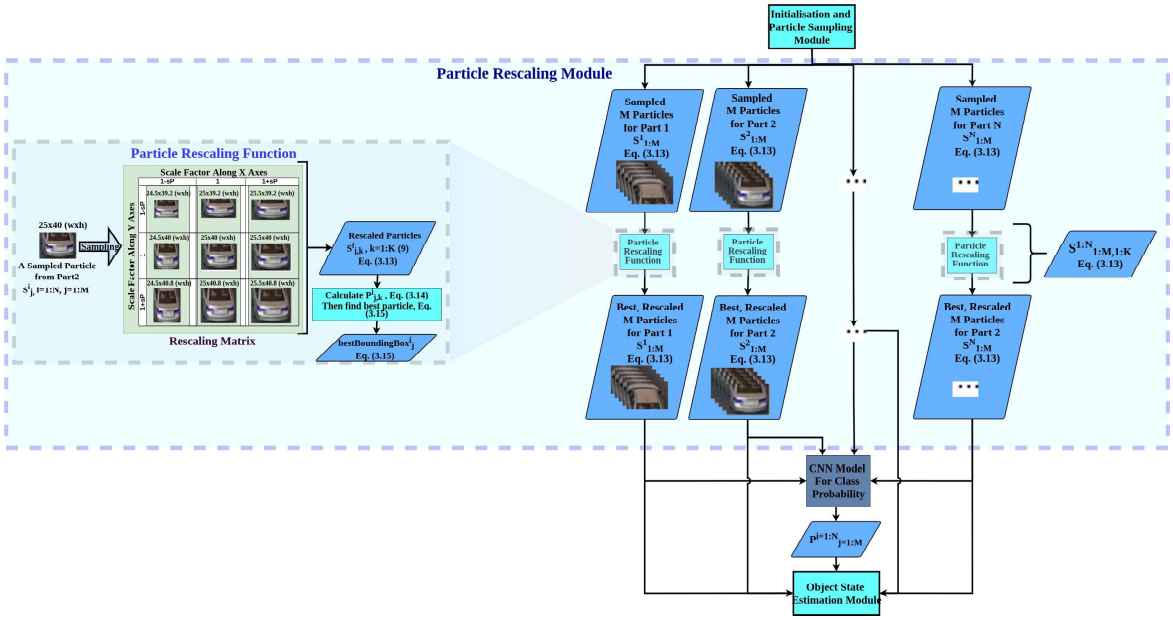


Figure 3.3. Particle Rescaling: The images in the rescaling matrix are scaled by 1-sP, 1, and 1+sP, depending on their position. The images on the diagonal are the ones that do not face any aspect ratio variation; thus, they are scaled by the same amount along both directions. However, the images on the off-diagonal positions experience aspect ratio variations; thus, they are scaled by different amounts along each direction.

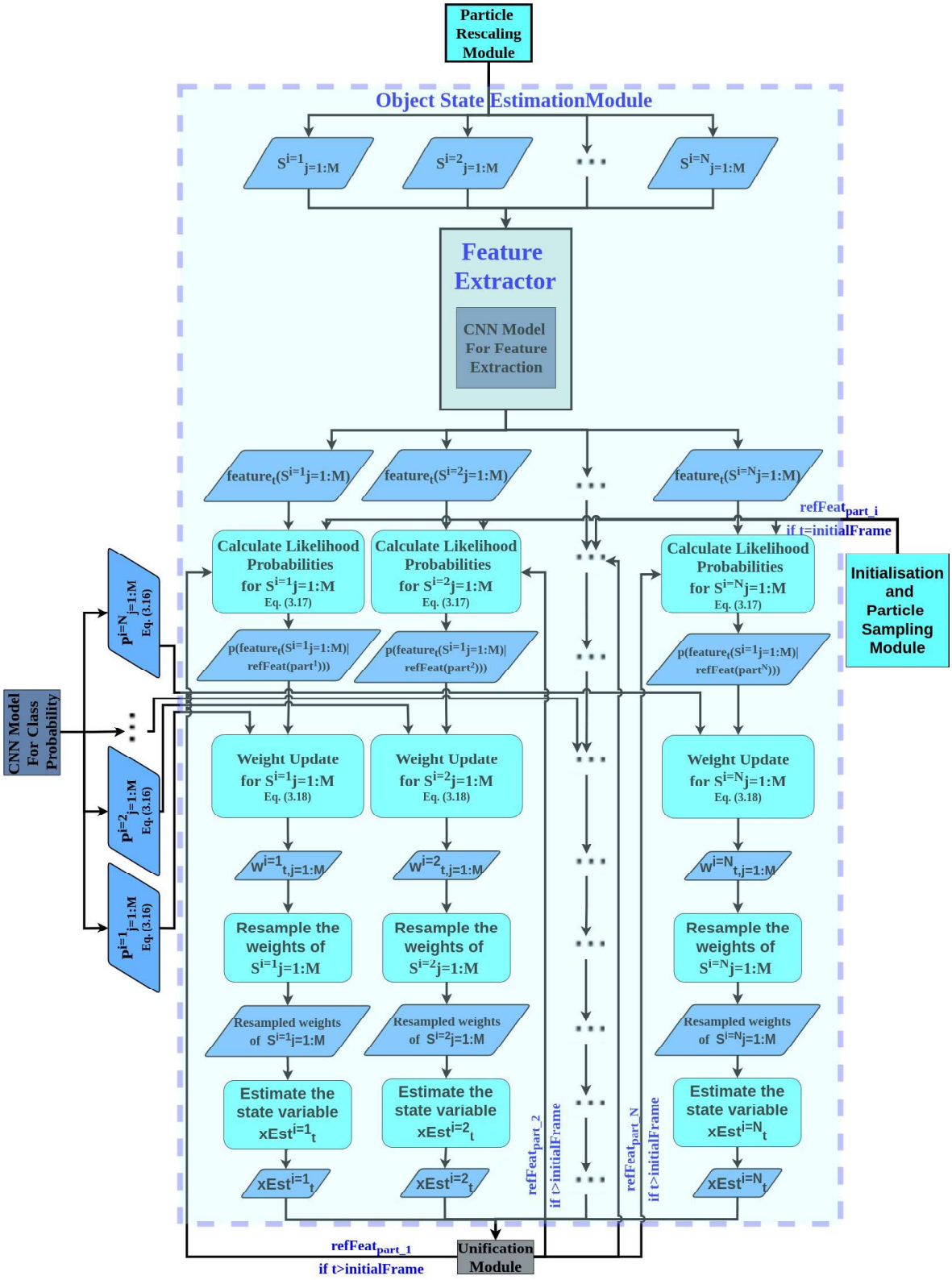


Figure 3.4. Object state estimation.

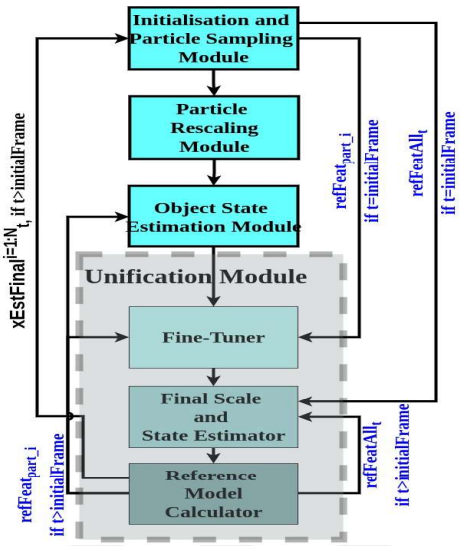


Figure 3.5. Overall unification module.

This computation is accomplished using the same methodology used in Equation (3.18) for the weight updating procedure of the particle filter. The optimal bounding box, as a fine-tuned estimate, $xEst_{ft,t}^{i=1:N}$, for each part, i, with the highest score, is chosen as the last step of the "Fine-Tuner", as

$$\begin{aligned} idx =& (score_{t,f=1:F}^{i=1:N} = max(score_{t,f=1:F}^{i=1:N})) \\ \mathbf{x}Est_{ft,t}^{i=1:N} =& \mathbf{x}Est_{t,f=idx}^{i=1:N} \end{aligned} \tag{3.21}$$

where idx refers to the index of the best bounding box for each part. "The Scale and Final Estimator"then receives these refined estimates for each part, combining them into a final estimated bounding box for the entire vehicle, as shown in Figure 3.7. First, using the refined estimates for each part, an initial bounding box or all-in-one estimate is unified, and its position and dimension along both axes are computed, as

$$\begin{aligned} xEstAll_{t,xLoc} &= min(xEst_{ft,t,xLoc}^{i=1:N}) \\ xEstAll_{t,yLoc} &= min(xEst_{ft,t,yLoc}^{i=1:N}) \\ xEstAll_{t,xBestBoxDim} &= max(xEst_{ft,t,xLoc}^{i=1:N}) - xEstAll_{t,xLoc} \\ xEstAll_{t,yBestBoxDim} &= max(xEst_{ft,t,yLoc}^{i=1:N}) - xEstAll_{t,yLoc}. \end{aligned} \tag{3.22}$$

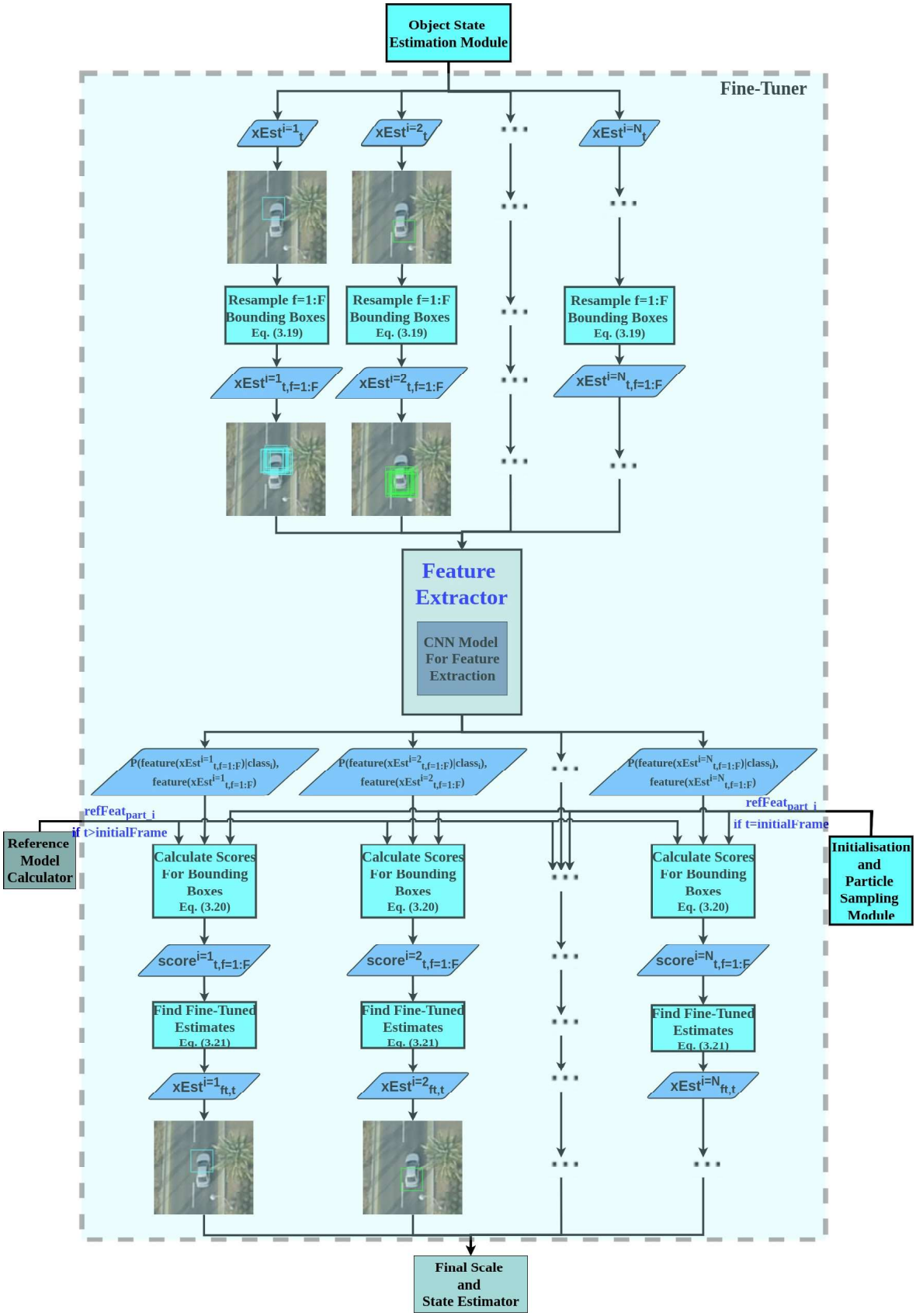


Figure 3.6. Unification module fine-tuner.

Then, using a function $\Theta(t, xEstAll^l)$ as described in Figure 3.3 with sP=0.05, the initially unified bounding box or all-in-one estimate is rescaled into L^2 number of bounding boxes, $RS_{t,l}, l = 1, 2, .., L^2$. L refers to the number of the different scale values for x and y axeses. Hence, the total number of combinations for the application of scaling to both axeses of the unified bounding box equal to L^2. This is done in order to determine the optimal bounding box under extreme variations in scale, aspect ratio, target motion, and rotation, as parameterized below

$$RS = RS_{t,l} = \Theta(t, xEstAll^l). \tag{3.23}$$

Finally, the best sP and dimensions for the unified bounding box with the highest score are chosen as the final estimate for the entire vehicle as follows

$$\begin{aligned} score_{t,l=1:L^2} =& P(feature_{RS_{t,l=1:L^2}} \mid refFeatAll_t) * P(feature_{RS_{t,l=1:L^2}} \mid class_{wholeCar}) \\ idx =& (score_{t,l=1:L^2} = max(score_{t,l=1:L^2})) \\ xEstAllFinal_t =& RS_{t,l=idx} \end{aligned} \tag{3.24}$$

where the scores, $score_{t,l=1:L^2}$, for each rescaled bounding box, $RS_{t,l}, l = 1, 2, .., L^2$, are computed as the last step, and idx refers to the index of the best bounding box for the whole vehicle in other words the final estimate of the bounding box. The procedure to update the estimate of the state variables and the reference features of the parts for the subsequent iteration is given in Equation (3.24), and is illustrated in Figure 3.7. The "Reference Model Calculator"is fed by the best sP and the values for the estimated state variables of the unified vehicle.

The reference feature for every part is determined in the "Reference Model Calculator", as illustrated in Figure 3.8. The parts are modified in accordance with the proportions of the whole vehicle. The unified bounding box is divided horizontally from the middle if the height of the vehicle is greater than the its width; if not, the box is cut vertically from the center.

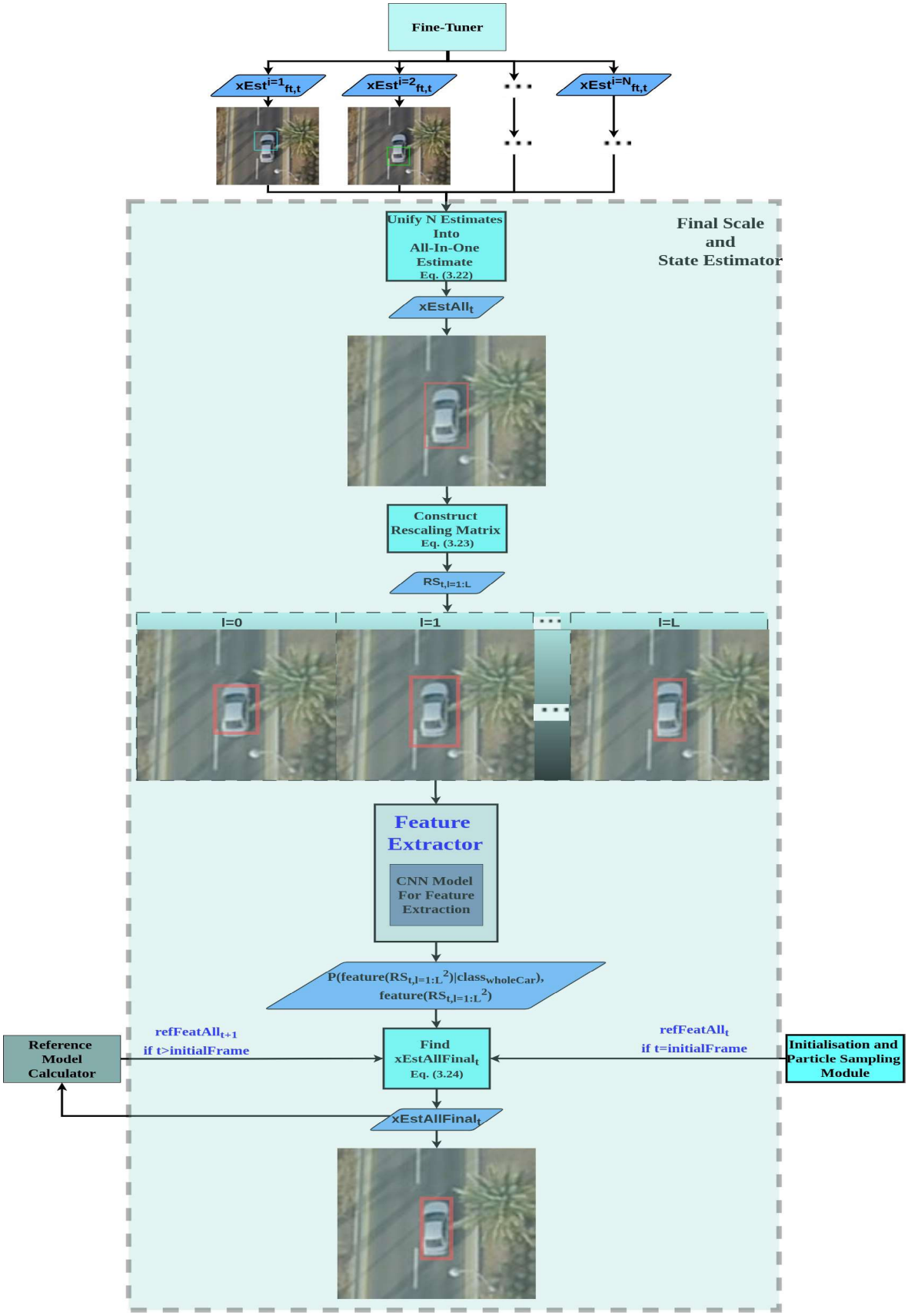


Figure 3.7. Unification module final scale and state estimator.

The final reference feature model for each part, $refFeatMat^{t+1}_{i=1:N}$, is then calculated as the moving average of the reference features from the past as

$$\begin{aligned} classProbVect^{t}_{i=1:N} &= [p(feature_{xEstFinal^{i=1:N}_{(t,:)}} \mid class_i)] \\ refFeatMat^{t}_{i=1:N} &= [feature_{xEstFinal^{i=1:N}_{(t,:)}}] \\ refFeat^{t+1}_{i=1:N} &= norm(classProbVect^{t}_{i=1:N})^T * norm(refFeatMat^{t}_{i=1:N})) \end{aligned} \tag{3.25}$$

where *norm* stands for the normalization. This is done by first calculating the reference features matrix, $refFeatMat^{t}_{i=1:N}$, and the class probability vector, $classProbVect^{t}_{i=1:N}$, for each part using outputs of the "CNN Feature Extractor"module. Additionally, the "Object State Estimation"module and the "Fine-Tuner"receive the updated reference features, which serve as a benchmark for the estimate. As shown in Figure 3.8, the state variables are updated with $xEstFinal^{i=1:N}_{t}$ in the interim and provided to the "Initialisation and Particle Sampling"module to sample the new particles in the following iteration.

3.1.4. Experiments

Aerial image sequences from a low-altitude UAV are used to build the UAV123 30 fps dataset. It contains 123 video segments totaling to 111.578 frames, each of which is shot for a very distinct scene. According to [12], the video sequences feature a wide range of objects, including people, cars, and boats, and the visual issues they provide are quite varied. Aspect ratio variation, background clutter, camera motion, fast motion, full or partial occlusion, illumination change, low resolution, out-of-view, similar object, scale variation, and viewpoint change are some of these issues. As ground truth, bounding box data is annotated and entered into text files for CNN training and performance assessment during tracking.

Only the image sequences of the cars and boats, labelled with "YES" in Table 3.1, are tailored to fine-tune CNN, and to test the tracking performance of the PFCNN. Those that contain severe full/partial occlusion and out-of-view problems, which are

out of scope of this chapter, are excluded. These will be considered in Chapter 5. Therefore, the specific image sequences including sudden scale variation, aspect ratio change, and rotation problems, as well as fast motion, viewpoint change, similar objects, and illumination variation, are particularly selected from the dataset.

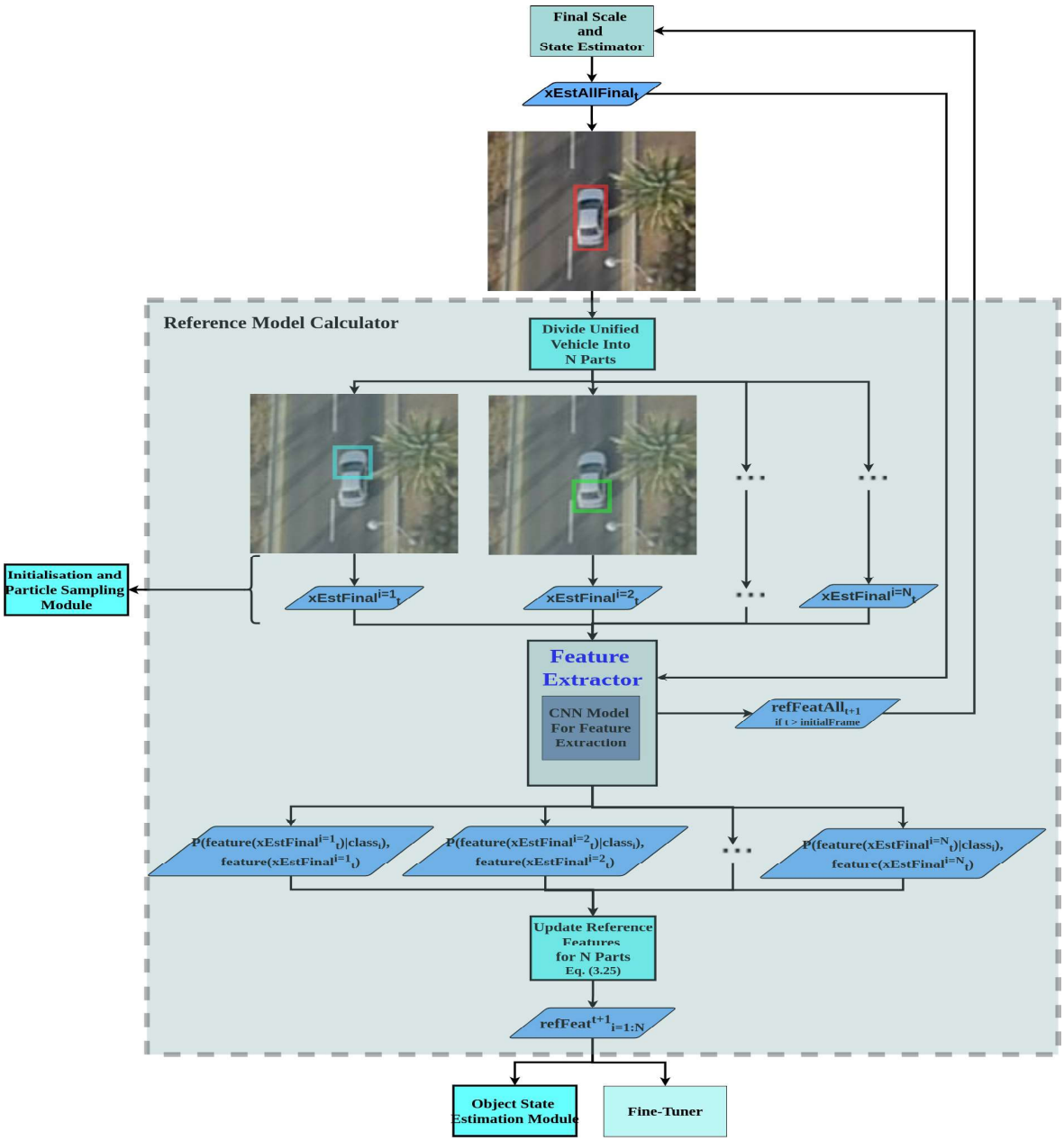


Figure 3.8. Unification module reference model calculator.

3.1.4.1. Particle Filter and CNN Training. The PFCNN achieves the highest tracking performances in the visual object tracking literature when compared to the best SOTA trackers, as demonstrated in [4], [13].

The evaluation method is supposed to be one pass evaluation (OPE) criterion, and measurements are taken from the start of the image sequence to the finish. There are numerous measures to assess the performance of visual trackers, with success and precision rates being the most commonly used in the literature. These metrics are based on the SOTA benchmarks [4], [15].

Table 3.1. Selected UAV123 image sequences for testing.

Name Of Sequence	Start Frame	Stop Frame	Is Tested?	Scale Variation	Aspect Ratio Change	Low Resolution	Fast Motion	Full Occlusion	Partial Occlusion	Out-Of-View	Illumination Variation	Viewpoint Change	Camera Motion	Similar Object
boat1	1	901	YES											
boat2	1	799	YES											
boat3	1	901	YES											
boat4	1	553	YES											
boat5	1	505	YES											
boat6	1	805	YES											
boat7	1	535	NO											
boat8	1	685	NO											
boat9	1	1399	NO											
car1_1	1	751	YES											
car1_2	1	877	NO											
car1_3	1	903	NO											
car2	1	1321	NO											
car3	1	1717	YES											
car4	1	1345	NO											
car5	1	745	YES											
car6_1	1	487	YES											
car6_2	1	1321	NO											
car6_3	1	1147	YES											
car6_4	1	973	YES											
car6_5	1	937	NO											
car7	1	1033	NO											
car8_1	1	1357	YES											
car8_2	1	1219	YES											
car9	1	1879	NO											
car10	1	1405	YES											
car11	1	337	NO											
car12	1	499	NO											
car13	1	415	NO											
car14	1	1327	NO											
car15	1	469	NO											
car16_1	1	415	NO											
car16_2	1	1579	YES											
car17	1	1057	NO											
car18	1	1207	YES											

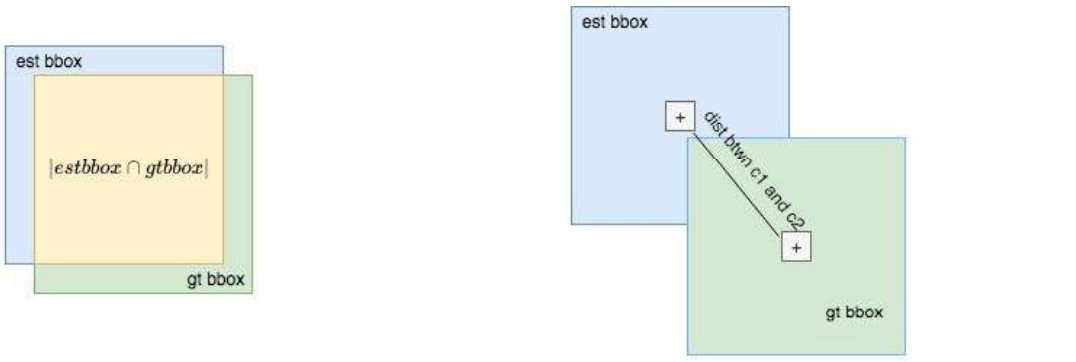


Figure 3.9. Performance metrics.

Figure 3.9 shows both measures. The area under the curve (AUC) is used to parameterize the success rate, which is determined by computing the percentage overlap between the estimated bounding box and the ground truth. Conversely, the precision rate is determined by measuring the separation between the centers of the ground truth data and the predicted bounding box. In this thesis, 20 pixels are taken as the reference since 20 pixels are commonly used in literature [4] to parameterize the precision results.

The working group websites [2], [4] provide the test results of the SOTA trackers against the image sequences in Table 3.1. Using a GPU and a lot of RAM, the PFCNN was trained and evaluated in the Google COLAB Pro environment. $M = 250$ (number of particles), $N = 2$ (number of parts), $K = 9$ (number of rescaled bounding boxes), $F = 25$ (number of fine-tuned estimates for each part), and $L^2 = 121$ (number of rescaled unified bounding boxes along both axes) are the pre-set hyperparameters of the PFCNN during experiments.

Variance for the process noise added to the location state variables is $\sigma^2_{proc,loc} = 300$, and $\sigma^2_{proc,vel} = 5$. The variance of the velocity is also utilized as a little jitter to sample the fine-tuned particles of estimations for the parts of the vehicle. Variance of the measurement noise is taken as $\sigma^2_{meas} = 0.00001$. Variance for the process noise is

added to the velocity state variables, to avoid an excessive computational load and to adequately sample the area surrounding the target, the number of particles is optimally chosen as $M = 250$. To improve the PFCNN's resilience to high, non-linear dynamic motions of the targets, the variance of the location, $\sigma^2_{proc,loc}$, is chosen extremely high.

VGG16 model is trained using Tensorflow backend of Keras API [89], and data augmentation is accomplished using a built-in function in Tensorflow. ADAM [90] is tailored as the optimizer with the following parameters: $\epsilon = 1e^{-}07$, $batchsize = 64$, $learning_rate = 0.00001$, $\beta 1 = 0.9$, $\beta 2 = 0.999$.

The suggested approach operates at about 0.1 fps. The tracking performance improves as the number of particles increases, in the expense of increased computational load. 250 particles were used during sampling to obtain the results, which will be elaborated in Section 6.1. Moreover, it is confirmed that the algorithm takes about two seconds to sample 50 particles and produce a final estimate.

3.1.5. Discussion

After DaSiamRPN [38], the PFCNN has the second-best overall precision and success scores [38]. It outperforms the best SOTA trackers like DSST [20], SRDCF [21], MEEM [24], and STRUCK [22,23] in [12], [4,13]. In particular, the overall success score of the PFCNN is nearly identical to that of the ECO.

The top-performing overall precision and success scores of the PFCNN are a result of the innovative scale estimation algorithm, the reference model calculator mechanism, the robust sampling approach, and the precise visual features of the PFCNN. The PFCNN produces extremely promising results for the uncommon partial occlusion and out-of-view sequences, while not being built against these scenarios, which include the occlusion and out-of-view visual difficulties. Furthermore, the PFCNN demonstrated remarkable resilience to changes in scale, aspect ratio, fast motion, camera motion, viewpoint, and similar objects on the scene.

3.1.6. Conclusion

The PFCNN focuses on resolving abrupt changes in scale, aspect ratio, visual model, target motion, and rotation for vehicle tracking in low-altitude UAV videos. The PFCNN, which combines a CNN and a particle filter, is robust to rotation, scale variation, changes in aspect ratio, fast motion, camera motion, viewpoint changes, and the existence of similar objects in the image. This new structure mimics the APF [7–9]. The particle rescaling module, which supports the core structure of the suggested technique, resizes each particle with a different aspect ratio to enhance tracking under scale and aspect ratio fluctuations.

The PFCNN also introduces the concept of a multi-part tracking framework to provide robust tracking against sudden rotations and fast motions. During rotation and fast motion, the aspect ratio of the vehicle changes abruptly while the motion vectors of the front and back parts of it show different directions. To be able to keep track of the diverse movement of the vehicle and to improve tracking performance against aspect ratio variation, abrupt rotational changes, and fast motion issues, this process is individually applied to different parts of the vehicle during tracking. Because of this, it is simpler to spot sudden variations in scale, aspect ratio, and rotation for parts than to spot them for the entire vehicle. In the last step, the "Unification"module estimates the final scaling parameter of the unified vehicle instead of the parts, updates the reference visual model of the parts, and resets the state variables that correspond to the width and height of the parts using the estimated sP.

The novel scale estimation scheme, the reference model calculator mechanism of the "Unification"module, the strong sampling strategy of the particle filter, and the accurate visual features provided by the CNN model are all key factors for the superiority of the proposed method over the UAV123@30fps dataset according to the overall precision and success results. While PFCNN is not developed against the image sequences which contain occlusion and out-of-view visual challenges, it gives very promising results for the rare partial occlusion and out-of-view sequences. Additionally,

the PFCNN shows exceptional adaptability to variations in view-point, fast motion, camera motion, scale, aspect ratio, and similar items on the scene.

For the time being, PFCNN outperforms all benchmark trackers in [4], [12, 13] explained at Sections 2.1.1 and 2.1.2, and shares the top-2 ranks with the DL based methods such as DaSiamRPN [38], explained in Section 2.1.2.3.

3.2. PFCNN versus APF

3.2.1. Introduction

In this section, it is verified theoretically as well as by simulations that PFCNN method [6] mimics the sequence of steps performed when optimal density is used. It is known that APF [7–9] mimics the steps performed when optimal density is used, and its weight update emulates the weight update procedure of Bayesian filters that use optimal importance density. It is also well known that choosing the importance density is a crucial step in the design and success of a particle filter [85]. There are only few cases where optimal importance density can be used since it is very hard or sometimes impossible to compute the optimal importance density. Thus, in practice, either suboptimal choices are used as the importance density or structures such as APFs, that mimic the sequence of steps when optimal importance density is available [85], are utilized. In this thesis, PFCNN [6] is presented as a new structure that mimics the sequence of steps when optimal importance density is available and is compared with APF. It is observed that PFCNN outperforms APF on UAV videos where non-linear, high-dynamic motion is coupled with abrupt visual model variations. The key part of improvement is the compensation for possible miss-calculation of likelihood probabilities by class probabilities which are calculated by CNN. These class probabilities are tailored during the weight update process of the particle filter.

The evolution of APF and the analogy between PFCNN and APF are discussed in Section 3.2.2. In Section 3.2.3, the performance comparison of both methods is

presented and the discussion and conclusions are summarized in Sections 3.2.4 and 3.2.3, respectively.

3.2.2. Architecture

3.2.2.1. Evolution of APF. As stated in Section 3.1.3.1, $\mathbf{z}_{t,j}^{i}, t = 0, 1, 2, ..., T$, is a measurement vector that is an observable time series from time $t = 0$ up to time T. It is independent, and conditioned on $\mathbf{x}_{t,j}^{i}$, unobserved state variable. The goal is to estimate the joint filtering density, $p(x_{t=0:t}|z_{t=0:t})$, or marginal joint pdf, $p(x_t|z_{t=0:t})$ to get the information about the state variable depending on present observations. In general, the observations are the noisy versions of the state variables. APF [7–9] mimics the steps performed when optimal density is used, and its weight update emulates the weight update procedure of Bayesian filters that use optimal importance density. To be able to compare the performance of the PFCNN against APF, APF is applied to the same object tracking problem in UAV videos as PFCNN is applied. The features extracted from CNN model are also used in APF implementation as measurements to update the weights of auxiliary and original particles in APF. During the derivations, i is the number of parts, and will be dropped for simplicity. j represents the number of particles. The evolution of APF is elaborated in the following paragraphs in a similar fashion that the theoretical background of SIR PF is explained in Section 3.1.

Bayes theorem is shown in Equation (3.7), and when we apply it to the joint posterior pdf, we get a recursive relation for the joint pdf updates as shown in Equation (3.8). Keep in mind that the recursive relation of joint posterior pdf can also be factorized as

$$\begin{aligned} p(x_{0:t}|z_{0:t}) &= p(x_t|x_{0:t-1}, z_{0:t})p(x_{0:t-1}|z_{0:t}) \\ &\neq p(x_t|x_{0:t-1}, z_{0:t})p(x_{0:t-1}|z_{0:t-1}). \end{aligned} \tag{3.26}$$

Conditioning the previous values of the state variables until time $t-1$ onto observations in the future that span until time t is impossible. However, changing the time tag of

observation at time t to $t-1$ breaks the equality. On the other hand, during the choice of importance density, the inequality in Equation (3.26) is ignored, and it is chosen in a recursive format as explained in Equation (3.9). However, the inequality condition prevents importance density from being optimal without being adapted to the present measurement. That is why it is not feasible to choose the optimal importance density as $p(x_{0:t-1}|z_{0:t})$, and it is best to choose a pdf that can converge to the optimal importance density.

Thus, to approximate optimal density, at time t, choosing an approximation of the past posterior pdf, $p(x_{0:t-1}|z_{0:t-1})$, given in Equation (3.26) would be a good starting point. Derivation of $p(x_{0:t-1}|z_{0:t})$, called smooth posterior, is shown as

$$\begin{aligned} p(x_{0:t-1}|z_{0:t}) &= \int p(x_{0:t}|z_{0:t})dx_t \\ p(x_{0:t}|z_{0:t}) &\propto p(z_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|z_{0:t-1}) \\ &\propto \int p(z_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|z_{0:t-1})dx_t \\ &= p(x_{0:t-1}|z_{0:t-1})\int p(z_t|x_t)p(x_t|x_{t-1})dx_t \end{aligned} \tag{3.27}$$

where $p(z_t|x_t)$ and $p(x_t|x_{t-1})$ are likelihood and the transition probabilities respectively, while $p(x_{0:t-1}|z_{0:t-1})$ represents the past posterior pdf. When we substitute the approximation of the past posterior pdf, given in Equation (3.5), in place of the joint posterior pdf at time t in Equation (3.27), the smooth posterior pdf is represented in terms of the weighted delta functions as

$$\begin{aligned} p(x_{0:t-1}|z_{0:t-1}) \approx \hat{p}(x_{0:t-1}|z_{0:t-1}) &= \sum_j^M w_{t-1,j}\delta(x_{0:t-1} - x_{0:t-1,j}) \\ p(x_{0:t-1}|z_{0:t}) &\propto [\sum_j^M w_{t-1,j}\delta(x_{0:t-1} - x_{0:t-1,j})]\int (z_t|x_t)p(x_t|x_{t-1})dx_t \\ &= \sum_j^M b_{t-1,j}\delta(x_{0:t-1} - x_{0:t-1,j}) \\ &\textit{where}, \\ &b_{t-1,j} = w_{t-1,j}\int p(z_t|x_t)p(x_t|x_{t-1,j})dx_t. \end{aligned} \tag{3.28}$$

The particles that we use to estimate the past posterior pdf, are also utilized to estimate the smooth posterior pdf, $p(x_{0:t-1}|z_{0:t})$. But the weights, $b_{t-1,j}$, are altered as shown in Equation (3.28), and they are called "Auxiliary or Parent Weights". After the derivation of the smooth posterior pdf in terms of parent weights, the importance pdf in Equation (3.9) can be written in terms of parent weights as

$$\begin{aligned} q(x_{0:t}|z_{0:t}) &= q(x_t|x_{0:t-1}, z_{0:t})q(x_{0:t-1}|z_{0:t}) \\ &\approx q(x_t|x_{0:t-1}, z_{0:t}) \sum_{j}^{M} b_{t-1,j}\delta(x_{0:t-1} - x_{0:t-1,j}). \end{aligned} \tag{3.29}$$

Then, the relation between smooth importance pdf and the past posterior pdf, presented in Equation (3.5), can be expressed as

$$\begin{aligned} q(x_{0:t-1}|z_{0:t}) &\approx \sum_{j}^{M} b_{t-1,j}\delta(x_{0:t-1} - x_{0:t-1,j}) \\ &\approx \frac{b_{t-1}}{w_{t-1}} p(x_{0:t-1}|z_{0:t-1}). \end{aligned} \tag{3.30}$$

After the deriving of the posterior pdf, $p(x_{0:t,j}|z_{0:t})$ in Equation (3.8) and the importance pdf, $q(x_{0:t,j}|z_{0:t})$ in Equations (3.29) and (3.30), the weight update equation in Equation (3.6) can be expressed in a recursive format, as

$$w_{t,j} \propto \frac{w_{t-1,j}}{b_{t-1,j}} \frac{p(z_t|x_{t,j})p(x_{t,j}|x_{t-1})}{q(x_{t,j}|x_{t-1}, z_t)} \tag{3.31}$$

where weights at time t are updated using original weights and auxiliary/parent weights at time $t-1$. Then, if we choose the importance pdf as transition pdf, $p(x_t|x_{t-1})$ in Equation (3.31), the weight update equation in Equation (3.31) simplifies to

$$\begin{aligned} q(x_{t,j}|x_{t-1}, z_t) &= p(x_t|x_{t-1}) \\ w_{t,j} &\propto \frac{w_{t-1,j}}{b_{t-1,j}} \frac{p(z_t|x_{t,j})p(x_{t,j}|x_{t-1})}{q(x_{t,j}|x_{t-1}, z_t)} \\ w_{t,j} &\propto \frac{w_{t-1,j}}{b_{t-1,j}} p(z_t|x_{t,j}). \end{aligned} \tag{3.32}$$

In the PFCNN model, the importance pdf is also chosen as the transition density, and the weight update process is simplified. But with this choice, the approximate optimal solution is lost. The weight update process of APF is elaborated in Equation (3.32). Depending on the evolution of the method, first of all, parent particle trajectories with probability b_{t-1}^{i} should be sampled in the same manner that resampling in the particle filter is achieved. At this point, the difference between APF and SIR PF is that APF utilizes the most recent measurement to sample the new particles at time t. In traditional SIR PF, the transition density does not depend on the present measurement at time t, $\mathbf{z}_t$, and that is the reason for calling SIR PF "not adapted". Hence, APF specifically has two useful properties;

- "Presampling": Using $\mathbf{z}_t$ so that we sample the best past trajectories at time $t-1$
- "Particle Sampling ": Using the state dynamics, we propagate the particle value from $t-1$ to t.

Presampling is used to presample the parent trajectories with probability $b_{t-1,j}$ as

$$\begin{aligned} q(x_{0:t}|z_{0:t}) &= q(x_t|x_{0:t-1}, z_{0:t})q(x_{0:t-1}|z_{0:t}) \\ &\approx p(x_t|x_{t-1}) \sum_{j}^{M} b_{t-1,j}\delta(x_{0:t} - x_{0:t,j}) \end{aligned} \tag{3.33}$$

where it drives the particles to the spaces with higher probabilities under high high-dimensional posterior pdf. But computing $b_{t-1,j}$ based on Equation (3.28) is very hard because of the high-dimensional integral. That is why, the transition density, $p(x_t|x_{t-1,j})$, is commonly chosen as

$$p(x_t|x_{t-1,j}) \approx \delta(x_t - f(x_{t-1,j})). \tag{3.34}$$

Here, $f(x_{t-1,j})$ can be modeled as a random walk in the first stage of the filter, and the transition density represents a deterministic model without a process noise. Impulse functions enable the approximation of the integral.

State variable of particles at time $t-1$, $x_{t-1,j}$, are updated with additive Gaussian noise of small variance, σ^2_{proc} to predict the auxiliary particles, $\mu_{t,j}$, and they can be expressed as

$$\begin{aligned} \mu_{t,j} &= \mathbf{x}_{t-1,j} + r_{t,j} \\ r_{t,j} &= N(0, \sigma^2_{proc}) \end{aligned} \tag{3.35}$$

where auxiliary particles, namely $\mu_{t,j}$'s, are treated as auxiliary state variables. This way, almost constant outputs are obtained in a similar fashion with PFCNN [6] as given in Equation (3.35). Then the transition pdf can be approximated with delta functions as

$$p(x_t|x_{t-1,j}) \approx \delta(x_t - \mu_{t,j}). \tag{3.36}$$

Depending on Equations (3.33-3.36), it can be understood that APF tailors measurements to weigh the sampled auxiliary particles around the target trajectory before the conventional sampling and estimation processes of the particle filter. Using the approximation in Equation (3.29) and the random walk model in Equation (3.35), $b_{t-1,j}$'s in Equation (3.28) can be expressed as

$$b_{t-1,j} = w_{t-1,j} p(z_t|f(x_{t-1,j})). \tag{3.37}$$

Using the definition of auxiliary particle weights in Equation (3.37) together with Equation (3.31), and taking the importance density as the transition density, weights can be expressed as

$$w_{t,j} \propto \frac{p(z_t|x_{t,j})}{p(z_t|\mu_{t,j})}. \tag{3.38}$$

If a random walk model with high noise variance is used, sampling generates auxiliary particles with very low weights during the first phase of the algorithm; this may lead the particles to vanish. On the other hand, if a random walk model with low noise variance is used to sample auxiliary particles while a target with a high,

non-linear motion vector exists, particles will most probably be sampled around a very small area and will have low weights, leading to the death of particles. Under both conditions, serious problems in target tracking are faced, and target tracking will not be possible. This is the weakness of the APF, and the way PFCNN compensates for this problem is explicated in Section 3.2.2.3.

3.2.2.2. Implementation. Pseudo-code of APF is shown in Figure 3.10, and its implementation is explained with a one-dimensional data to represent the posterior distribution that APF constructs at the end of each iteration for better understanding.

At the beginning of the iteration, the particle states, $x_{t-1,j}$, are initialized with uniform weights, and the number of particles, i, are taken as $N = 10$ as shown in Figure 3.10a. After the measurement, z_t, is observed at time t, APF update phase is achieved. At this phase, auxiliary particles, $\mu_{t,j}$, are sampled from a prior distribution, $p(x_{t,j}|x_{t-1,j})$. Then, likelihoods, $p(z_t|\mu_{t,j})$, for each particle is calculated. At the final step of this phase, the auxiliary weights, $b_{t-1,j}$, are updated by multiplying the likelihoods with the weights from the previous iteration. Meanwhile, new indices, j, are resampled depending on the updated values of auxiliary weights, belonging to the propagated, $\mu_{t,j}$, and auxiliary weights are reinitialized as shown in Figure 3.10b. After the new indices are determined, original particle states at time $t-1$, $x_{t-1,j}$, $j = 1, 2, ..., c$, with new indices, are propagated to $x_{t,j}$, and the likelihoods for original particles, $p(z_t|x_{t,j})$ are calculated. Then, as the final weight update process is achieved as given in Figure 3.10c, the posterior is constructed for the state estimation.

Pseudo code for APF, in Figure 3.10, is applied to the tracking problem to prove that the PFCNN mimics the APF in terms of the particle sampling from optimal importance density. The overall method for APF is shown in Figure 3.11 to emphasize the important differences compared to the PFCNN architecture, and these differences can be itemized as

(i) “Auxiliary Particle Presampling”module is utilized to sample auxiliary particles at the beginning of the iteration as explained in Section 3.2.2.1.

(ii) “Class Probabilities of Particles ”output of “CNN Model For Class Probability”module in Figure 3.1, utilized as $p(z_t|x_{t,j}, class)$ in updating particle weights in Equation (3.18) for SIR PF in PFCNN is ignored for APF. Because $p(z_t|x_{t,j}, class)$ is not utilized to update the weights of particles of APF at “Object State Estimation ”module as shown in Equation (3.38).

On the other hand, APF uses the output features of the “CNN Model For Feature Extraction”as a measurement to be able to calculate the likelihoods for auxiliary and original weights, as SIR PF in PFCNN does for its unique weights. “Unification”module is a common function, used by both methods.

3.2.2.3. Analogy Between APF and SIR PF in the Proposed PFCNN Model. An analogy between the weight update equations of PFCNN and APF, given in Equations (3.18) and (3.38), respectively, can be established as

$$\frac{1}{p(z_t|\mu_{t,j})} = w_{t-1,j}p(z_t|x_{t,j}, class). \tag{3.39}$$

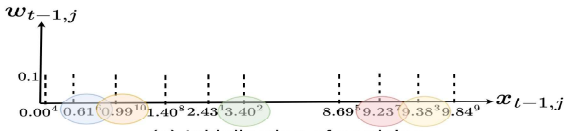
It is seen from Equation (3.39), the likelihood of auxiliary particles in APF, namely $p(z_t|\mu_{t,j})$, are in the denominator, and they are related to the product of the class probability scores ($p(z_t|x_{t,j}, class)$) of PFCNN, and the non-updated weight values ($w_{t-1,j}$) from the previous iteration. On the other hand, probability terms on both sides of Equation (3.39) have a similar purpose. In the PFCNN method, even though particles are sampled from transition density without knowing observation, z_t, at time t, the term, measuring how much the sampled particles resemble observation at present time in terms of class probability, is determined by ($p(z_t|x_{t,j}, class)$), and included into weight update process. In APF, observation, z_t, at time t, is considered to sample auxiliary particles along the target trajectory, and then the actual particles are guided through the regions with a high probability of the posterior pdf.

Auxiliary Particle Filter Implementation Pseudo Code

1. Initialize the particles with uniform weights,

$$x_{t-1,j}, j = 1:10$$

(x is 6-dim state vector, but its first component of x is used for presentation)



(a) Initialization of particles

2. Target moves, t=t+1, t=0, 1, ..., T
Get measurement,

$$z_t$$

3. Auxiliary Particle Filter Update,
for j=1:10
a. sample Auxiliary or Parent Particles

$$\mu_{t,j}$$

from

$$p(x_t|x_{t-1,j})(characterization)(Eqs.\ 3.34 - 3.36)$$

b. calculate likelihoods for each auxiliary particle

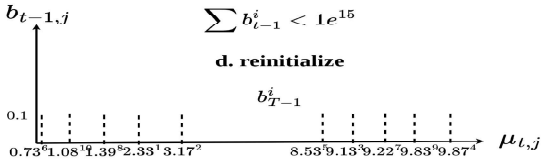
$$p(z_t|\mu_{t,j})$$

c. do auxiliary particle weight update

$$b_{t-1,j} = w_{t-1,j} * p(z_t|\mu_{t,j})(Eq.\ 3.37)$$

if

$$\sum b^i_{t-1} < 1e^{15}$$



(b) Re-Initialization of auxiliary particles

4. Resample indices

$$j^c$$

from propagated particles,

$$\mu_{t,j}$$

new indices = [9, 6, 5, 1, 5, 6, 5, 1, 2, 5]
(add one to indices, samples start from 1 to 10)

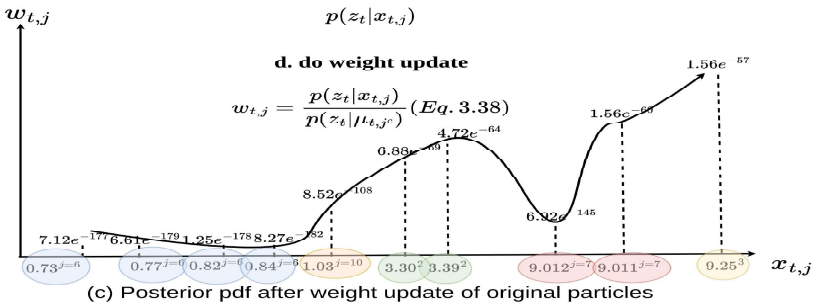
5. Propagate the state of all particles indices that survived and update particle weights for state estimation,
for j in new indices (c):
a. get particle state using idx from original particles set

$$x_{t-1,j}$$

b. propagate sample to find

$$x_{t,j} \approx p(x_{t,j}|x_{t-1,j})$$

c. calculate likelihoods for each original particle



(c) Posterior pdf after weight update of original particles

Figure 3.10. The pseudocode of APF with evolution of posterior pdf.

Another point of view to understand the convergence level of both methods is to investigate the way how posterior pdf is constructed. APF steers both auxiliary and parent particles through the regions, where the weights are alive or not-degenerated. A similar functionality in SIR PF, as a part of PFCNN, is achieved via class probability scores, strengthening the particles with high resemblance with the current measurement, and killing the particles with less similarity. Hence, PFCNN [6], explained in Section 3.1, functions as a kind of APF but achieves this in a distinctive way from APF while contributing to the posterior density differently.

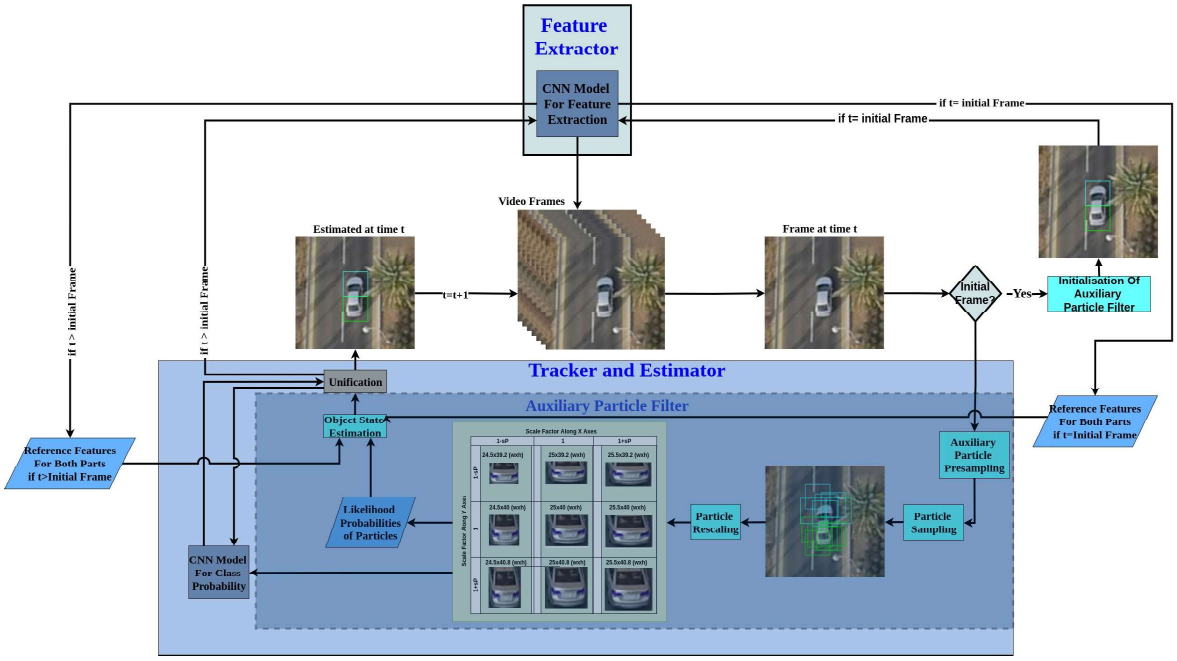


Figure 3.11. Overall method of APF

3.2.3. Experiments

PFCNN [6], explained in Section 3.1, and APF are both applied to the target tracking problem in the same videos from UAV123 dataset, and compared in terms of precision and success criteria using the OPE metric as done in PFCNN [6]. The parameters and hyperparameters, utilized for particle filter and CNN in 3.1.4.1, are kept the same during APF experiments. As a difference between APF and PFCNN, during the presampling of APF, the number of parent particles, M, is taken as 175,

and the variance of process noise, $r_{t,j}$, added to characterization function in Equation (3.35) is taken as 10.

For this purpose, 16 videos, including linear and non-linear target motion are selected. The state variable is six-dimensional and constructed by the location, the velocity, and the scale factor of the bounding box along x and y axes. Both methods were compared based on their corresponding precision and success scores, and these scores were constructed from three different points of view; 1. Overall, 2. Attribute-Based, and 3. Scenario-Based, and the details are given in Section 6.2.

3.2.4. Discussion

Based on the results, PFCNN [6], explained in Section 3.1, surpasses APF in terms of both precision and success scores. These results can be evaluated as the average performance of methods against all visual problems faced during all videos. Hence, it can be concluded that PFCNN mimics the case where optimal density is used, like APF. However, it does this differently compared to the procedure APF uses.

3.2.5. Conclusion

In conclusion, while PFCNN and APF mimic the sampling scheme of the particle filter that uses the optimal importance density, PFCNN outperforms APF, when used for target tracking under challenging scenarios, and when target and camera motion are non-linear with high dynamics. A random walk model with high noise variance is used; presampling in APF generates auxiliary particles with very low weights during the first phase of the algorithm; this may lead the particles to vanish. On the other hand, if a random walk model with low noise variance is used to presample auxiliary particles while a target with a high, non-linear motion vector exists, particles will most probably be sampled around a very small area, and will have low weights, leading to the death of particles. Under both conditions, serious problems in target tracking are faced. Hence, the contribution of the PFCNN is provided by utilizing the class

probability term in the weight update process. Eventhough, both PFCNN and APF mimic the case where the optimal density is utilized, the use of the class probability, calculated by CNN, is the novelty of the structure, and the key point that paves the way to its success.

4. A NOVEL APPROACH TO UAV DATASET CREATION

4.1. Introduction

CNN structures need extensive datasets for training and testing. This chapter is devoted to synthetic UAV dataset creation. In Section 4.2, the development of the proposed synthetic UAV dataset, so-called S-UAV [10], is explained in detail. In Section 4.3, a summary of the creation of segmentation ground truth data for the UAV 123 Dataset [2] is elaborated.

4.2. A Novel, Synthetic UAV Dataset Creation Approach:S-UAV

As explained in Section 2.2.2, CNN models, designed to detect optical flow, occlusion, as well as segmentation and classification, require extensive synthetic or real-time data sets for training and testing. To the best of our knowledge, there is no UAV dataset, including ground truths for optical flow, occlusion, segmentation, and classification at the same time, in the literature. Creating optical flow data from real-time scenarios requires an extensively controlled environment in terms of measuring device set-up from camera sensors. In the meantime, creating sunshine, rain, and snow in desired instances may not always be possible.

Considering that the creation of controlled environments and weather conditions in question is very difficult, and sometimes even impossible. Utilizing graphics cards with high processing capabilities stands out as a faster, more logical, and more accurate solution for creating synthetic or virtual environments with game engines. These virtual environments can be easily prepared using game engines like Unity [91] or Unreal, where realistic models can be added, and occlusion and segmentation data can also be extracted along with optical flow vectors. With the reference data obtained, CNN models can be trained to create models that can perform both optical flow and overlap detection simultaneously [65, 66] and [77].

In this context, it has been decided to create labeled synthetic videos of UAVs with various weather conditions, different backgrounds or landscapes, and a wide variety of vehicle models as targets, leading to the development of the proposed method. For the first time in literature, a unique UAV synthetic dataset has been produced, which includes reference optical flow, occlusion, and segmentation images as well as classification data at the same time. Using this dataset, CNN models, capable of predicting optical flow, occlusion, segmentation, and classification for targets or vehicles present in the scene, can be developed and trained. Thus, the gap in the literature regarding UAV synthetic datasets has been filled.

As shown in Figure 4.1 [10], in the first stage, a three-dimensional UAV virtual world is created using the Unity engine. Then, when the engine is run, color video images are first generated, followed by video images containing vehicle mask information (.png), along with a file that contains triangle (mesh) information related to the vehicles (.json). Afterwards, reference optical flow vectors (.flo, .png) are extracted using the triangle file and the masked images. During the optical flow extraction, the homography matrix between the triangles, which are automatically generated by the Unity engine for consecutive images stored in the triangle file, is determined, and the position vectors for all pixels within the triangles are transformed using the homography matrix to identify their locations in the other image coordinates. Subsequently, boundary information in COCO format is produced using the masked video frames. Images of resolution of 1920x1080 are obtained; however, due to the large file sizes of the generated data and potential size issues that may arise during training in CNN models, the data can be downscaled and converted to any size depending on the user's preference. Additionally, in the proposed method, a dataset with optical flow, occlusion, and segmentation reference images of 250x250 size has also been produced.

The details of the proposed dataset are presented in the subsequent sections. In Section 4.2.1, the features of the Unity engine and the method for preparing a virtual world are explained in detail. The method for creating the ground truths of optical flow, occlusion, segmentation, and classification for the objects or vehicles in the synthetic

dataset is explained in Section 4.2.2. While the creation process of the segmentation ground truth for the UAV 123 dataset is explicated in Section 4.3, Section 4.4 discusses the contributions of the proposed method and future work.

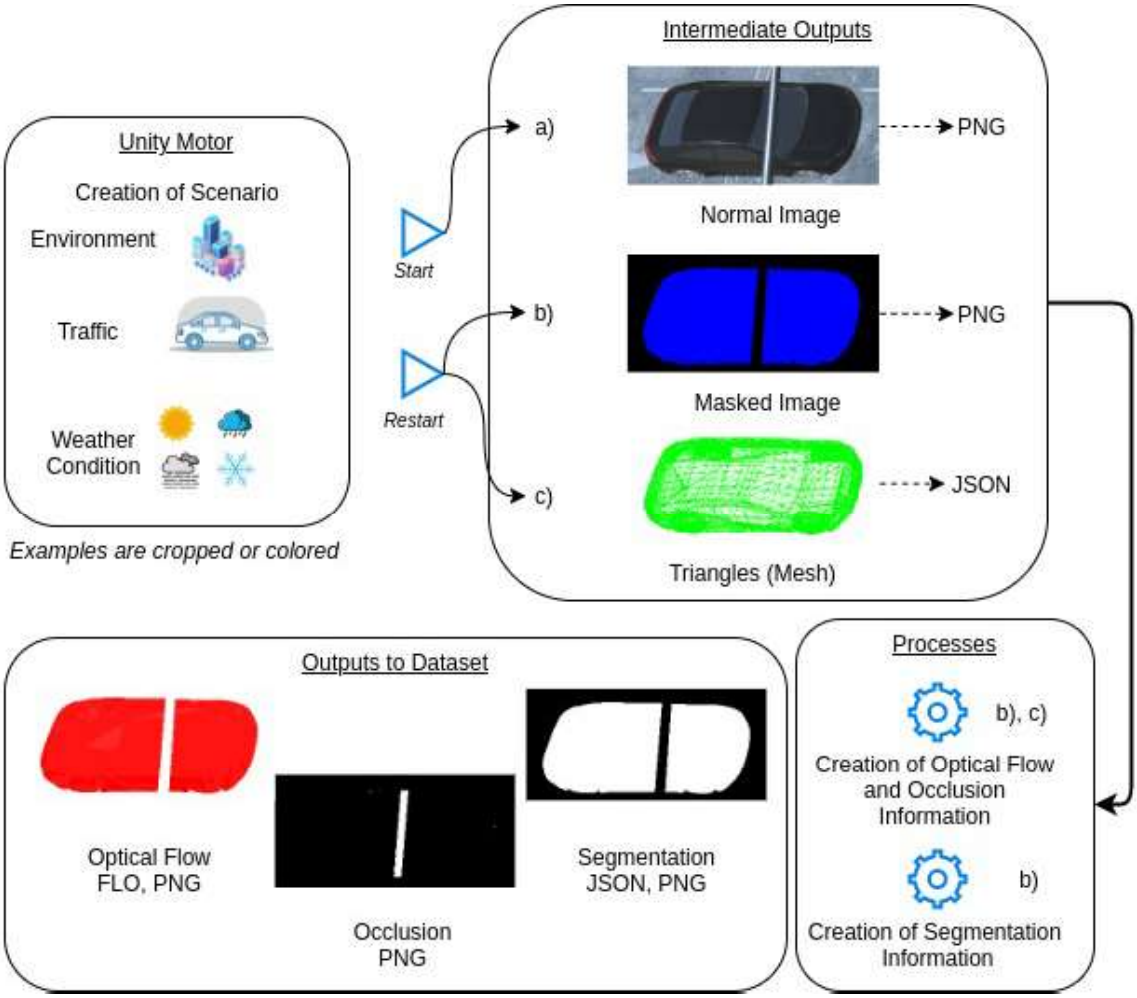


Figure 4.1. General method to create S-UAV dataset.

4.2.1. Unity Motor and Models

The Unity engine is an extremely user-friendly game development engine [91]. It is easy to learn, and can be used on any platform, utilizing the C# programming language. Additionally, thanks to user support, ready-made vehicle models are available. These models are offered by their manufacturers with high graphical quality, available for both paid and free options. In the proposed method, various world and vehicle models have been obtained, ready for creating a virtual world. Before using these models,

they are subjected to a series of optimization processes because some vehicle models contain interior details that are not considered important in this method. Therefore, they are removed to improve the speed of the system. For these operations, the Blender application [92] is used to rearrange the models. Subsequently, the models intended for use are integrated with the system, making them ready for scenario setup.

4.2.1.1. Modeling the Virtual World with the Unity Engine. In the proposed method, a Unity virtual world represents the scenario that is intended to be modeled or created. Scenarios can be divided into segments based on the length of the videos. Hence, a bunch of scenarios with different vehicles are obtained by segmenting a single video, and they are independent from each other.

In order to distinguish vehicles from one another or from other non-vehicle objects, different colored pixels have been assigned to each vehicle, thereby giving them unique identities. Consequently, it is possible to determine the location of all pixels belonging to a vehicle and whether they occlude with another object or vehicle. All objects outside the vehicle are represented in black. This method allows the extraction of the boundary information of the vehicle.

The Unity Recorder library has been utilized in the system to record images. This library can dynamically generate the requested video information for a specified duration, during the selected time frame, and at a defined frame size. Video images are saved with names corresponding to the number of images at 30 fps. The scenario naming policy is explained with an example in Figure 4.2, and the examples of synthetic video images produced in response to different landscapes and weather conditions are presented in Figure 4.3.

4.2.1.2. Unity Vehicle Mask/Segmentation Information. Unique identifiers are required to track the positions of a vehicle in video images. Therefore, the types and quantities of vehicles are selected based on “RGB”color codes within the range of 0 to 255. The

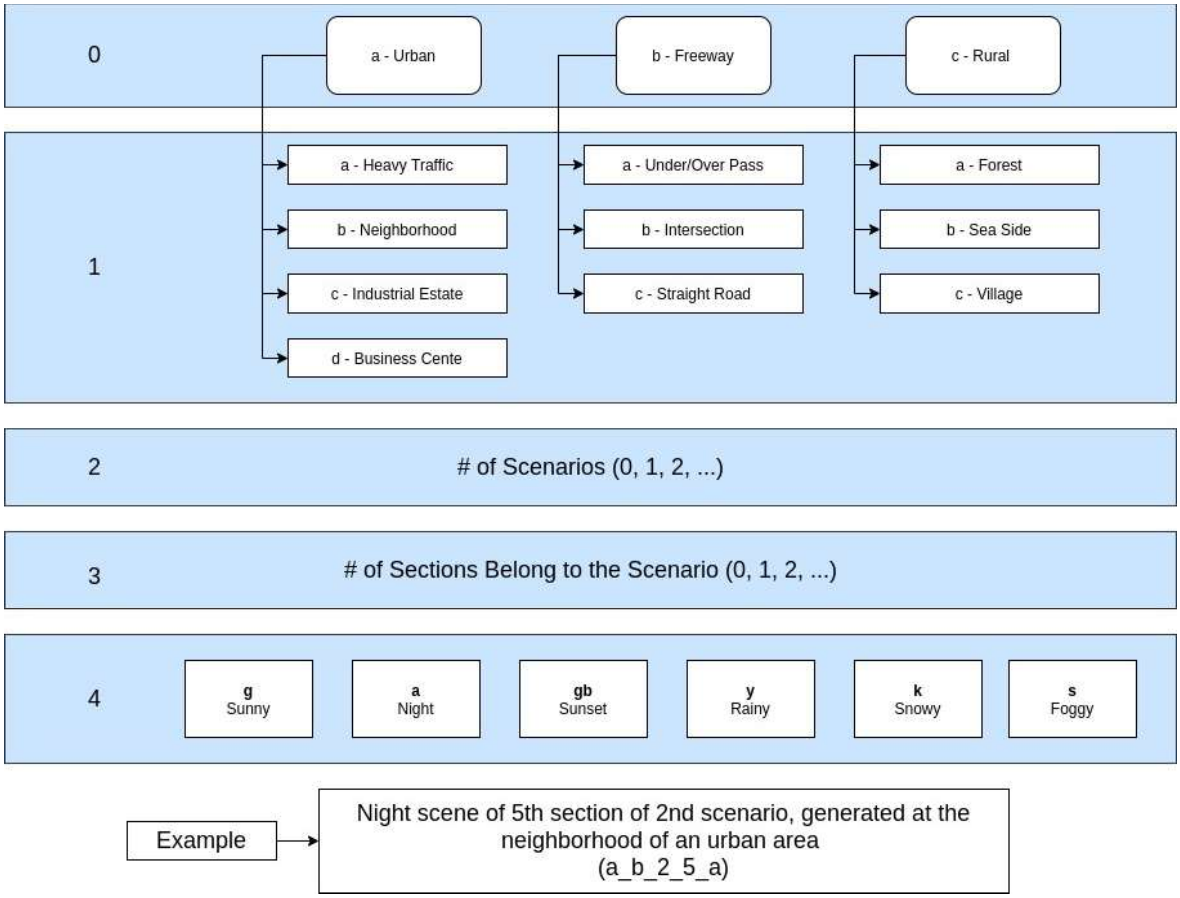


Figure 4.2. Scenario naming policy.

"G"value represents the type of the vehicle, and it is categorized into five classifications: bus, car, truck, lorry, and minibus. If there is more than one vehicle of the same type on the scene, this information is stored in the "B"value. The "R "value is kept fixed as zero. These values are recorded as the mask information for a vehicle. This will facilitate the extraction of optical flow, occlusion, and segmentation information, as well as the identification and tracking of vehicles.

4.2.1.3. Unity Vehicle Location Information. The Bezier library has been used for the movements of vehicles that have been prepared for use in the system within the virtual world. Using this library, a path or trajectory must be prepared for moving objects. While vehicles or the camera can follow the created path at a predetermined constant speed, multiple vehicles can also be added by the user to move along this path with



Figure 4.3. Sunny and snowy conditions of urban and rural scenarios.

a predetermined fixed time interval. When recording the video, the positions of the triangles in the 3-D world (i.e the vehicle models shown in Figure 4.1) are transformed into two dimensions. Then, they are placed onto the video image. This allows the determination of the position of the vehicle in the image coordinate plane. Subsequently, the triangle data created in 2-D for vehicles, along with their mask information, is converted into json format and recorded into the dataset. However, the triangle data consists of floating-point (float) variables. Since floating-point variables cause errors during the homography process, they are converted to integer format prior to homography processing. Consequently, due to number conversions, a few pixel errors may occur on the vehicles after obtaining optical flow and occlusion information. These erroneous pixels have been minimized by rounding to neighboring pixel values or using filters. Experiments have shown that a few pixel errors are negligible in terms of the use of the dataset in CNN training and their impact on performance as published in [10], [11].

4.2.2. Creation of Optical Flow, Occlusion and Segmentation as Ground Truth

To create the optical flow, occlusion, and segmentation reference dataset, masked video images and triangle (mesh) data are required. For this purpose, the system first generates optical flow (.flo, .png) information of resolution of 1920x1080, followed by the sequential generation of occlusion (.png) and segmentation (.json, .png) data, which are then copied to the appropriate files.

4.2.2.1. Creation of Optical Flow and Occlusion Dataset Using Homography. Horizontal and vertical motion vector values of each vehicle need to be found or previously known to obtain optical flow files. The triangle (mesh) data of the vehicles are used to obtain these values as shown in Figure 4.4, and as elaborated in Appendix A.

The optical flow vectors, obtained as explained, belong to only a single triangle. The vertical and horizontal vector values are recorded at the locations corresponding to the triangles in the T_n image in the vertical and horizontal planes. While T corresponds to the image of a triangle from a vehicle in the dataset, n represents the time or the number of the frame in the video sequence for the same triangle. The optical flow image created by taking the vector sum of the vertical and horizontal optical flow vectors is presented in Figure 4.5. To generate the optical flow vector for all pixels belonging to the vehicle, the operations described in Figure A.1 are applied iteratively for all triangles that make up the vehicle, resulting in the overall optical flow image presented in Figure 4.6.

If a pixel or pixels belonging to a triangle in the T_n image, represented as $[x_n^i, y_n^i]$, do not correspond to or coincide with a pixel location that is projected to the associated triangle in T_{n+1} after the application of the homography, or if the projected pixels fall outside the image coordinate boundaries of the triangle in T_{n+1}, it is assumed that the pixel is not present in the T_{n+1} image coordinates, in other words, it is considered to be subjected to occlusion. The occluded vector or value for the pixels is recorded by

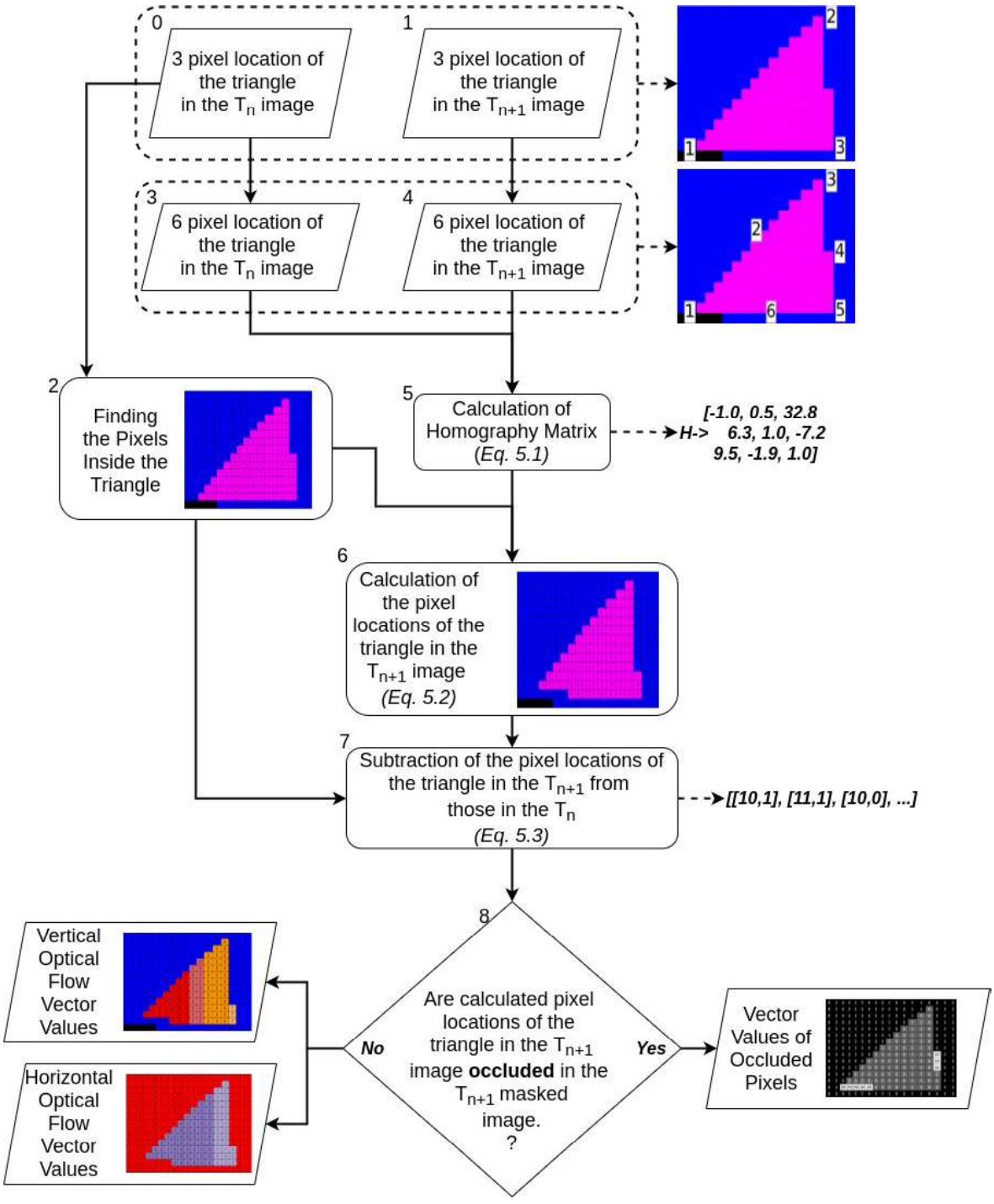


Figure 4.4. Calculation of optical flow vectors.

changing the corresponding pixel location in the triangle of the T_n video image to a color code of 255. However, the triangle information obtained from Unity is converted from a 3-D world to real numbers in the camera frame. During the homography process,

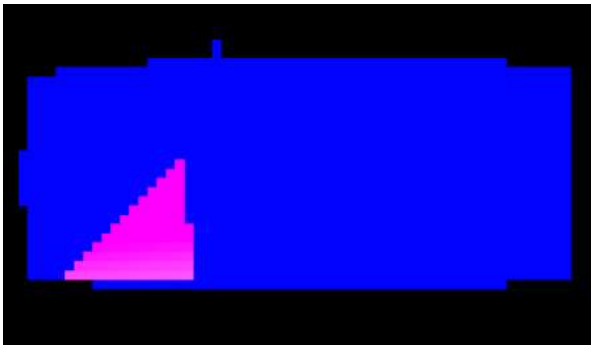


Figure 4.5. Combined image of horizontal and vertical optical flow vectors.

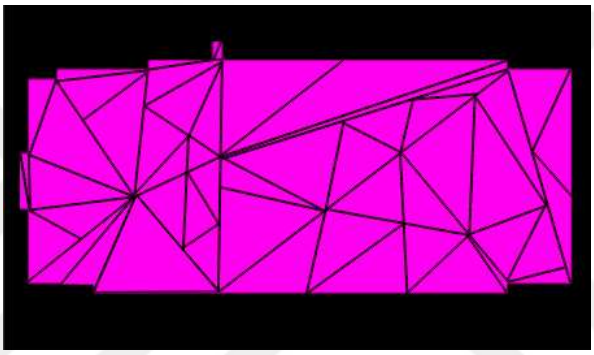


Figure 4.6. Final optical flow image.

it was converted to integer variables. Therefore, some information loss may appear as errors outside the vehicle. This situation causes the formation of erroneous white pixels in the occluded vector and consequently around the vehicle. To minimize this error, the Canny edge detector is applied to the images with resolution of 1920x1080, and theerosion technique is applied to the images with resolution of 250x250.

After the completion of the operations performed on the triangle in the T_n image in relation to the corresponding triangle in the T_{n+1} image, the same operations are applied iteratively to all triangles that make up the vehicle. Once the triangles associated with the processed vehicle are evaluated, this process is also repeated for all vehicles in the T_n image. When there are no other vehicles in the T_n image, the horizontal and vertical optical flow vectors are combined for each vehicle and converted from the FLO format into an optical flow file. At the same time, the overlap vector is saved as an

image in PNG format. As a final step, the time index n is incremented to read the next image pair, and all operations are repeated.

Using the approach described in Figure A.1, the optical flow vectors calculated for a car are recorded as a FLO file and visually presented in Figure 4.7. While the vehicle moves downwards, it also turns to the left. The variable motion observed in different parts of the vehicle, due to the turn, causes variability in the optical flow vectors. This change manifests itself as a change in color. The optical flow occurring from top to bottom is represented in yellow, while lateral movement from right to left is associated with blue. Because the motion develops as a resultant, there are green and various shades of intermediate color codes in different parts of the vehicle. Therefore, it can be observed that the pixels have independent optical flow vector values.

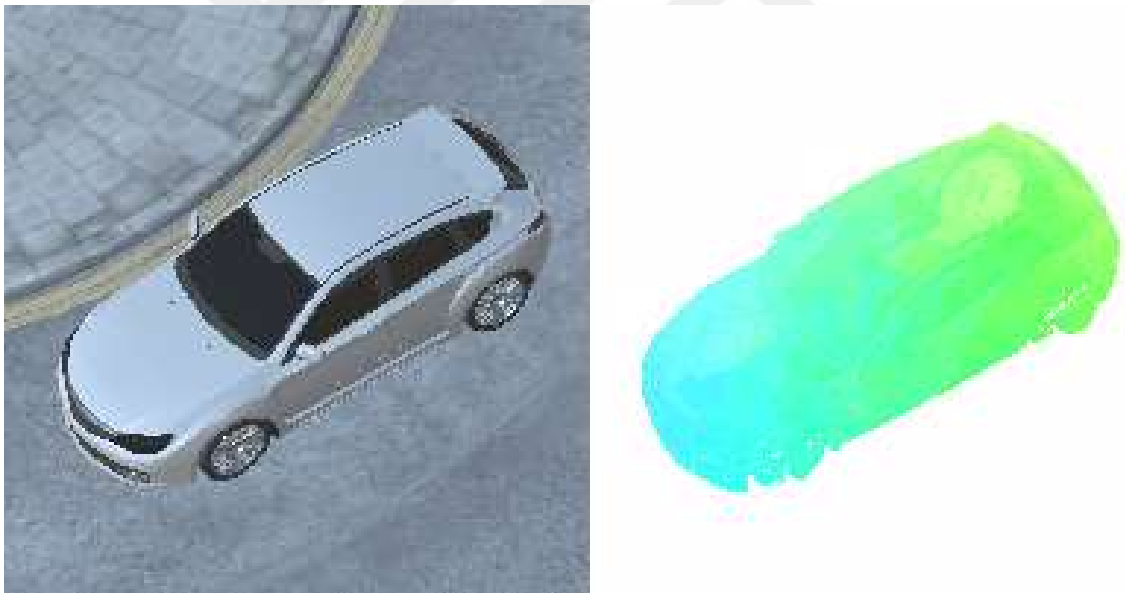


Figure 4.7. Visualized flo file.

The dataset consists of 100 scenarios. Among these, 55 are sunny, 17 are rainy, and 14 correspond to each snowy and foggy weather conditions. The quantity of outputs from the dataset with the resolution of 1920x1080 is listed in Table 4.1.

4.2.2.2. Creation of the 250x250 Resolution Dataset. In the training of CNN models used for optical flow, occlusion, segmentation, and classification, issues can arise re-

Table 4.1. Content of the dataset with the resolution of 1920x1080.

Output of Dataset	Quantity	Format
Normal and masked images	16285	PNG
Optical Flow	16185	PNG, FLO
Occlusion	16185	PNG
Segmentation	16285	PNG, JSON

garding the processing of large-sized video images such as 1920x1080. Due to proposed approach to object detection and tracking, it has been decided to create a dataset of lower resolution. For this purpose, the dataset with a resolution of 250x250 has been produced along with reference label information. This process requires masked video images and a data file containing information about the triangles that make up the vehicles. The flow diagram for the operations to be performed in this section is shown in Figure 4.8.

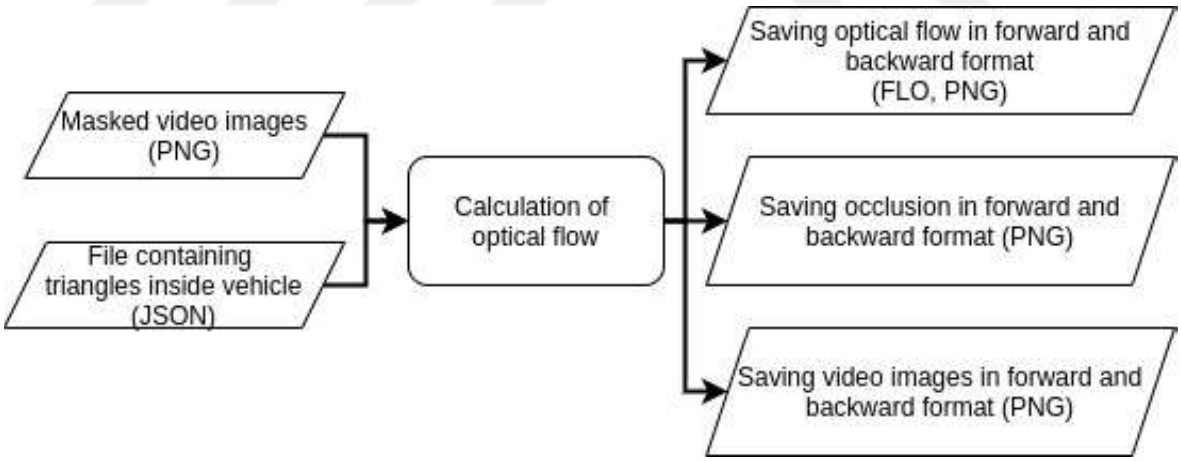


Figure 4.8. Flow diagram for the creation of optical flow, occlusion, and segmentation information with a resolution of 250x250.

For this process, the center point of each vehicle in the video frame at instant T_n is used as a reference to crop a 250x250 section. In the cropping process, the video images are saved as T_n. Then, the vehicle in the T_{n+1} image is cropped at the same pixel coordinates as the cropped vehicle in the T_n image, and the resulting video image

is saved as T_{n+1}. Thus, the vehicles in the two successive images are cropped in relation to the vehicle in the T_n image. This way, the amount of progress of a vehicle in the next image is extracted. For example, video image number 367 is divided into 367-02 and 368-01, while video image 368 is divided into 368-02 and 369-01. This allows the movements of the vehicles in the next video frame to be visible. Otherwise, in the cropping process performed at a size of 250x250, any vehicle present in the T_n and T_{n+1} images would always be located at the center, making it impossible to obtain progress information.

In Figure 4.9, the optical flow, occlusion, and segmentation information obtained from video images 367 and 368 are shown. In the optical flow example, the optical flow values of the vehicle left behind by the object have been disregarded. In the occlusion example, if a pixel in the T_n image is covered by an object in the T_{n+1} image or goes beyond the boundaries of the FOV of the camera, it is considered to be overlapping. This process has been created to calculate the movement of the vehicle at T_{n+1}. This procedure is referred to as 'forward' because it detects the next movement of the vehicles. To train CNN models to detect both optical flow and occlusion, a reference dataset for backward optical flow and occlusion is also needed. By reversing the forward process, the movement of the vehicle at instant T_n is calculated based on its position at T_{n+1} and saved in the local files, starting with under the names "forward"and "backward". The forward and backward optical flow and occlusion datasets are produced only in 250x250 format, while only the forward optical flow and occlusion references have been extracted from the dataset with a resolution of 1920x1080.

As a result, the dataset with a resolution of 250x250 has been produced with a larger number of images, focusing on the vehicles in the dataset with a resolution of 1920x1080 through the cropping process. This dataset, with a resolution of 250x250 is used to train CNN models that require the ground truth for forward and backward optical flow. The distribution of the dataset, generated, is listed in Table 4.2. The distribution of the dataset according to the weather conditions and the type of vehicle is shown in Figure 4.10. According to the graph, 8% of the vehicles are buses, 68% are

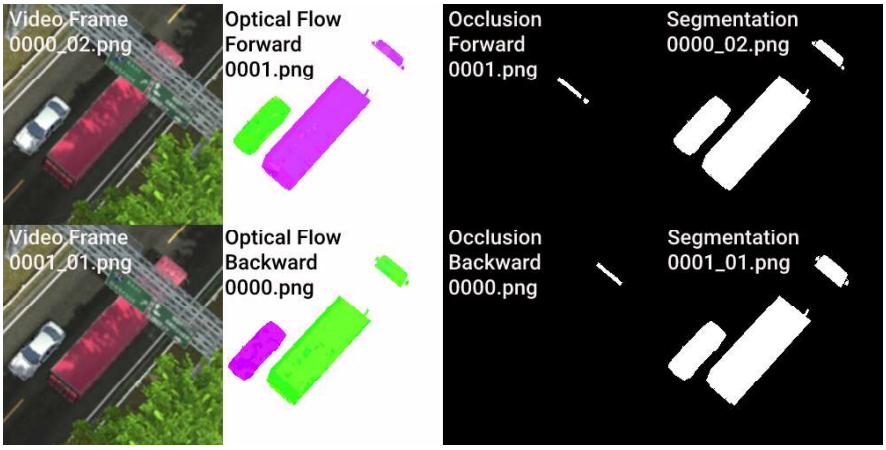


Figure 4.9. Results of optical flow, occlusion, and segmentation information for video images 398 and 399.

cars, 7% are trucks, 7% are lorries, and the remaining 8% are minibuses.

Table 4.2. Quantity of outputs from the 250x250 resolution dataset.

Output of Dataset	Quantity	Format
Normal images	99520	PNG
Masked images	99520	PNG
Forward Optical Flow	49760	PNG, FLO
Backward Optical Flow	49760	PNG, FLO
Forward Occlusion	49760	PNG
Backward Occlusion	49760	PNG
Segmentation	99520	PNG, JSON

4.2.2.3. Creation of Segmentation Information. To obtain the vehicle boundary data, mask information from the dataset is required. For this purpose, the system reads all pixels individually and classifies them according to RGB color codes. After reading the pixel values from the masked video image, all vehicles are classified based on the mask information. The conversion of classified vehicles to segmentation data is encoded using the Run Length Encoding (RLE) [93] method necessary for the COCO format, and then the bounding box information for the top, left, right, and bottom pixels of each mask is determined. After extracting the segmentation and bounding box information

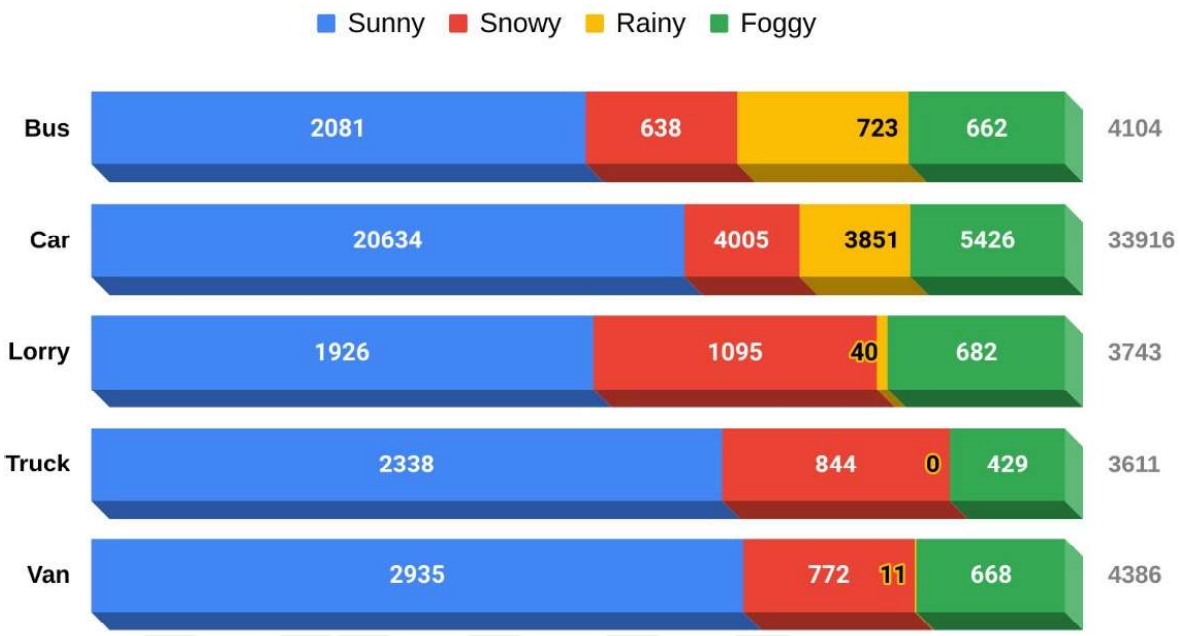


Figure 4.10. Distribution graph of the 250x250 resolution dataset labels prepared with a total of 100 scenarios.

for a video image, this process is applied to all video images. Then, a tracking code is assigned for each vehicle. The tracking code obtained, along with segmentation and boundary information, is converted to a json file in COCO format.

4.3. Creation of the Segmentation Ground Truth for UAV 123 Dataset

UAV123 10 fps dataset [2] does not include the segmentation ground truth to train/fine-tune the segmentation model, trained using S-UAV dataset [10], [11] by "Segmentation Model". It just has the ground truth data for the location and the size of the bounding box. Therefore, Computer Vision Annotation Toolbox (CVAT) in [94] is tailored to create the segmentation ground truth for 21 videos in the UAV123 dataset [2], including only cars as vehicles. As shown in Figure 4.11, each car on the scene is segmented using an artificial intelligence (AI) interactor, named "Segment Anything 2.0", and two important outputs are generated as output 1. Segmentation mask, 2.Segmentation data in COCO format as json file. Segmentation data in COCO format comprises the following information, named as;

- id: the number of annotated cars.
- image_id: the number of the images, annotated.
- category_id: category of the annotated object, car or background.
- segmentation points: 12 pairs of points along the x and y axes that constitute the segmented area.
- area: area of the bounding box.
- bounding box points: 4 pairs of points of the bounding box of the annotated car.

In segmentation mask, cars are annotated or marked with white color (255), and the background with black (0). 1263 images and segmentation masks are annotated.

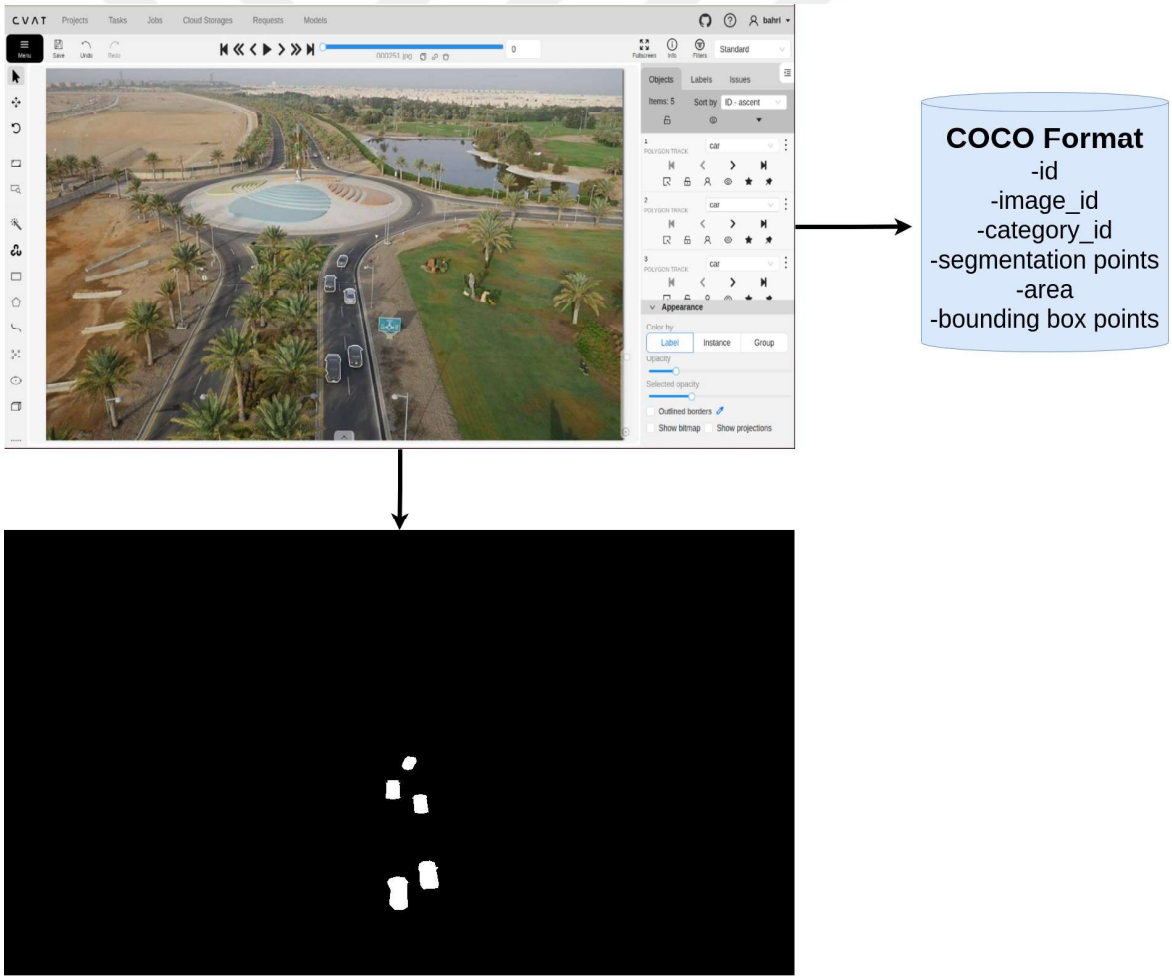


Figure 4.11. Segmentation annotation with computer vision annotation toolbox.

4.4. Conclusion

CNN models require large-scale datasets for training that include a rich diversity of objects and encompass various weather conditions and landscapes, in order to perform robust object detection and tracking in UAV video images. Currently, no dataset exists in the literature that includes the ground truth for the optical flow, occlusion, and segmentation information obtained from UAVs. In the method, developed, a synthetic dataset containing optical flow, occlusion, and segmentation reference information of vehicles from the perspective of UAV has been created for the first time using the Unity game engine, thereby filling the existing gap in the literature. A total of 16,285 video frames of size of 1920x1080 are generated through 100 scenarios featuring various weather conditions. A cropped version with a size of 250x250 has also been presented as a separate dataset. Examples of S-UAV Dataset are presented in [95], and the details are also given in [10], [11]. S-UAV dataset is very vital to train the first version of the proposed method, PFCNNOcc, elaborated in Chapter 5, and plays a very important role in the success of PFCNNOcc for the target tracking.

The final version of the PFCNNOcc, explained in Section 5.3.1.2, is designed based on the first version of the PFCNNOcc to fill the gap that is comprised of visual problems in the UAV123 dataset [2] that are not observed in the synthetic S-UAV dataset [10], [11]. These visual problems that are not present in the synthetic data set are;

- Sudden and large-scale variations,
- Very small vehicles with low resolution as well as very large vehicles whose bounding size is even bigger than the size of ROI (250x250), used to sample the region at the beginning of the iteration,
- Small motion vectors between the sequential images because of the relative motion of the vehicles, depending on the camera motion,
- Merging or the occlusion of the target by the other vehicles during tracking.

To overcome of these limitations in S-UAV dataset, as a future work, the diversity of vehicle models will be increased, and new urban and rural environments and scenarios will be prepared to adapt it to the real-time UAV scenarios in terms of the resolution in size and the visibility of the target as well as the magnitude of the optical vectors of the vehicles. In the mean time, the segmentation of the vehicles, especially cars, in UAV123 dataset [2], is achieved to create the ground truth using CVAT. Segmentation dataset is used to fine-tune the segmentation model in the final version of the PFCNNOcc, and boosts its tracking performance.

5. PFCNNOCC : A ROBUST VEHICLE TRACKING METHOD UNDER OCCLUSION USING OPTICAL FLOW AND SEGMENTATION IN LOW-ALTITUDE UAV VIDEOS

5.1. Introduction

In this chapter, an end-to-end design that can track a vehicle in both a synthetic and a real-life UAV dataset under severe visual problems, especially occlusion is explained in detail. To simultaneously achieve such a hard task requiring robustness against all visual problems, especially partial and/or full occlusion, the tracking method should be designed considering all these anomalies or difficulties in the environment where tracking takes place.

Methods on optical flow and occlusion detection covered in Section 2.2, have shown that the most advanced methods are those capable of performing occlusion detection along with optical flow with high precision and accuracy. These methods are mostly designed to leverage the symmetry property of optical flow based on supervised learning. It has been observed that algorithms developed to improve the total error in the CNN model, which predicts optical flow and occlusion at each pyramid level and finds the errors in relation to the exact reference information, achieve high success. The most important steps in such CNN models can be listed as;

- Extraction of features with different resolutions and the number of channels at each pyramid level.
- Correlation of features, belonging to two consecutive images, by warping one of them with predicted flow, depending on the direction that the flow and occlusion are predicted at each level.

- Downscaling or upscaling of predicted forward or backward flow, depending on the pyramid level.
- Cumulating the errors between predicted forward/backward flows and occlusions with their ground truths at each level.
- Backpropagation of the total error to update model parameters, shared between the layers of the model.

The bidirectional (bilateral) calculation of optical flow in the forward and backward directions further increases the success of these methods. In the PWC-Net [65,66] method, the aforementioned approach is applied to the successful prediction of optical flow, utilizing the same CNN model with different model parameters, learned at each pyramid level. However, this situation results in a significant increase in the number of parameters, making it challenging to use the method in real-time applications. As elaborated in Chapter 2, the most prominent and successful method in the literature is the IRR-PWC [77] method.

As explained in Section 1.3, the PFCNNOcc method, proposed in this thesis, is inspired by IRR-PWC [77], and tailors its CNN models to detect occlusion using optical flow vectors simultaneously. PFCNNOcc also extracts segmentation and classification information of vehicles in synthetic and real-time UAV video datasets by using additional CNN models as shown in the general flow diagram of the PFCNNOcc in Figure 5.1. As seen from Figure 5.1, the way to approach tracking problems changes drastically depending on the percentage of the overall degree of occlusion, called degreeOfOccAll. DegreeOfOccAll is calculated by "Occlusion Detector and Calculator Module Final PFCNNOcc". Three different states of occlusion are determined depending on the value of degreeofOccAll.

- No occlusion: degreeOfOccAll $< 30\%$
- Partial occlusion: $30\% \leq degreesOfOcclusionAll < 100\%$
- Full occlusion : $degreesOfOcclusionAll = 100\%$

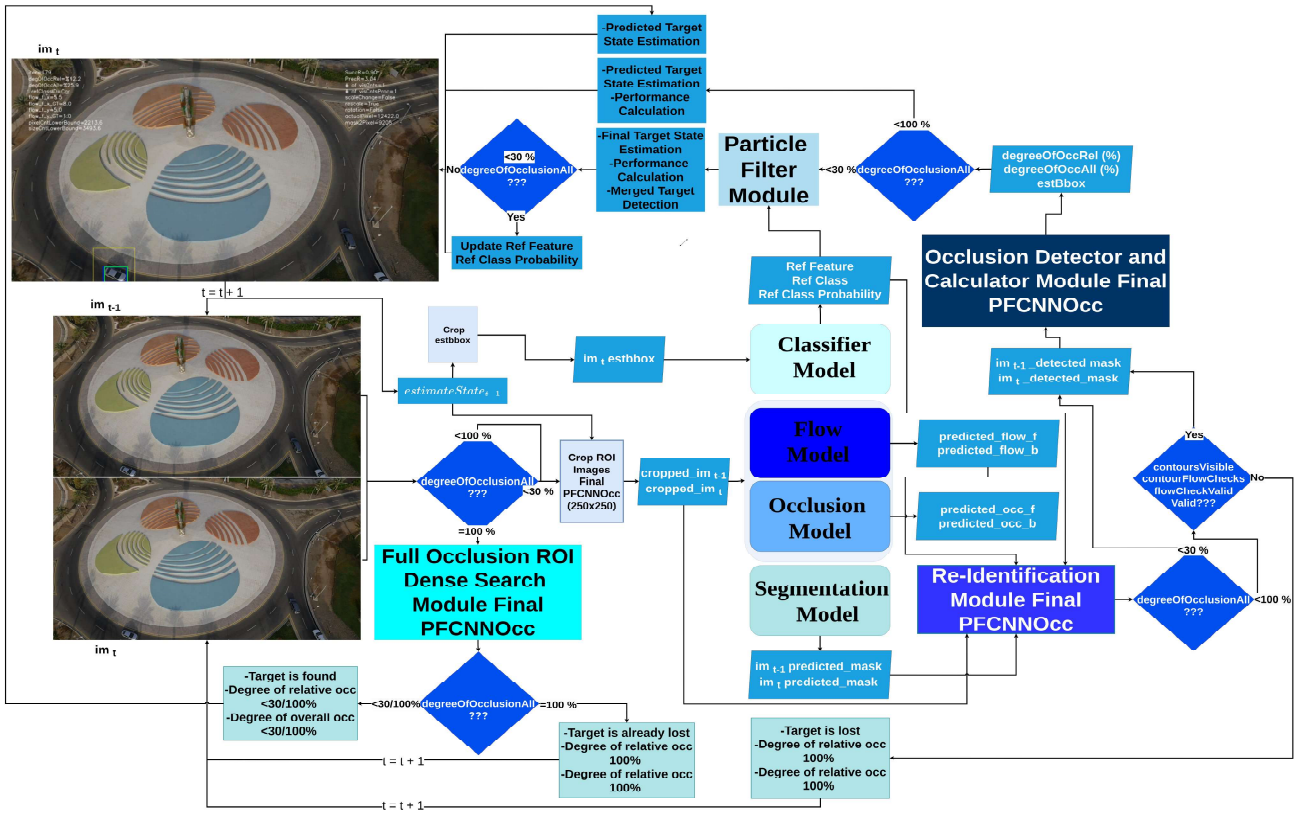


Figure 5.1. General flow diagram of the PFCNNOcc.

It is observed that as the degree of occlusion that the target faces increases, the visual model of the target is changed, and the variance in the visual model of the target gets very high. Therefore, during tracking, the update of the visual model is stopped, and the particle filter is not used to estimate the target kinematics anymore because of the high variance of the visual model that the occlusion triggers. All in all, 30% is assumed to be the optimal threshold to decide the transition between No Occlusion and Partial Occlusion states in the proposed method.

The predictions from these CNN models are tailored by the novel functions, named as;

- Re-Identification Module Final PFCNNOcc,
- Occlusion Detector and Calculator Module Final PFCNNOcc,
- Full Occlusion Dense Search Module Final PFCNNOcc,
- Merged-Target Condition Detection.

These functions are the contributions of the PFCNNOcc to the literature. “Re-Identification Module Final PFCNNOcc”utilizes the forward and backward optical flow predictions, and the segmentation masks of the consecutive images with a novel matching algorithm to re-detect the target at the beginning of each image pair and to determine its location and size at pixel-level with high precision. In the meantime, the average motion vector for the target is also calculated.

“Occlusion Detector and Calculator Final PFCNNOcc”is very vital to decide whether a pixel is non-occluded, will-be-occluded, or already-occluded, and to calculate the degree of occlusion via considering the possible scale and aspect ratio variations.

“Full Occlusion Dense Search Module Final PFCNNOcc”is a combination of the “Re-Identification Module Final PFCNNOcc”and “Occlusion Detector and Calculator Final PFCNNOcc”with a dense-search algorithm to re-detect the target after a short or long-term occlusion.

"Merged Target Detection"function at the end of each iteration detects whether a target is occluded by another vehicle during tracking. After the occlusion or merged-target condition ends, the discrimination capability of the function, separating the target from other vehicles on the run, helps the PFCNNOcc to keep on tracking precisely.

To the best knowledge of the author, the PFCNNOcc is the first method in the literature that predicts and utilizes the optical flow and segmentation to estimate the state variables of the target as well as the occlusion state and the degree of it during target tracking in UAV videos.

The details about the design and the training of these CNN models are elaborated in Section 5.2. The novel tracking method, which is able to track a vehicle in a low-altitude UAV video under partial and full occlusion as well as other visual problems, is detailed in Section 5.3.

5.2. Design and Training of CNN Models in the PFCNNOcc

5.2.1. Architecture

The knowledge and experience, gained from the PWC [65,66] and IRR-PWC [77] methods, have formed the foundation for the occlusion detection mechanism, presented in this thesis. In order to develop a method, capable of simultaneously performing optical flow and occlusion detection, segmentation, and classification, encoder and context network models are included. With the inspiration from the CNN models for optical flow and occlusion prediction in IIR-PWC [77] as well as U-Net [60] for segmentation and VGG16 [87] for classification, the CNN architecture, created by the combination of these models, is the core part of the proposed method in this thesis, namely PFCNNOcc. The CNN architecture estimates the occlusion as well as optical flow simultaneously, and also finds the segmentation masks and classification scores with features. The training method of the CNN architecture is illustrated in the general

flow diagram as shown in Figure 5.2. The details of CNN models and loss functions in flow diagram are explained in Appendix B. It is noteworthy that each model receives two consecutive images as input because optical flow and occlusion arise from the visual differences between two. Although it is not mandatory, the segmentation and classification models are also provided with two consecutive images as input. The only difference in the classification model is that this model accepts the bounding box around the vehicle as input instead of the entire image.

As shown in Figure 5.2, respectively, the "Optical Flow Estimation Model" (Flow Model) and the "Occlusion Estimation Model" (Occlusion Model) are trained simultaneously in relation to each other, while the "Segmentation Estimation Model" (Segmentation Model) and the "Classification Model" (Classifier Model) are trained independently. The structure of CNN models for optical flow and occlusion are the same except the tensor channels, used for optical flow and occlusion. The segmentation branch is used as U-Net model [60], and the classifier branch is utilised as VGG16 model [87]. The training process of these models is elaborated in Appendix B.

5.3. Tracking with PFCNNOcc

First, to extract optical flow, occlusion, segmentation, and classification maps or scores of two consecutive images from low-altitude UAV videos, CNN models are designed and trained. Then, the novel PFCNNOcc method is designed, implemented and tested using the synthetic dataset, S-UAV [10], and the real-time dataset, UAV123 [2]. The architecture of the novel PFCNNOcc is to be explained in Section 5.3.1. Parameters, utilized in experiments, are given in Sections 5.4.2 and 5.4.3. In Sections 5.5 and 5.6, the discussion and conclusion about PFCNNOcc are presented. The simulation results of the novel PFCNNOcc method are elaborated in Section 6.3.

Overall, attribute-based and scenario-based plots are evaluated, and it is validated that PFCNNOcc outperforms the SOTA methods against all visual problems, especially partial and full occlusion.

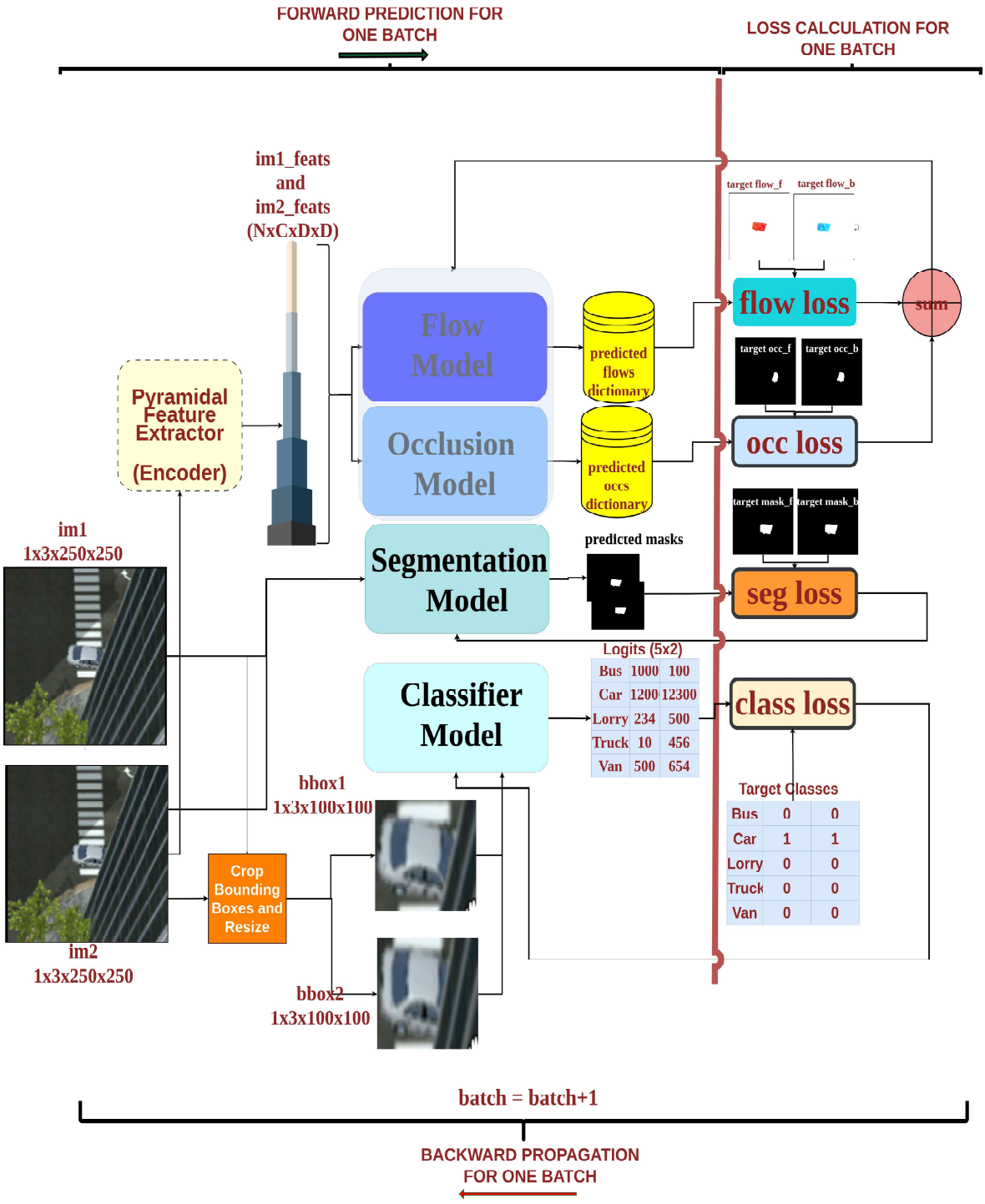


Figure 5.2. General training flow diagram of CNN models for detection of occlusion with optical flow, segmentation, and classification.

5.3.1. Architecture

There are two versions of the PFCNNOcc. The design of the first version of the PFCNNOcc is created to track vehicles on UAV videos from the S-UAV dataset [10]. After the first version of PFCNNOcc is tested against a subset of the S-UAV dataset, including all visual problems, as well as partial and full occlusion, and is shown to be robust against these problems, it is tested on the real-time dataset, UAV123 [2]. Real-time tracking brought additional problems that are not faced in S-UAV tests; thus, to handle these problems, a second and a final version of the novel PFCNNOcc is designed by implementing reformative and evolutionary functions. This final architecture overperforms the SOTA trackers. In this section, the first version of PFCNNOcc is explained utilizing the overall and detailed flow diagrams with the details of sub-modules. On the other hand, all the details of the final version of PFCNNOcc are given in detail using flow diagrams of overall and detailed architecture as well as those of sub-modules. These sub-modules are designed to achieve important tasks during tracking.

5.3.1.1. The First Version of PFCNNOcc Tested in S-UAVDataset. The first version of the PFCNNOcc method tailors CNN models, explained in Section 5.2, to detect optical flow, occlusion, segmentation, and classification. The overall architecture is shown in Figure 5.4.

At the beginning of each iteration, two 1080x1920 (heightxwidth) images, im_{t-1} and im_t, are read from the folder where the S-UAV dataset [10] is stored. Tracking process starts for a specific target depending on the imageID, trackID and categoryID of the target in the ground truth JSON file. While imageID represents the number of the image in that specific scenario or video from the dataset, trackID gives information about the order number of the vehicle, seen on the scene. categoryID is given depending on the type of vehicle, such as bus, car, lorry, truck, or van, as explained in Section 4.2.

Once the degree of occlusion is determined, the way to approach tracking problems changes drastically depending on the percentage of degreeOfOccAll as displayed in Figure 5.4. Three different states of degreeOfOccAll are determined as,

- No occlusion: degreeOfOccAll $< 30\%$
- Partial occlusion: $30\% \leq degreeOfOccAll < 100\%$
- Full occlusion : $degreeOfOccAll = 100\%$

A threshold should be determined to be able to decide the state of the partial occlusion between No Occlusion and Full Occlusion. To figure out this threshold, a lot of experiments with PFCNNOcc have been performed, and instances, the visual model of the target remains almost the same under occlusion, are observed. Even if 25-30 % of the target is occluded, it is verified that the feature of the visual model under the occlusion has changed slightly, and the similarity score between its feature and the reference feature becomes very high. It is observed that as the degree of occlusion increases, the visual model of the target is changed, and the variance of the visual model of the target gets very high. Therefore, during tracking, the update of the visual model is stopped, and the particle filter is not used to estimate the target kinematics anymore because of the high variance in the visual model that the occlusion triggers. All in all, 30% is assumed to be the optimal threshold to decide the transition between No Occlusion and Partial Occlusion states in the proposed method.

The diagram, sketching transitions between states that degreeOfOccAll can have, is given in Figure 5.3.

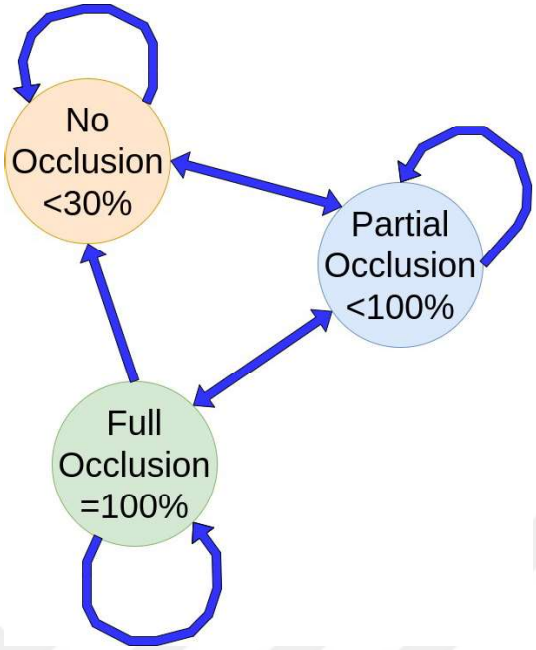


Figure 5.3. Transition of occlusion states.

As understood from Figure 5.3, there are two-way relations between No Occlusion-Partial Occlusion as well as Partial Occlusion-Full Occlusion states. This is a design criteria and subsequently will be explained. There is a one-way transition from the Full Occlusion state to the No Occlusion state because of the possible scale variations during the Full Occlusion that affect the number of pixels inside the bounding box of a segmented car. If the number of pixels is equal to or greater than the reference number of pixels, which is the running average of the pixel counts of the segmented target during the previous iterations with the No Occlusion, after the Full Occlusion vanishes, and the whole vehicle seems on the scene, then the transition case from the Full Occlusion to the No Occlusion is verified. In summary, there are two different states that can be transitioned from the Full Occlusion state, and the possibility of this two-way transition comes from the scale variation that the video scene faces during the Full Occlusion state. The only state that can be transitioned from the No Occlusion state is the Partial Occlusion.

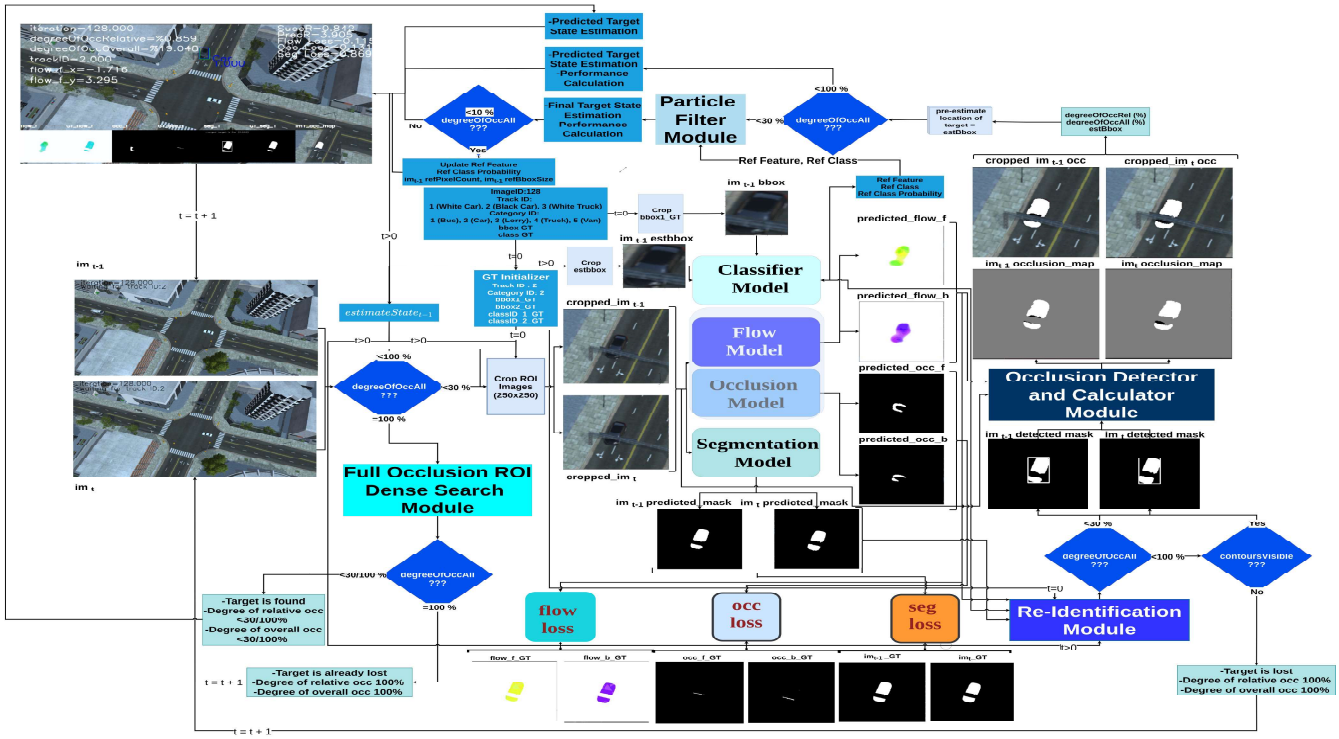


Figure 5.4. Overall flow diagram of the first version of PFCNNOcc.

As illustrated in Figure 5.4, a different path for each occlusion state is designed to be able to track a vehicle under occlusion properly. Hence, the explanation for each different path is given separately in the following paragraphs.

The functionality of the path that will be followed during No Occlusion ($< 30\%$) state, shown in Figure 5.4, is explained in the following paragraphs. The overall degree of occlusion, named degreeOfOccAll is set as 0 before the tracking process starts. targetLostIdx is set to 0 to keep the counter of the full occlusion state during tracking. For the first image pair at $t = 0$, the location, size, and class information of the bounding box of the target on im_{t-1} is read from the ground truth file, and the tracking process starts. Then, ROI boxes, $cropped_im_{t-1}$ and $cropped_im_t$ (3x250x250 (chXwXh)), are cropped from both images, im_{t-1} and im_t, and fed into "Flow Model and Occlusion Model"as well as "Segmentation Model"to predict the forward and backward optical flow maps, $predicted_flow_f$ and $predicted_flow_b$, forward and backward occlusion maps, $predicted_occ_f$ and $predicted_occ_b$, segmented masks, $im_{t-1}_predicted_mask$ and $im_t_predicted_mask$, for both images as shown in Figure 5.4. If $t > 0$, then ROI boxes are cropped using the location and size information of the bounding box from $estimateStates_{t-1}$, representing the estimated state values of the tracked target from the previous state at $t - 1$. In the meantime, at the beginning of the tracking process, or in other words at $t = 0$, the bounding box, $im_{t-1}bbox$, around the target or vehicle is cropped using the ground truth data and feed to "Classifier Model"to calculate the reference feature (Ref Feature), the reference class ID (Ref Class) and the reference class probability (Ref Class Probability). The reference feature and reference class ID are then used in the "Particle Filter Module "to find the likelihood and class scores of particles, and to update their weights, as explained in Section 3.1. As illustrated in Figure 5.4, optical flow loss (flow loss), occlusion loss (occ loss) and segmentation loss (seg loss) are calculated between predicted values and ground truth using the same loss functions, explicated in Section 5.2.1.

After predictions for optical flow, occlusion, and segmentation are computed, one of the most vital processes, called the "Re-Identification Module","starts. While

the flow diagram of the "Re-Identification Module" is illustrated in Figure 5.5, the pseudocode, elaborating the logic of the algorithm, is given in Figure 5.6.

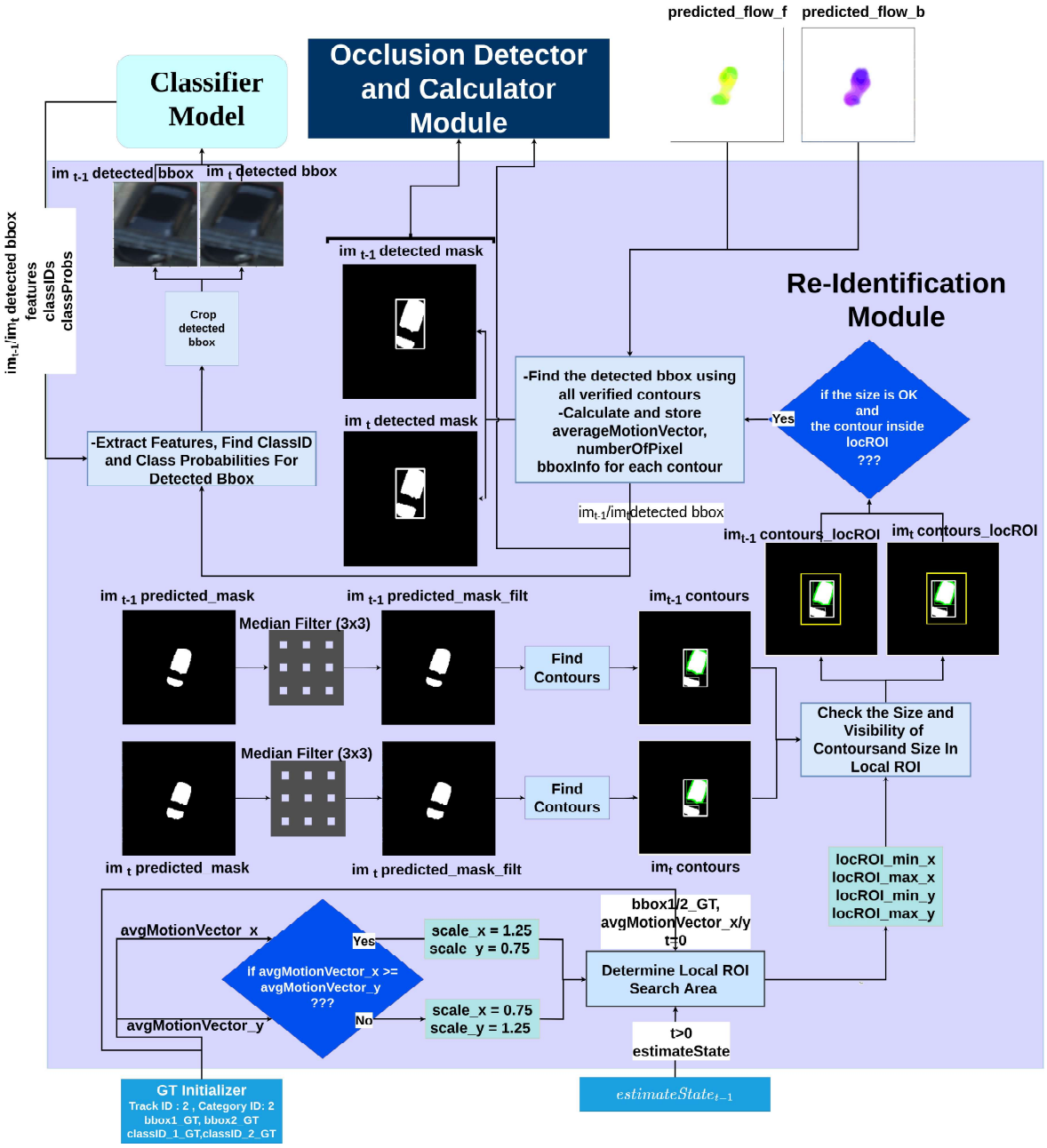


Figure 5.5. Re-Identification module.

As can be understood from the Figures 5.5 and 5.6, "Re-Identification Module" is responsible for the re-identification or re-detection of the target at the beginning

Input: $predicted_masks, predicted_flows,$ **Output:** $detected_bboxes$

function RE-IDENTIFICATION(.) ▷ Runs the algorithm
  $im_{t-1/t}\ predicted_mask_filt \leftarrow$ **medianFilter**$(im_{t-1/t}\ predicted_mask)$
  $im_{t-1/t}\ contours \leftarrow$ **findContours**$(im_{t-1/t}\ predicted_mask_filt)$
  $scale_x/y \leftarrow$ **findScaleFactors**$(avgMotionVector_x/y)$
  $locROI_min_x/y, locROI_max_x/y \leftarrow$ **determineLocROI**$(estimateState_{t-1})$
  for $i \leftarrow 1, size(im_{t-1/t}\ contours)$ **do** ▷ Check size and visibility of contours
    $[contour_bboxArray, min_x/y, max_x/y] \leftarrow$ **findLoc**$(im_{t-1/t}\ contours[i]))$
    if $max_x - min_x \leq 2 \mid max_y - min_y \leq 2$ **then** ▷ Check size
      ignore $im_{t-1/t}\ contours[i]$ ▷ Ignore outlier
    else ▷ Check visibility in locROI
      if $contour_bboxArray\ inside\ locROI$ **then** ▷ Visibility satisfied
        $visible_contour_list \leftarrow$ **storeVisibleContours**$(min_x/y, max_x/y)$
        for $j \leftarrow min_x, max_x$ **do** ▷ Form the motionlists with forward flow
          for $k \leftarrow min_y, max_y$ **do**
            $motionList_x/y \leftarrow$ **storeMotionVector**$(predicted_flow_f)$
          end for
        end for ▷ Get histogram of motion lists
        $avgMotionVector_x/y \leftarrow$ **histogram**$(motionList_x/y))$
        if $degreeOfOccAll \leq 10\%$ **then** ▷ Update reference features
          $avgMotionVect, numOfPixel, sizeBbox \leftarrow$ **updateRefFeat()**
        end if
      else ▷ Visibility not satisfied
        ignore $im_{t-1/t}\ contours[i]$ ▷ Ignore contour
      end if
    end if
  end for
  if $counterContourVisible \geq 1$ **then** ▷ Find the coords. of detected bbox
    $detected_bbox \leftarrow$ **findLoc**$(visible_contour_list)$
  end if
end function

Figure 5.6. The pseudocode of the Re-Identification module.

of each iteration using the inputs such as segmented masks, $im_{t-1}_predicted_mask$ and $im_t_predicted_mask$, and predicted forward and backward optical flow maps, $predicted_flow_f$ and $predicted_flow_b$. The precision of the predicted masks and optical flow maps has a key role in the performance of the "Re-Identification Module".

After the re-identification of the target is achieved, "Occlusion Detector and Calculator Module", shown in Figure 5.7, is utilized to detect or fine-tune forward and backward occlusion maps as well as to calculate the degree of the relative occlusion, degreeOfOccRel, and the overall occlusion, degreeOfOccAll. While degreeOfOccRel is measured only between the pixels, seen in two consecutive images, degreeOfOccAll is calculated between the pixels seen in an image at a specific time, and the pixels seen in an image without occlusion. As understood from Figure 5.7, the novel "Occlusion Detector and Calculator Module"is designed with the combination of three distinct functions; 1. "Forward Processor", 2. "Backward Processor", and 3. "Occlusion Calculator".

The functionality of "Forward Processor"is to detect the non-occluded pixels and seen-pixels in $im_{t-1}detected_mask$ as well as the pixels that will be occluded in $im_tdetected_mask$ (will-be-occluded). In other words, "Forward Processor"detects the forward occlusion via operating on a fundamental idea, called "Forward-Backward Consistency of Optical Flow". The details of the function are explained with the flow diagram in Figure 5.8, and are elaborated in Figure 5.9. There are two requirements to be checked to guarantee that the processed pixel is not occluded; values of pixels at $(x1, y1)$ on $im_t_detected_mask$ and $(x2, y2)$ on $im_{t-1}_detected_mask$ are of value 255. These checks guarantee the consistency between $predicted_flow_f$ and $predicted_flow_b$ as well as the decision whether the pixel seen on $im_{t-1}_detected_mask$, is occluded or not. If the consistency between $predicted_flow_f$ and $predicted_flow_b$ holds, the location values of (i, j) and $(x2, y2)$ are the same. While requirements are satisfied, the counter for the number of pixels that will not be occluded, $im_{t-1}_detected_mask_non_occ_cnt$, is increased, the pixel location at $im_{t-1}occlusion_map$ mask and $cropped_im_{t-1}occ$ image is set with 255 (white) to mark and keep track of the non-occluded and occluded pixels.

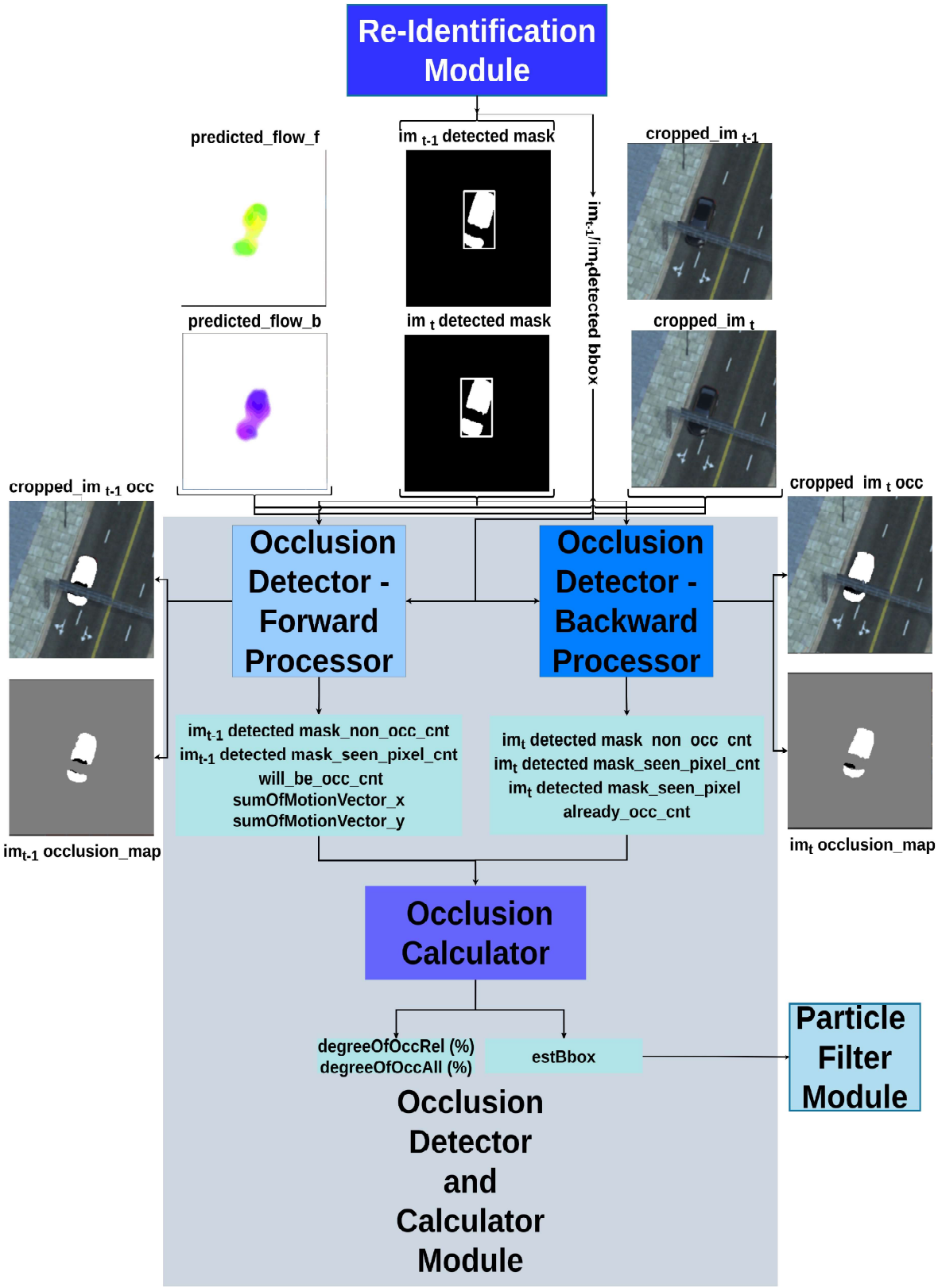


Figure 5.7. Occlusion detector and calculator module.

In the meantime, the total motion of the target along the x and y axes is calculated via summation of the forward optical flow vector values of $predicted_flow_f$ for non-occluded pixels as shown in Figure 5.9. If these requirements are not met, it means that the consistency between $predicted_flow_f$ and $predicted_flow_b$ does not hold, and the pixel will be occluded in $im_t_detected_mask$ coordinates. In $im_{t-1}occlusion_map$ mask, grey pixels show the background, white pixels represent the ones that are seen on $cropped_im_{t-1}$ as well as non-occluded on $cropped_im_t$. Pixels in black color are the ones that are seen on $cropped_im_{t-1}$ and will be occluded in $cropped_im_t$. $cropped_im_{t-1}occ$ shows all pixels with the same manner as done in $im_{t-1}occlusion_map$, but the background belongs to the $cropped_im_{t-1}$. As a final step, the output of "Forward Processor"is sent to the "Occlusion Calculator"function of "Occlusion Detector and Calculator Module".

The functionality of "Backward Processor"is to detect the non-occluded pixels and seen-pixels in $im_tdetected_mask$ as well as the pixels that are already occluded in $im_{t-1}detected_mask$ (already-occluded). "Backward Processor"detects the backward occlusion depending on "Forward-Backward Consistency of Optical Flow". The output of the "Backward Processor"is sent to the "Occlusion Calculator"function of the "Occlusion Detector and Calculator Module". The details of the function are explained with the flow diagram in Figure 5.10, and are elaborated in Figure 5.11.

The functionality of "Forward Processor" and "Backward Processor" is very similar in terms of the detection of non-occluded pixels via utilizing the consistency of forward and backward flow maps. The main difference emerges because of the way that occluded pixels are determined. While the "Forward Processor" mainly concentrates on pixels that will not be seen or occluded in the next image of the pair, in other words, focusing on the situation of the pixels in the future, the "Backward Processor" keeps track of already-occluded pixels in the previous image of the pair. With the combination of these important functions, non-occluded, will-be-occluded, and already-occluded pixels are detected and used to calculate the relative and overall degree of occlusion during target tracking.

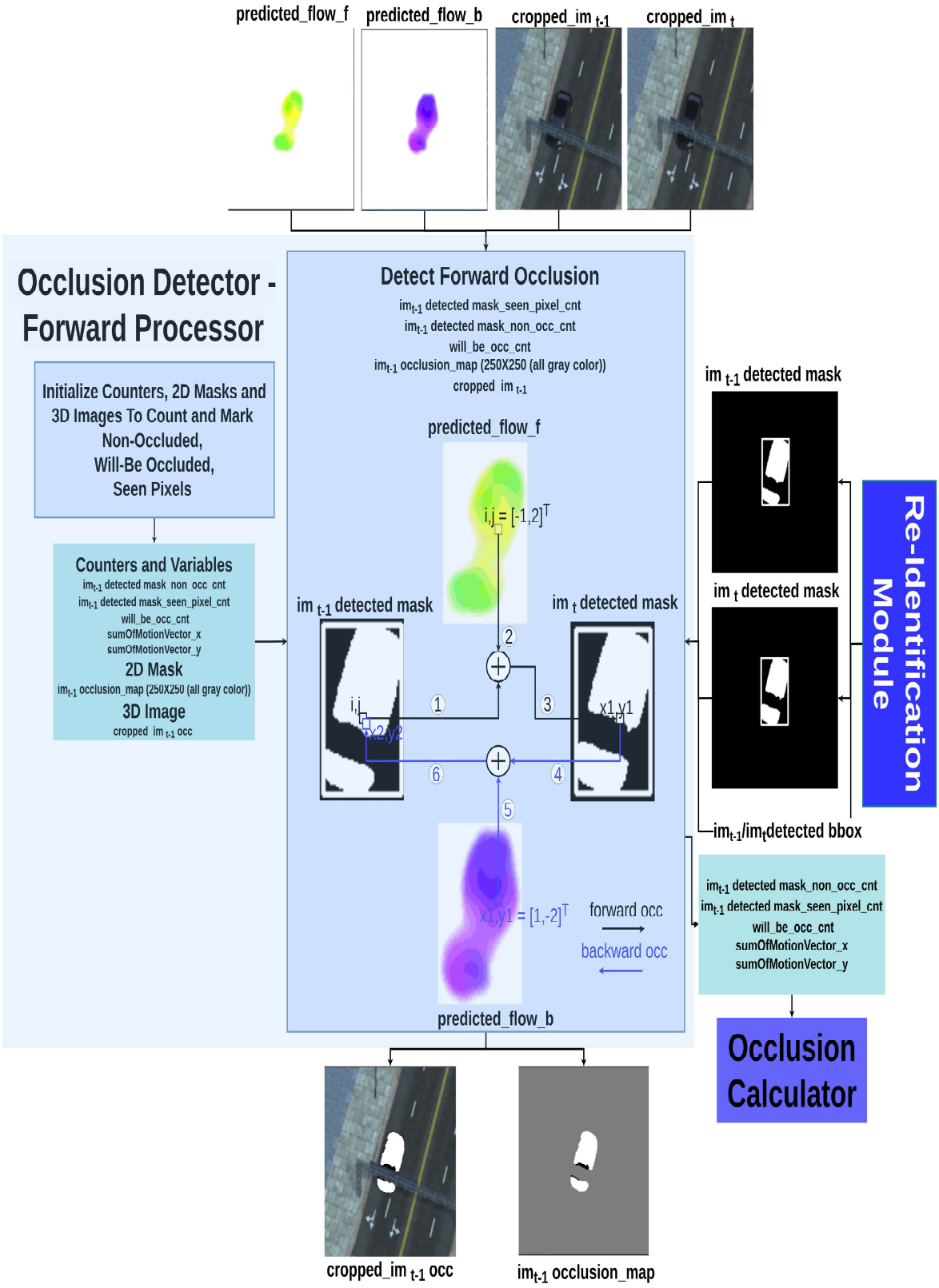


Figure 5.8. Occlusion detector and calculator module-forward processor.

Input: $detected_masks, predicted_flows, cropped_images, detected_bboxes$

Output: $im_{t-1}_detected_mask_non_occ_cnt, im_{t-1}_detected_mask_seen_pixel_cnt, will_be_occ_cnt, sumOfMotionVectors$

function FORWARD PROCESSOR(.) ▷ Runs the algorithm
  for $i \leftarrow im_{t-1}\ detected_bbox[1], im_{t-1}\ detected_bbox[3]$ **do** ▷ Along y axis
    for $j \leftarrow im_{t-1}\ detected_bbox[0], im_{t-1}\ detected_bbox[2]$ **do** ▷ Along x axis
      if $im_{t-1}_detected_mask[i,j] == 255$ **then** ▷ Check if it is target
        $x1 \leftarrow i + predicted_flow_f[i,j,1]$
        $y1 \leftarrow j + predicted_flow_f[i,j,0]$
        $x2 \leftarrow i + predicted_flow_f[i,j,1] + predicted_flow_b[x1,y1,1]$
        $y2 \leftarrow j + predicted_flow_f[i,j,0] + predicted_flow_b[x1,y1,0]$
        if $im_t_detected_mask[x1,y1] == 255$ & $im_{t-1}_detected_mask[x2,y2] == 255$ **then**
          $im_{t-1}\ detected_mask_non_occ_cnt += 1$
          $im_{t-1}\ occlusion_map[i,j] \leftarrow 255$
          $cropped_im_{t-1}\ occ[i,j,:] \leftarrow 255$
          $sumOfMotionVector_x \leftarrow sumOfMotionVector_x + predicted_flow_f[i,j,1]$
          $sumOfMotionVector_y \leftarrow sumOfMotionVector_y + predicted_flow_f[i,j,0]$
        else
          $im_{t-1}\ occlusion_map[i,j] \leftarrow 0$
          $cropped_im_{t-1}\ occ[i,j,:] \leftarrow 0$
          $will_be_occ_cnt += 1$
        end if
      end if
    end for
  end for
end function

Figure 5.9. The pseudocode of the Occlusion Detector-Forward Processor.

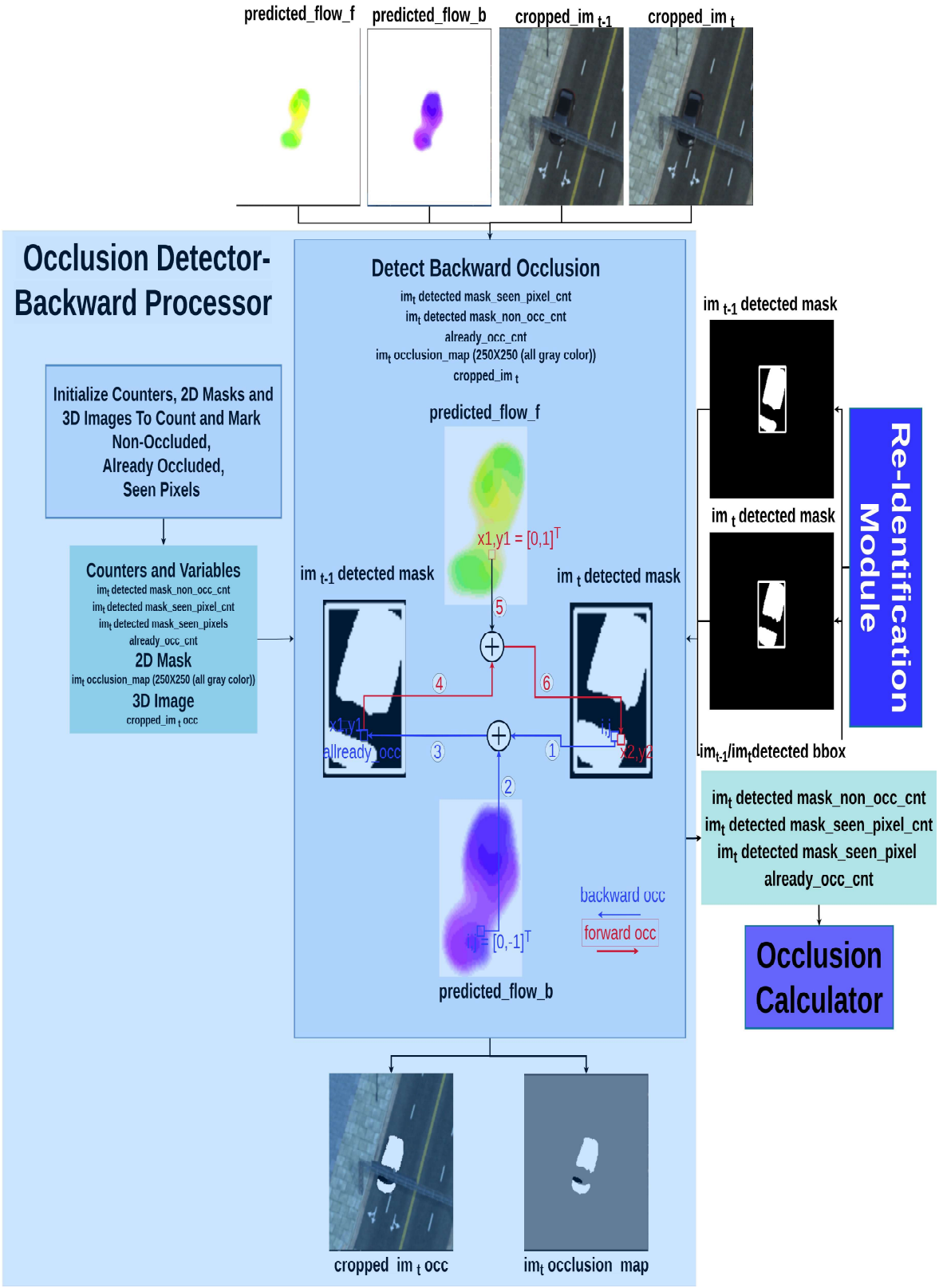


Figure 5.10. Occlusion detector and calculator module-backward processor.

Input: $detected_masks, predicted_flows, cropped_images, detected_bboxes$

Output: $im_t_detected_mask_non_occ_cnt, im_t_detected_mask_seen_pixel_cnt, already_occ_cnt$

1: **function** BACKWARD PROCESSOR(.) ▷ Runs the algorithm
2: **for** $i \leftarrow im_t\ detected_bbox[1], im_t\ detected_bbox[3]$ **do** ▷ Along y axis
3: **for** $j \leftarrow im_t\ detected_bbox[0], im_t\ detected_bbox[2]$ **do** ▷ Along x axis
4: **if** $im_t_detected_mask[i,j] == 255$ **then** ▷ Check if it is target
5: $im_t_detected_mask_seen_pixel_cnt += 1$
6: $im_t_detected_mask_seen_pixels_x.append \leftarrow i$
7: $im_t_detected_mask_seen_pixels_y.append \leftarrow j$
8: $x1 \leftarrow i + predicted_flow_b[i,j,1]$ ▷ Transform the pixel using predicted backward flow into $im_{t-1}_detected_mask$ coordinates.
9: $y1 \leftarrow j + predicted_flow_b[i,j,0]$
10: $x2 \leftarrow i + predicted_flow_b[i,j,1] + predicted_flow_f[x1,y1,1]$ ▷ Transform the pixel back into $im_t_detected_mask$ coordinates using predicted forward flow.
11: $y2 \leftarrow j + predicted_flow_b[i,j,0] + predicted_flow_f[x1,y1,0]$
12: **if** $im_{t-}_detected_mask[x1,y1] == 255 \& im_t_detected_mask[x2,y2] == 255$ **then**
13: $im_t\ detected_mask_non_occ_cnt += 1$
14: $im_t\ occlusion_map[i,j] \leftarrow 255$
15: $cropped_im_t\ occ[i,j,:] \leftarrow 255$
16: **else**
17: $im_t\ occlusion_map[i,j] \leftarrow 0$
18: $cropped_im_t\ occ[i,j,:] \leftarrow 0$
19: $already_occ_cnt += 1$
20: **end if**
21: **end if**
22: **end for**
23: **end for**
24: **end function**

Figure 5.11. The pseudocode of the Occlusion Detector-Backward Processor.

At this point, “Occlusion Calculator”is the last function of “Occlusion Detector And Calculator Module”, and is designed to calculate the “degreeOfOccRel”and “degreeOfOccAll”as well as the preliminary estimate of bounding box location and size in 250X250 local image coordinates, *estBox*, using the inputs from “Forward Processor”and “Backward Processor”as elaborated in Figure 5.12. At the end of the function, *estBox* is sent to the “Particle Filter Module”for final estimation of the state variables of the target.

The final phase of the tracking process is the estimation of the state variables using the “Particle Filter Module”, shown in Figure 5.13 while degreeOfOccAll is smaller than 30%. “Particle Filter Module”, utilized here, is designed via the combination of “Initialization and Particle Sampling Module”and “Object State Estimation Module”, which are parts of the novel PFCNN method, explained in Section 3.1.

While all derivations and equations are modelled for multi-part tracking for PFCNN in Section 3.1, “Particle Filter Module”do estimation for the whole body of the target; hence, equations are derived based on only one part. The process and measurement equations are shown in Equations (3.1-3.4). The state variables are defined as

$$x_t = \begin{bmatrix} estBBox_{xLoc} \\ estBBox_{yLoc} \\ estBBox_{xDim} \\ estBBox_{yDim} \end{bmatrix} v_t = \begin{bmatrix} n_{estBBox_{xLoc}} \\ n_{estBBox_{yLoc}} \\ n_{estBBox_{xDim}} \\ n_{estBBox_{yDim}} \end{bmatrix} r_t = \begin{bmatrix} n_{estBBox_{xLoc}} \\ n_{estBBox_{yLoc}} \\ n_{estBBox_{xDim}} \\ n_{estBBox_{yDim}} \end{bmatrix} \tag{5.1}$$

where v_t and r_t are the vectors of process and measurement noises, respectively, sampled from i.i.d. normally distributed probability distribution functions (pdfs), and z_t is the measurement vector. x_t includes the location and dimension of the bounding boxes along x and y coordinates, and it is initialized with the preliminary estimate of the states in global image coordinates, *estBbox*, as shown in Figure 5.13 and Equation (5.1). The random walk model provides the basis for the formulation of the process equation given in Equation (3.1).

Input: $im_{t-1}_detected_mask_non_occ_cnt, im_{t-1}_detected_mask_seen_pixel_cnt, will_be_occ_cnt, sumOfMotionVectors, im_t_detected_mask_non_occ_cnt, im_t_detected_mask_seen_pixel_cnt, im_t_detected_mask_seen_pixel, already_occ_cnt$

Output: $degreeOfOccRel, degreeOfOccAll, estBbox, avgMotionVectors$

function OCCLUSION CALCULATOR(.) ▷ Runs the algorithm

  $sumOfOccCnts \leftarrow im_t_detected_mask_non_occ_cnt + will_be_occ_cnt + already_occ_cnt$

  if $sumOfOccCnts == 0$ **then** ▷ Check if full occlusion exists!

    $degreeOfOccRel(\%), degreeOfOccAll(\%) \leftarrow 100$ ▷ Full occlusion exist!

  else ▷ (No full occlusion!Calculate degreeOfOccRel)

    $degreeOfOccRel(\%) \leftarrow \frac{will_be_occ_cnt}{sumOfCnts} * 100$

  end if

  if t==0 **then** ▷ Check if it is the initial frame

    $actual_non_occ_pixel_cnt \leftarrow im_{t-1}_detected_mask_seen_pixel_cnt$

  else ▷ It is not the initial frame

    if $im_t_detected_mask_seen_pixel_cnt > actual_non_occ_pixel_cnt$ **then**

      $degreeOfOccAll(\%) \leftarrow 0$ ▷ No occlusion!

      $actual_non_occ_pixel_cnt \leftarrow im_t_detected_mask_seen_pixel_cnt$

    else ▷ Occlusion exists!

      $degreeOfOccAll(\%) \leftarrow \frac{(actual_non_occ_pixel_cnt - im_t_detected_mask_seen_pixel_cnt)*100}{actual_non_occ_pixel_cnt}$

    end if

  end if

  if $im_{t-1}_detected_mask_non_occ_cnt > 0$ **then**

    $avgMotionVector_x/y \leftarrow \frac{sumOfMotionVector_x/y}{im_{t-1}_detected_mask_non_occ_cnt}$

  end if

  $estBbox[min_x/y, max_x/y] \leftarrow$ **findLoc**$(im_t_detected_mask_seen_pixel)$ ▷ Find the coords. of the preliminary estimate of bbox

end function

Figure 5.12. The pseudocode of the Occlusion Calculator.

At the final step, after calculating the estimated values for the state variables at time t, $xEst_t$, if degreeOfOccAll is smaller than 10% ($< 10\%$), reference data for the motion vector, pixel counts, and bounding box sizes for the target are updated as

$$\begin{aligned} hist_{motionVectorArray_x} &= histogram(motionVectorArray_x) \\ averageMotionVector_x &= hist_{motionVectorArray_x}(argmax(hist_{motionVectorArray_x})) \\ hist_{motionVectorArray_y} &= histogram(motionVectorArray_y) \\ averageMotionVector_y &= hist_{motionVectorArray_y}(argmax(hist_{motionVectorArray_y})) \end{aligned} \tag{5.2}$$

$$\begin{aligned} hist_{im_{t-1}_pixelCountArray} &= histogram(im_{t-1}_pixelCountArray) \\ im_{t-1}_refPixelCount &= hist_{im_{t-1}_pixelCountArray}(argmax(hist_{im_{t-1}_pixelCountArray})) \\ hist_{im_t_pixelCountArray} &= histogram(im_t_pixelCountArray) \\ im_t_refPixelCount &= hist_{im_t_pixelCountArray}(argmax(hist_{im_t_pixelCountArray})) \end{aligned} \tag{5.3}$$

$$\begin{aligned} hist_{im_{t-1}_bboxSizeArrayArray} &= histogram(im_{t-1}_bboxSizeArray) \\ im_{t-1}_refBboxSize &= hist_{im_{t-1}_bboxSizeArray}(argmax(hist_{im_{t-1}_bboxSizeArray})) \\ hist_{im_t_bboxSizeArray} &= histogram(im_t_bboxSizeArray) \\ im_t_refBboxSize &= hist_{im_t_bboxSizeArray}(argmax(hist_{im_t_bboxSizeArray})) \end{aligned} \tag{5.4}$$

$$\begin{aligned} vectorSimilarity &= N(feature_t(im_{t-1}_bbox)^T - feature_t(im_t_bbox), \sigma^2_{meas}) \\ im_{t-1}_score &= p(feature_t(im_{t-1}_bbox) \mid refFeat_t) \\ &\quad * P(feature_t(im_{t-1}_bbox) \mid classIdx) \\ im_t_score &= p(feature_t(im_t_bbox) \mid refFeat_t) \\ &\quad * P(feature_t(im_t_bbox) \mid classIdx) \\ confusionScore &= im_{t-1}_score * im_t_score * vectorSimilarity \end{aligned} \tag{5.5}$$

and the summary for the calculations are itemized below:

- $refFeat_t$ is updated using "Classifier Model"as shown in Figure 5.4.
- averageMotionVectors of the target for forward direction along x and y axes are calculated as shown in Equation (5.2).
- references for # of pixels of the target at $im_{t-1}detected_mask$ and $im_tdetected_mask$ are found as in Equation (5.3).
- references for sizes of the target at $im_{t-1}detected_mask$ and $im_tdetected_mask$ are calculated as given in Equation (5.4).
- confusionScore for the detected targets at $im_{t-1}detected_mask$ and $im_tdetected_mask$ is calculated as given in Equation (5.5).

Estimate of state variables, named as $estimateState_{t-1}$, is utilized to crop images at the beginning of the next iteration after the counter for t is increased, as shown in Figure 5.4, and this is the end of a tracking iteration if degreeOfOccAll is smaller than 30 %.

The functionality of the path that will be followed during Partial Occlusion (30% < & <100%) state, shown in Figure 5.4, is explained in the following paragraphs.

In that case, the iteration starts with initializing/updating $predictedState_t$ utilizing the random walk model with $estimateState_t$ or $predictedState_t$ depending on the value of the degreeOfOccAll from the previous iteration as

$$\begin{aligned} motionVector &= [averageMotionVector_x, averageMotionVector_y, 0, 0] \\ predictedState_t &= estimateState_{t-1} + motionVector(degreeOfOccAll_{t-1} < 30\%) \\ predictedState_t &= predictedState_{t-1} + motionVector(degreeOfOccAll_{t-1} \leq 100\%). \end{aligned} \quad (5.6)$$

At this state of occlusion, the degree of occlusion is high, and it is considered that most of the target or all of it may not be seen. Hence, predicted values of the state variables are utilized. After $predictedState_t$ is calculated, the steps, until the output of "Re-Identification Module"in Figures 5.4 and 5.5 is generated while degreeOfOccAll is smaller than 30%, are followed in the same order. After the "Re-Identification

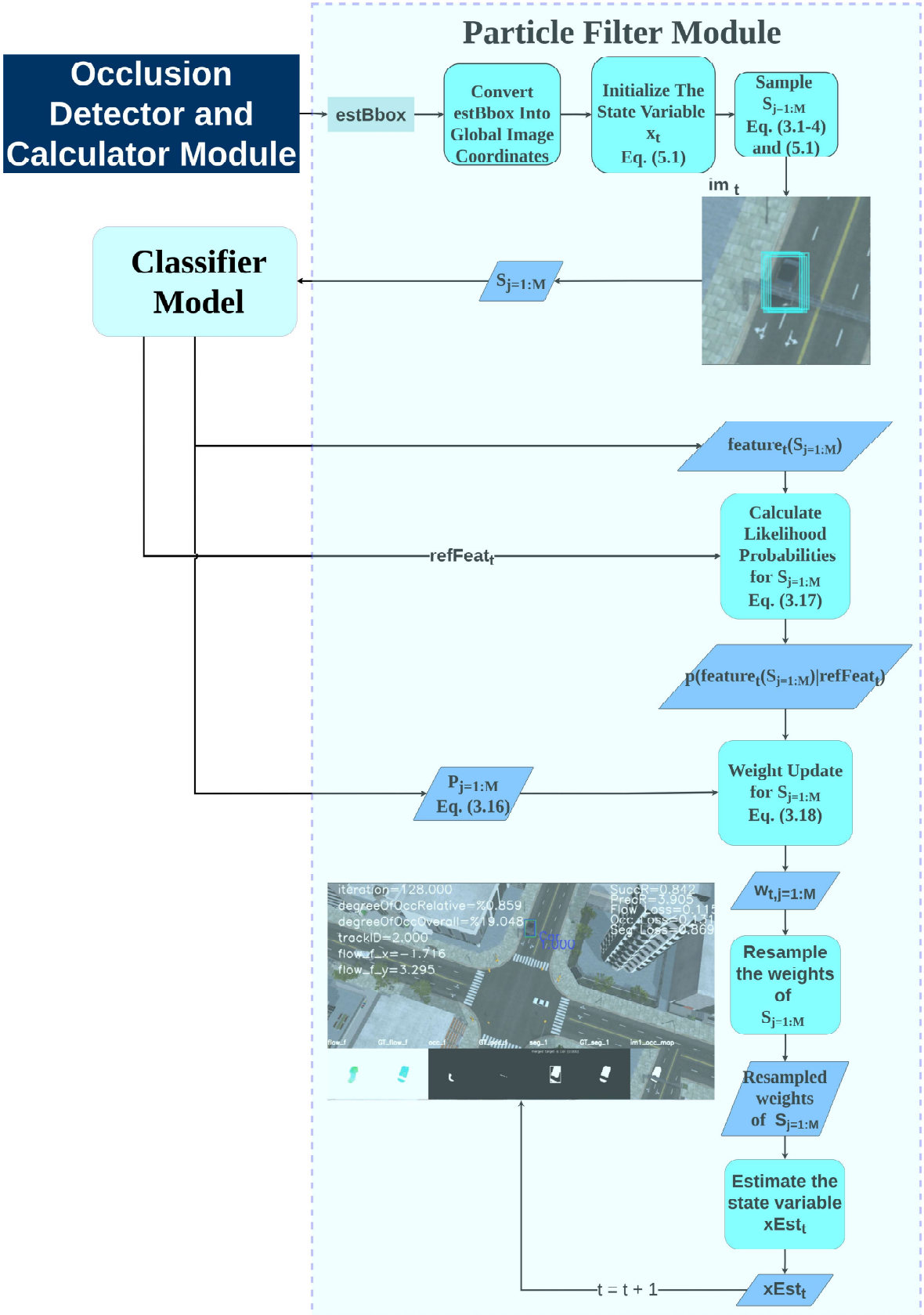


Figure 5.13. Particle filter module.

Module"is processed, the counters for the number of visible contours in the local search areas, $locROIs$, at $im_{t-1}_contours_locROI$ and $im_t_contours_locROI$, calculated by "Re-Identification Module"as shown in Figure 5.5, are utilized to check whether or not a contour is visible inside $locROIs$. If there exist no contours inside $locROIs$ in one or both of the images, then it is considered that the target is lost or fully occluded. If the target is lost, degreeOfOccRel and degreeOfOccAll are set to 100 %, and the iteration stops at that stage, the counter for time t is incremented, and the next iteration starts with a fully occluded state (if degreeOfOccAll = 100 %).

If there exist contours inside $locROIs$ at both images, $im_{t-1}detected_mask$ and $im_tdetected_mask$ are fed to "Occlusion Detector and Calculator Module"to keep on the iteration as shown in Figure 5.4. The path that the iteration follows is the same with the one, achieved during $degreeOfOccAll$ is smaller than 30% until pre-estimate location of the target or $estBox$ is calculated. For this state of occlusion ($30\% < degreeOfOccAll < 100\%$), $predictedState_t$ is updated with $estBox$ while transitioning from 250X250 image coordinates to the original image coordinates without the application of "Particle Filter Module"as shown in Figure 5.4. Then, $estimateState_t$ is set to $predictedState_t$, the counter for time t is incremented, and the next iteration starts depending on the value of degreeOfOccAll, calculated by "Occlusion Detector and Calculator Module"during the previous iteration. This is the end of a tracking iteration if $30\% < degreeOfOccAll < 100\%$. In summary, the iteration starts with "Partial Occlusion", because degreeOfOccAll was between 30% and < 100%, and depending on the tracking process, explained up to now, the state of occlusion can transition to No Occlusion or Full Occlusion as different states. Also, it can stay in its state, Partial Occlusion, as shown in Figure 5.3.

The functionality of the path that will be followed during Full Occlusion (= 100%) state, shown in Figure 5.4, is explained in the following paragraphs.

If the target is lost, and degreeOfOccAll is calculated as 100% at the end of the previous iteration, then, "Full Occlusion ROI Dense Search Module", shown in Figure

5.14, takes the role of detecting and tracking the lost target during the present iteration as depicted in Figure 5.4. For a better explanation, image sequences, belonging to a different scenario video from the S-UAV dataset [10], are utilized in Figure 5.14, except the ones, shown in Figure 5.4.

The reason for using different image sequences is to show the processes followed during the full occlusion state and how a target is detected after the full occlusion state is finished. Before starting to explain the details of the process during the full occlusion state, note that *output on* im_{t-1} in Figure 5.14 is the output of the method in Figure 5.4 when im_{t-2} and im_{t-1} are input images. A red circle is drawn on im_{t-2} to border a small part of the target or car, visible at time $t-2$. The car is fully occluded at im_{t-1}, and the full occlusion state occurs as displayed at *output on* im_{t-1}. At that moment, time is incremented to $t = t+1$, and the full occlusion state starts.

At the beginning of the full occlusion state, two images, im_{t-1} and im_t, are read, and $predictedState_t$ is updated using Equation (5.6) with the motion vector, calculated during the previous iteration. Then, the "Closed-Loop Target Detection Processor" sub-module takes the role of processing the dense search method based on an iterative sampling loop around the target as shown in Figure 5.14. First of all, the loop for dense search or sampling is initialized to be able to sample eight, 250X250 ROI's. Here $m, m = 0, 1, ..., 7$, stands for the number of ROI that are sampled during the iteration in the loop. For each mth sample within the loop, the number of pixels along the x and y axes to be sampled, $sample_x$, and $sample_y$, change to cover four sides of the last known position of the target, $predictedState_t$. Then, $predictedState_t$ is updated again to sample an ROI with $sample_x$ and $sample_y$ to get $predictedState_t^m$ as

$$\begin{aligned} samplingVector &= [sample_x, sample_y, 0, 0] \\ predictedState_t^m &= predictedState_t + samplingVector \end{aligned} \tag{5.7}$$

and regions, corresponding to $predictedState_t^m$, are displayed with yellow bounding

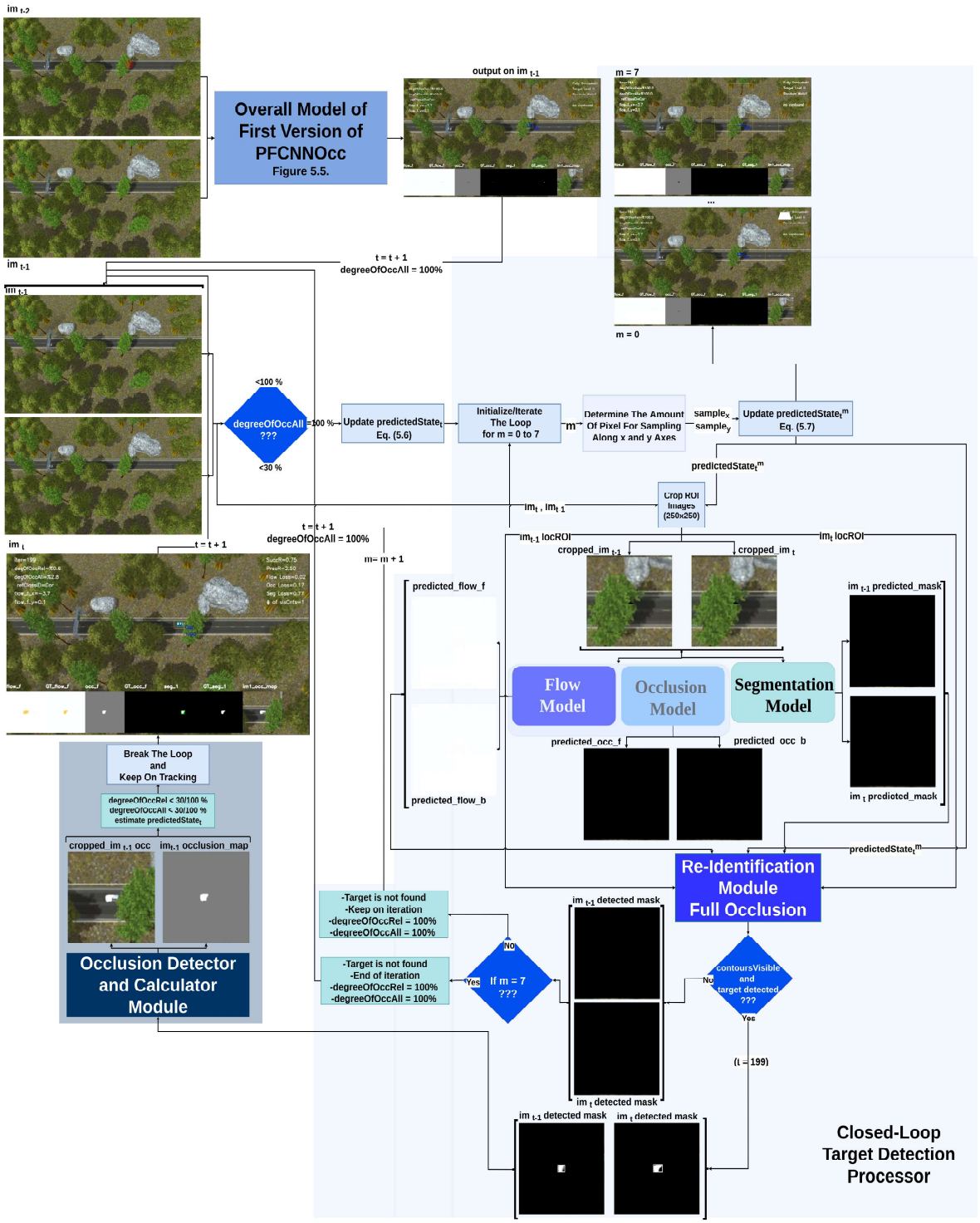


Figure 5.14. Full occlusion ROI dense search module.

boxes for all sampled ROIs as shown in Figure 5.14. The updated $predictedState_t^m$ is tailored to crop ROI images (250X250), $cropped_im_{t-1}$ and $cropped_im_t$, and they are fed to "Flow and Occlusion Model"and "Segmention Model"to find $predicted_flow_f$, $predicted_flow_b$, $predicted_occ_f$, $predicted_occ_b$, im_{t-1} $predicted_mask$ and im_t $predicted_mask$ as depicted in Figure 5.14. Then, $predicted_flow_f$, $predicted_flow_b$, im_{t-1} $predicted_mask$ and im_t $predicted_mask$, $predictedState_t^m$ as well as vectors, storing the location and size information of ROIs at original or global image coordinates, $im_{t-1}locROI$ and $im_tlocROI$ are sent to the novel "Re-Identification Module Full Occlusion", shown in Figure 5.16. "Re-Identification Module Full Occlusion"differs from "Re-Identification Module", shown in Figure 5.5, in terms of additive functionality, designed to filter out the false alarm contours and to detect the target during full occlusion state. The architecture of the novel "Re-Identification Module Full Occlusion"is elaborated in as a pseudocode in Figure 5.15. The most important and unique functionality of this module is the calculation of the confusion scores between the contours, detected in im_{t-1} $predicted_mask$ and im_t $predicted_mask$, and deciding the best matching ones to re-identify the target after the full-occlusion state. At the end of "Re-Identification Module Full Occlusion", three different paths can be followed depending on the result of the target detection as displayed in Figure 5.16.

If the target is not detected by "Re-Identification Module Full Occlusion", and m, the number of sampled ROI during loop iteration, is smaller than seven, the iteration in the loop continues to sample another ROI search area, m is incremented with one, "Target Is Not Found"warning is issued, and occlusion metrics, degreeOfOccRel and degreeOfOccAll, are kept as 100%, meaning "Full Occlusion". If the target is not detected by "Re-Identification Module Full Occlusion", and m equals seven, the iteration in the loop ends, no more ROI sampling is performed, "Target Is Not Found"warning is issued, and occlusion metrics, degreeOfOccRel and degreeOfOccAll, are kept as 100%, tracking process returns to the beginning of the method via incrementing time t by one, and a pair of new images are read for the next iteration which follows the path to detect the target under "Full Occlusion"state.

Input: im_{t-1} $locROI, im_t$ $locROI, predicted_masks, predicted_flows$ im_{t-1} $refPixelCount, im_{t-1}$ $refBboxSize$

Output: $detected_bboxes$

function RE-IDENTIFICATION FULL OCCLUSION(.) ▷ Runs the algorithm
  $im_{t-1/t}$ $predicted_mask_filt \leftarrow$ **medianFilter**($im_{t-1/t}$ $predicted_mask$)
  $im_{t-1/t}$ $contours \leftarrow$ **findContours**($im_{t-1/t}$ $predicted_mask_filt$)
  $locROI_min_x/y, locROI_max_x/y \leftarrow$ **determineLocROI**(im_{t-1} $locROI$)
  $pixelCountLowerBound, sizeTargetLowerBound \leftarrow$ **updateRefFeat()**
  for $i \leftarrow 1, size(im_{t-1/t}$ $contours)$ **do** ▷ Check size and visibility of contours
    $[contour_bboxArray, min_x/y, max_x/y] =$ **findLoc**($im_{t-1/t}$ $contours[i]$))
    if $max_x - min_x \leq 2 \mid max_y - min_y \leq 2$ **then** ▷ Check size
      ignore $im_{t-1/t}$ $contours[i]$ ▷ Ignore outlier
    else ▷ Check visibility in locROI
      if $contour_bboxArray$ $inside$ $locROI$ **then** ▷ Visibility satisfied
        $visible_contour_list \leftarrow$ **storeVisibleContours**($min_x/y, max_x/y$)
        $motionList_x/y \leftarrow$ **storeMotionVector**($predicted_flow_f$)
        $avgMotionVector, pixelCount, bboxSize \leftarrow$ **storeFeats()**
        $contourFlowCheck \leftarrow$ **filterOutContour**($contourFeat, refFeat$)
      else ▷ Visibility not satisfied
        ignore $im_{t-1/t}$ $contours[i]$ ▷ Ignore contour
      end if
    end if
  end for
  $confusionMatrix[i, j] \leftarrow$ **calcConfusionScore()** ($Eq.5.5$) ▷ Calculate confusion scores of contours in im_{t-1} and im_t, satisfying step 15.
  $im_{t-1/t}$ $detected_bbox \leftarrow$ **findDetectedBbox**($confusionMatrix$) ▷ Find the coords. of detected bboxes, using contours, satisfying step 21
end function

Figure 5.15. The pseudocode of the Re-Identification Module Full Occlusion.

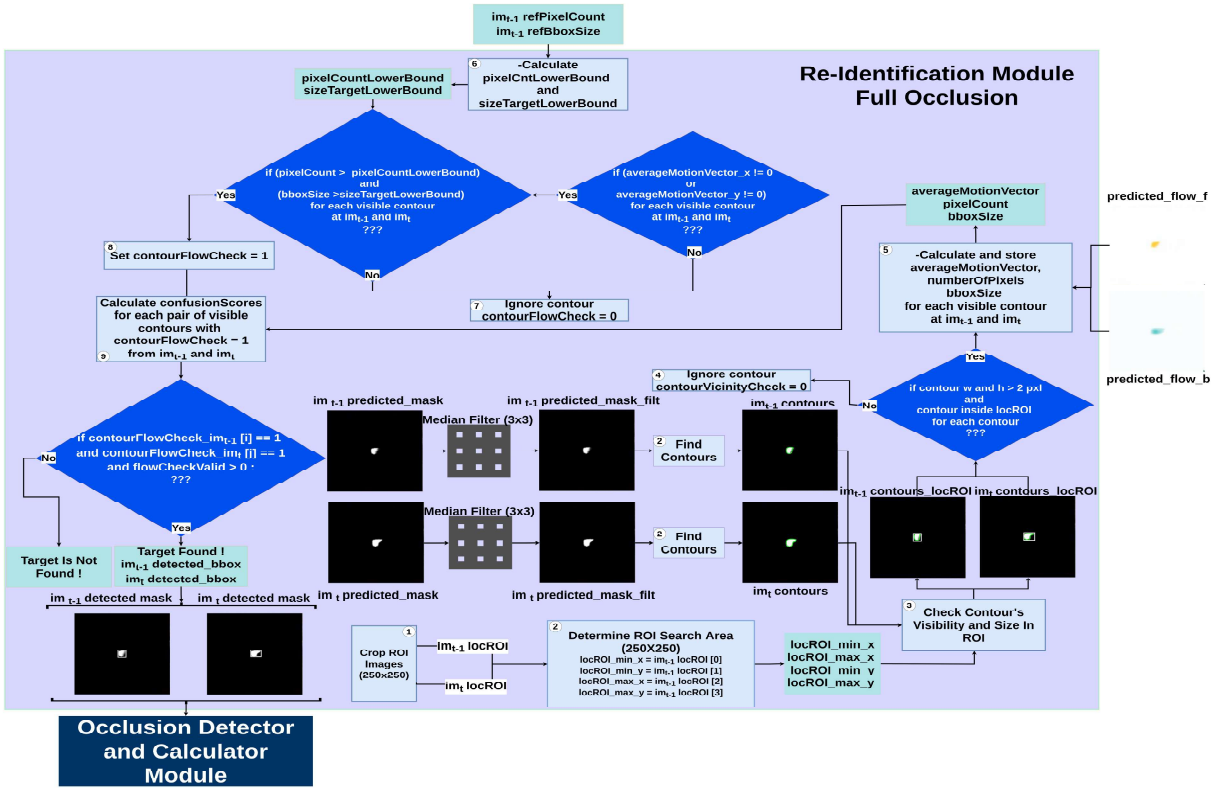


Figure 5.16. Re-Identification module full occlusion.

The third and last path is the one that will be followed after the target is detected, as shown in Figure 5.14. After the target is fully occluded at the end of the 183rd image, the "Full Occlusion"state stays until the 199th image when the target is detected by "Re-Identification Module Full Occlusion". Detection results, $im_{t-1}detected_mask$ and $im_{t}detected_mask$, are sent to "Occlusion Detector and Calculator Module"to detect the occlusion, calculate the degree of it, and estimate $predictedState_t$ as depicted in Figures 5.7, 5.8, 5.10, and 5.14, and as elaborated in Figures 5.9, 5.11, and 5.12. Then, the loop for ROI sampling is broken without sampling the remaining possible ROIs, and the $predictedState_t$ is shown at im_t, and time t is incremented by one. A pair of new images is read for the next iteration, which follows the path to detect the target under the "No Occlusion"or "Partial Occlusion"state as shown in Figures 5.4 and 5.14.

5.3.1.2. The Final Version of PFCNNOcc Tested in Real-Time UAV Dataset, UAV123.

The final version of the PFCNNOcc method tailors CNN models, trained using UAV123 dataset [2] as explained in Section 5.2, to detect optical flow, occlusion, segmentation, and classification during tracking under severe visual problems such as out-of-view, partial, and full occlusion. The final version of the PFCNNOcc resolves the unexpected issues faced during the initial tests of the first version of PFCNNOcc with UAV123 dataset [2]. The reason for these problems is the differences between scenarios from the S-UAV dataset [10] and UAV123 [2]. These differences are itemized as

- There is a minor scale variation in the S-UAV dataset [10], but the UAV123 dataset [2] consists of sudden and large-scale changes in the image of the target. Also, the size of the vehicles in the S-UAV dataset does not show a huge variance, the vehicles have the optimal size that fits in the ROI window of size 250x250 easily with a high-resolution image quality. On the other hand, some vehicles in UAV123 dataset [2] are small in size and have very low resolution, and the size of some is even larger than the size of the ROI window.
- The vehicles in the S-UAV dataset [10] navigate almost at the same velocity with an equal amount of pixel displacement between the video frames. In the UAV123

dataset [2], the vehicles demonstrate different motion types such as stop, forward, and backward motion with acceleration or deceleration, or constant but different speed levels.

- The vehicles in the S-UAV dataset [10] do not occlude each other, hence they do not merge while driving. However, the tracked vehicle is occluded and also merged with other vehicles in the UAV123 dataset [2].

Hence, the overall architecture of the final version of PFCNNOcc, resolving the issues, itemized above, and outperforming the SOTA trackers in the UAV123 dataset [2] against all visual problems especially partial and full occlusion, is shown in Figure 5.17. At the beginning of the iteration, two consecutive images of size 720x1280 (height,width), im_{t-1} and im_t, are read from the folder where the video or image sequences from the UAV123 dataset [2] are stored. At this point, the way to approach tracking problems changes vitally depending on the percentage of degreeOfOccAll as explained in Section 5.3.1.1. Hence, there are three different paths to choose for three different states; 1. No Occlusion, 2. Partial Occlusion and 3. Full Occlusion. The diagram in Figure 5.3, illustrating the transition of these states, is also valid for the final version of PFCNNOcc.

While the details of the final version of PFCNNOcc are explained in this section, the common functions, also explained for the first version of PFCNNOcc in Section 5.3.1.1, are skipped, and only the important and newly designed or evolved functions are considered. Before going deep into the flow diagram, shown in Figure 5.17, these vital and recently developed, novel functionalities are summarized as;

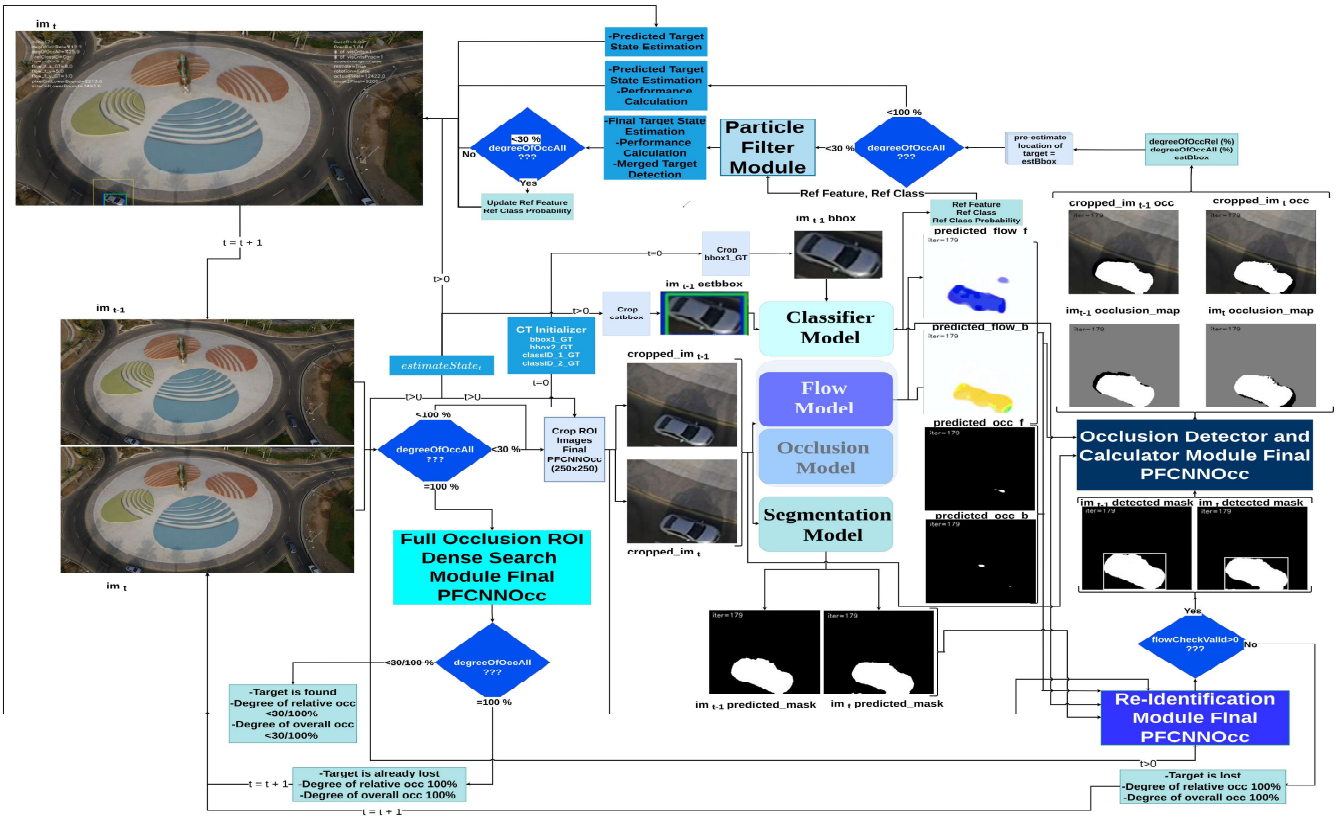


Figure 5.17. Overview of overall flow diagram of the final version of PFCNNOcc

- Calculation and the control of the scale and aspect ratio variations to get rid of false alarms in occlusion detection,
- Decision to rescale the target depending on its size from the previous iteration,
- Detection of the sudden occlusion by another vehicle or the merged condition between the tracked vehicle and the others,
- Design of the evolved "Re-Identification Module Final PFCNNOcc", "Occlusion Detector and Calculator Module Final PFCNNOcc", and "Full Occlusion ROI Dense Search Module Final PFCNNOcc"depending on the contributions, mentioned above.

The functionality of the path that will be followed during No Occlusion ($< 30\%$) state, shown in Figure 5.17, is explained in the following paragraphs. degreeOfOccAll is set as 0 before the tracking process starts. targetLostIdx is set to 0 to keep the counter of the full occlusion state during tracking. For the first image pair at $t = 0$, the location, the size, and the class information of the bounding box of the target on im_{t-1} are read from the ground truth file, and the tracking process starts. Then, ROI boxes, $cropped_im_{t-1}$ and $cropped_im_t$ (3x250x250 (channelXwidthXheight)), are cropped from both images, im_{t-1} and im_t, using "Crop ROI Images Final PFCNNOcc"module. This module decides whether rescaling is required. If the width and height of the target are smaller than 75 pixels, then the rescale variable is set to "true"and the images are rescaled by 2. degreeOfOccAll and actual_non_occ_pixel_cnt counters are multiplied by 4 if the rescaling state is changed from no rescaling to upscaling condition at this iteration. Otherwise, if the width or the height is larger than 200 pixels, rescale is set to "downscale", and the images are downscaled by 2. degreeOfOccAll and actual_non_occ_pixel_cnt counters are divided by 4 if the rescaling state is changed from no rescaling to downscaling condition at this iteration. If both conditions mentioned above are not met, then the rescaling is not applied, and the rescaling is set to "false". Since the rescaling state is just changed in this iteration to no rescaling condition, the correction to degreeOfOccAll and actual_non_occ_pixel_cnt counters is applied once, as mentioned above, as long as no rescaling condition is retained. The lower bounds for reference pixel count and reference bounding box size,

pixelCountLowerBound and *sizeTargetLowerBound*, are also updated adaptively by the division of actual_non_occ_pixel_cnt with 4 that are used by "Re-Identification Module Final PFCNNOcc"as illustrated in Figure 5.20.

After cropping the ROI images, the local ROI search area is calculated adaptively as

$$\begin{aligned} locROI_min_x &= bbox1_GT_x - bbox1_GT_w \\ locROI_min_y &= bbox1_GT_y - bbox1_GT_h \\ locROI_max_x &= bbox1_GT_x + bbox1_GT_w * 2 \\ locROI_max_y &= bbox1_GT_y + bbox1_GT_h * 2 \end{aligned} \tag{5.8}$$

where the location and the size information of $bbox1_GT$ are used to find locROI at $t = 0$. While $t > 0$, the location and the size information of the bounding box from $estimateStates_{t-1}$ are tailored to calculate the locROI search area.

If $t = 0$, the bounding box, $im_{t-1}bbox$, is cropped using the ground truth data and is utilized to extract the different reference features: 1. The normalized histogram of the color channels in RGB and HSV spaces. 2. The reference feature, the reference class ID, and the reference class probability. While the reference histograms of the different color channels are utilized in "Re-Identification Final PFCNNOcc Module"to re-identify the tracked target, the reference feature and reference class ID are used in the "PFCNN Module"to find the likelihood and class scores of particles and to update their weights, as explained in Section 3.1. Then, the cropped images are fed into "Flow Model and Occlusion Model"as well as "Segmentation Model"to predict the forward and backward optical flow maps, $predicted_flow_f$ and $predicted_flow_b$, forward and backward occlusions maps, $predicted_occ_f$ and $predicted_occ_b$, segmented masks, $im_{t-1}_predicted_mask$ and $im_t_predicted_mask$, for both images as shown in Figure 5.17.

After predictions for optical flow, occlusion, and segmentation are computed, "Re-Identification Final PFCNNOcc Module"starts. The pseudocode, elaborating the logic of the algorithm, is given in Figure 5.18. In the final step of the pseudocode, $confusionScore$ between the detected contours in im_{t-1} and im_t is calculated as

$$
\begin{aligned}
RGB_hist_im_{t-1/t} &= histogram_RGB(cropped_im_{t-1/t}) \\
HSV_hist_im_{t-1/t} &= histogram_HSV(cropped_im_{t-1/t}) \\
eucDist_im_{t-1/t} &= \sqrt{\sum_{m=0}^{1}\Big(estimateStates_{t-1}[m] - im_{t-1/t}_bbox\Big)^2} \\
cosDist_RGB_im_{t-1/t} &= \frac{RGB_hist_im_{t-1/t} \cdot ref_RGB_Hist}{\left\|RGB_hist_im_{t-1/t}\right\|_2 \left\|ref_RGB_Hist\right\|_2} \\
cosDist_HSV_im_{t-1/t} &= \frac{HSV_hist_im_{t-1/t} \cdot ref_HSV_Hist}{\left\|HSV_hist_im_{t-1/t}\right\|_2 \left\|ref_HSV_Hist\right\|_2} \\
cosDistTotal &= cosDist_RGB_im_{t-1} * cosDist_HSV_im_{t-1} \\
&\quad * cosDist_RGB_im_t * cosDist_HSV_im_t \\
eucDistTotal &= eucDist_im_{t-1} * eucDist_im_t \\
confusionScore &= \frac{cosDistTotal}{eucDistTotal}
\end{aligned}
\tag{5.9}
$$

where $eucDist_im_{t-1}$ and $eucDist_im_t$ are the Euclidean distance measurements between the vectors, storing the corner position of the bounding box for the pair of visible contours and the estimated target position in the previous iteration, $estimateStates_{t-1}$, and $cosDist_RGB_im_{t-1}$, $cosDist_HSV_im_{t-1}$, $cosDist_RGB_im_t$, $cosDist_HSV_im_t$ refer to the cosine distance measurements between the feature vectors, storing histograms of RGB and HSV channels of cropped images, $cropped_im_{t-1/t}$, and reference RGB and HSV histogram features, ref_RGB_hist and ref_HSV_hist. The $confusionScore$ is calculated by the division of $cosDistTotal$ with $eucDistTotal$. While the features in RGB and HSV histograms of the cropped images and reference images get closer, $cosDistTotal$ increases and converges to 1. On the other hand, when the distances between the location of bounding boxes of the visible contours and the estimated location of the target are close, $eucDistTotal$ decreases and converges to 0. Therefore, $confusionScore$ is calculated to measure how much a pair

of visible contours resemble each other. The more similar they are to each other, to higher the $confusionScore$. The $confusionScores$ for each pair of contours are stored in $confusionMatrix$. The way the $confusionMatrix$ is calculated is one of the most important and unique features of "Re-Identification Final PFCNNOcc Module", compared to the "Re-Identification Module"and "Re-Identification Module Full Occlusion"of the first version of PFCNNOcc, explained in Section 5.3.1.1.

When the re-identification of the target is completed, the "Occlusion Detector and Calculator Module Final PFCNNOcc", designed for the final version of PFCNNOcc, is utilized to detect or fine-tune forward and backward occlusion maps as well as to calculate the relative occlusion, degreeOfOccRel, and the overall occlusion, degreeOfOccAll. Otherwise, it is declared as full occlusion, and the function for the full occlusion case runs for the next iteration. As illustrated in Figure 5.19, the novel "Occlusion Detector and Calculator Module Final PFCNNOcc"is designed with the combination of three distinct functions: 1. "Forward Processor", 2. "Backward Processor", and 3. "Occlusion Calculator Final PFCNNOcc". "Forward Processor"and "Backward Processor"sub-modules, designed for the "Occlusion Detector and Calculator Module"of the first version of the PFCNNOcc, are already utilized in "Occlusion Detector and Calculator Module Final PFCNNOcc"as they are, and the functionality of these sub-modules is not detailed here. But the "Occlusion Calculator Final PFCNNOcc"is re-designed, and utilized with the new functions that are developed to keep track of the aspect ratio and scale variations of the estimated bounding box of the target to get rid of the false alarm conditions during the occlusion calculations. The function of this sub-module is key in deciding the reasons for the pixel reduction on the scene.

If the pixel count of a target is decreasing during tracking, there are two reasons: 1. Decrease in scale, mostly associated with aspect ratio variation, 2. Occlusion. "Occlusion Calculator Final PFCNNOcc"is a function at the last phase of "Occlusion Detector And Calculator", and is designed to calculate the "degreeOfOccRel"and "degreeOfOccAll"as well as the preliminary estimate of bounding box location and size

Input: im_{t-1} $locROI, im_t$ $locROI, predicted_masks, predicted_flows$
im_{t-1} $refPixelCount, im_{t-1}$ $refBboxSize, locROI_min_x/y, locROI_max_x/y,$
$pixelCountLowerBound, sizeTargetLowerBound, cropped_im_{t-1/t}, ref_RGB_Hist$
ref_HSV_Hist

Output: $detected_bboxes$

function RE-IDENTIFICATION FINAL PFCNNOCC(.) ▷ Runs the algorithm
  $im_{t-1/t}$ $predicted_mask_filt \leftarrow$ **medianFilter**($im_{t-1/t}$ $predicted_mask$)
  $im_{t-1/t}$ $contours \leftarrow$ **findContours**($im_{t-1/t}$ $predicted_mask_filt$)
  for $i \leftarrow 1, size(im_{t-1/t}$ $contours)$ **do** ▷ Check size and visibility of contours
    $[contour_bboxArray, min_x/y, max_x/y] =$ **findLoc**($im_{t-1/t}$ $contours[i]$))
    if $max_x - min_x \leq 2 \mid max_y - min_y \leq 2$ **then** ▷ Check size
      ignore $im_{t-1/t}$ $contours[i]$ ▷ Ignore outlier
    else ▷ Check visibility in locROI
      if $contour_bboxArray$ $inside$ $locROI$ **then** ▷ Visibility satisfied
        $visible_contour_list \leftarrow$ **storeVisibleContours**($min_x/y, max_x/y$)
        $motionList_x/y \leftarrow$ **storeMotionVector**($predicted_flow_f$)
        $avgMotionVector, pixelCount, bboxSize \leftarrow$ **storeFeats()**
        $contourFlowCheck \leftarrow$ **filterOutContour**($contourFeat, refFeat$)
        $classScoreIdx \leftarrow$ **calcClassIdx**($cropped_im_{t-1/t}$)
      else ▷ Visibility not satisfied
        ignore $im_{t-1/t}$ $contours[i]$ ▷ Ignore contour
      end if
    end if
  end for
  if Steps 13 and 14 satisfied **then**
    $confusionMatrix \leftarrow$ **calcConfusionScore()** ($Eq.5.9$)
    $im_{t-1/t}$ $detected_bbox \leftarrow$ **findDetectedBbox**($confusionMatrix$)
  else
    $degreeOfOccAll, degreeOfOccRel(\%) \leftarrow 100\%$ ▷ Target is lost/not found.
  end if
end function

Figure 5.18. The pseudocode of the Re-Identification Module Final PFCNNOcc.

in 250X250 local image coordinates, *estBox*, using the inputs from "Forward Processor"and "Backward Processor". The pseudocode, elaborating the "Occlusion Calculator Final PFCNNOcc", is given in Figure 5.21.

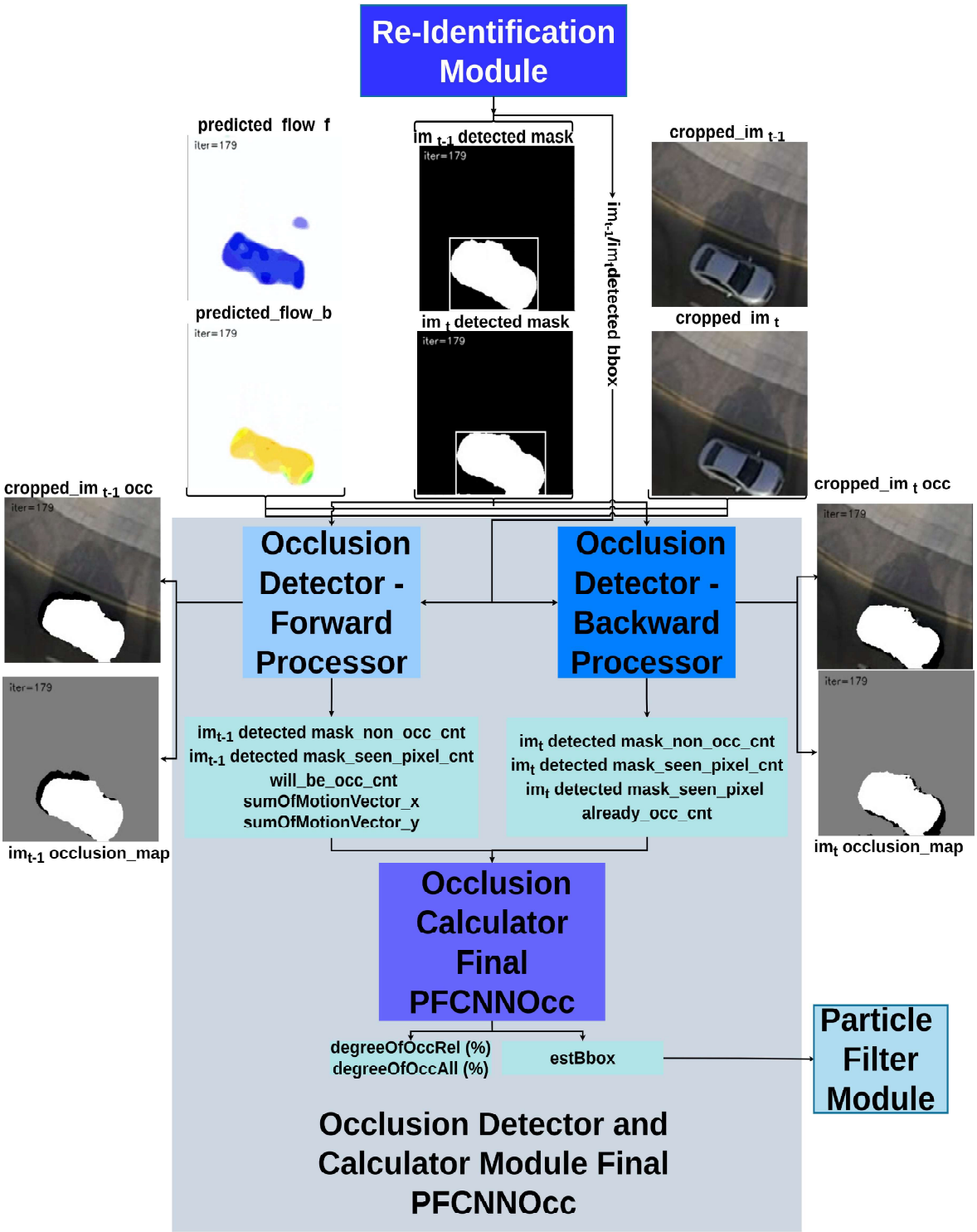


Figure 5.19. Occlusion detector and calculator module final PFCNNOcc.

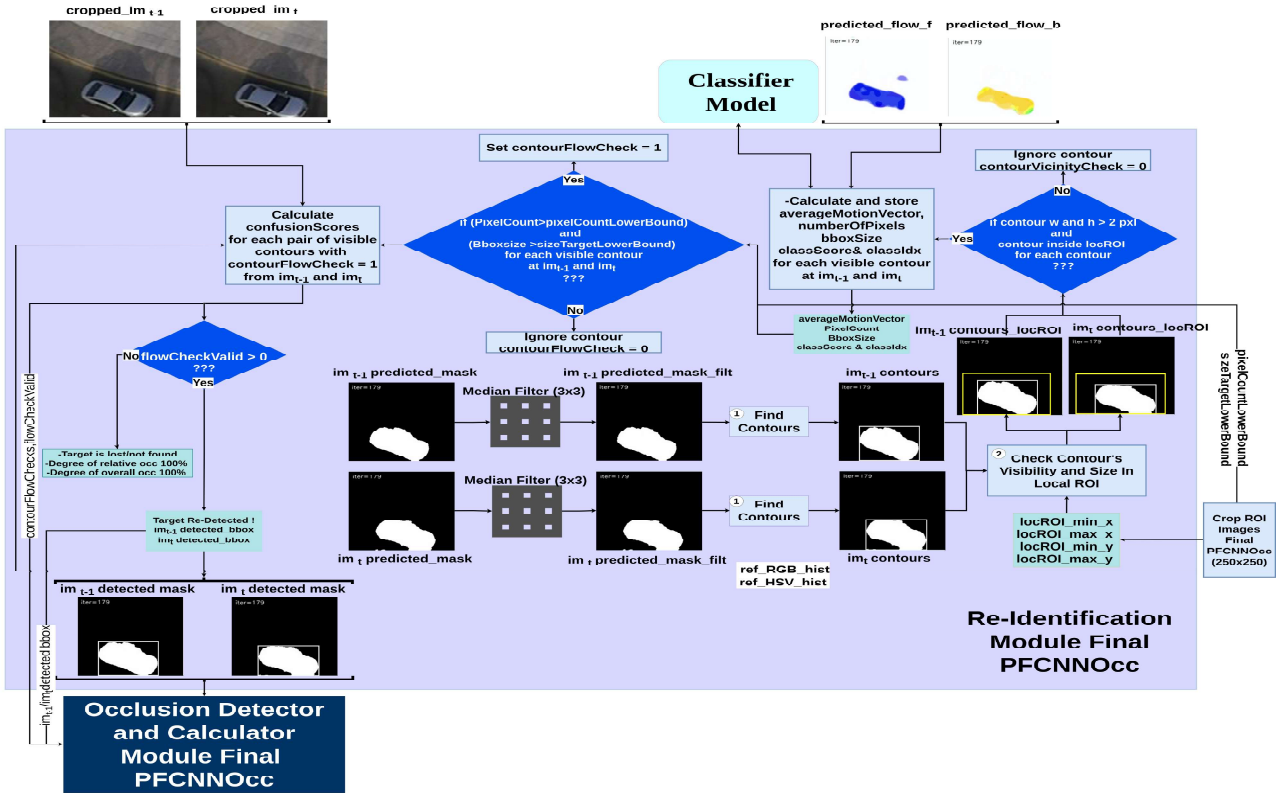


Figure 5.20. Re-Identification module final PFCNNOcc.

The final phase of the tracking process is the estimation of the state variables using the "Particle Filter Module", shown in Figure 5.13. It is utilized in the same manner as designed and used in the first version of PFCNNOcc, elaborated in Section 5.3.1.1, hence, the usage of "Particle Filter Module"is not detailed in this section. After the estimate of the state variables, $estimateState_t$, is calculated, the rescaling, applied in the beginning of the iteration depending on the size of the target, is applied again in a reverse manner to find the value of the estimate of the state variables in original image coordinates. Then, $estimateState_t$ is utilized to crop images at the beginning of the next iteration as illustrated in Figure 5.17.

Another important functionality, designed exclusively for the final version of the PFCNNOcc, is called "Merged Target Detection". As understood from the title of the function, it is aimed to detect the condition or instant that the tracked target is suddenly occluded by a different vehicle or merged while they navigate almost side-by-side. As a consequence of this instant, the vehicles seem to be merged, and the segmentation masks of the sampled ROI images show that the number of pixels, belonging to the tracked car, becomes very large as shown in Figure 5.22. Merged target condition is detected by comparing the areas of the bounding boxes depending on the estimate of state variables at previous and current iterations, $estimateState_{t-1}$ and $estimateState_t$. If the area of the bounding box of the target at time, t, is larger than the product of the adaptive coefficient and the area of the target at time $t-1$, then the merged target condition is flagged, and a counter for this instant is started. A threshold for the area of the bounding box is stored so as to keep it as a reference value in the beginning of this condition, and $estimateState_t$ is updated as

$$\begin{aligned} estimateState_t[0:1] = & estimateState_{t-1}[0:1] + \\ & [avgMotionVector_x, avgMotionVector_y] \end{aligned} \tag{5.10}$$

where $avgMotionVector_x$ and $avgMotionVector_y$ refer to the reference motion vector with which the tracked target is navigating. Hence, updating $estimateState_t$ with Equation (5.10) eliminates the error, introduced in the estimation process of the location of the target at time t, $estimateState_t[0:1]$. These errors are introduced by the merged targets, illustrated in Figure 5.22. The most important reason to update

Input: $im_{t-1/t}_detected_mask_non_occ_cnt, im_{t-1/t}_detected_mask_seen_pixel_cnt, will_be_occ_cnt, sumOfMotionVectors, im_t_detected_mask_seen_pixel, already_occ_cnt$

Output: $degreeOfOccRel, degreeOfOccAll, estBbox, avgMotionVectors$

function OCCLUSION CALCULATOR FINAL PFCNNOCC(.)
  $estBbox[min_x/y, max_x/y]$ = **findLoc**$(im_t_detected_mask_seen_pixel)$
  if $t == 0$ **then**
    $actual_non_occ_pixel_cnt \leftarrow im_{t-1}_detected_mask_seen_pixel_cnt$
    $aspectRatioRef \leftarrow \frac{estBbox[2]-estBbox[0]}{estBbox[3]-estBbox[1]}$
  else
    $aspectRatio \leftarrow \frac{estBbox[2]-estBbox[0]}{estBbox[3]-estBbox[1]}$
    if $im_t_detected_mask_seen_pixel_cnt \geq actual_non_occ_pixel_cnt$ **then**
      if $aspectRatioRef * 0.9 \leq aspectRatio \leq aspectRatioRef * 1.1$ **then**
        $scaleChange \leftarrow True,\ rotation \leftarrow False,\ degreeOfOccAll(\%) \leftarrow 0$
      else
        $scaleChange \leftarrow True,\ rotation \leftarrow True,\ degreeOfOccAll(\%) \leftarrow 0$
      end if
      $actual_non_occ_pixel_cnt \leftarrow im_t_detected_mask_seen_pixel_cnt$
      $aspectRatioRef \leftarrow aspectRatio$
    else
      if $aspectRatioRef * 0.9 \leq aspectRatio \leq aspectRatioRef * 1.1$ **then**
        $scaleChange \leftarrow True,\ rotation \leftarrow False,\ degreeOfOccAll(\%) \leftarrow 0$
        $actual_non_occ_pixel_cnt \leftarrow im_t_detected_mask_seen_pixel_cnt$
        $aspectRatioRef \leftarrow aspectRatio$
      else
        $scaleChange \leftarrow False,\ rotation \leftarrow False$
        $degreeOfOccAll(\%) \leftarrow \frac{(actual_non_occ_pixel_cnt - im_t_detected_mask_seen_pixel_cnt)*100}{actual_non_occ_pixel_cnt}$
      end if
    end if
  end if
  $degreeOfOccRel(\%) \leftarrow \frac{will_be_occ_cnt*100}{im_t_detected_mask_non_occ_cnt+will_be_occ_cnt+already_occ_cnt}$
end function

Figure 5.21. The pseudocode of the Occlusion Calculator Final PFCNNOcc.

$estimateState_t$ with a random walk model is to keep the estimate of the location of the target in its own trajectory. Using the $estimateState_t$ in "Re-Identification Final PFCNNOcc Module", shown in Figure 5.20, enables to re-detect the tracked target correctly when the merged target condition vanishes.

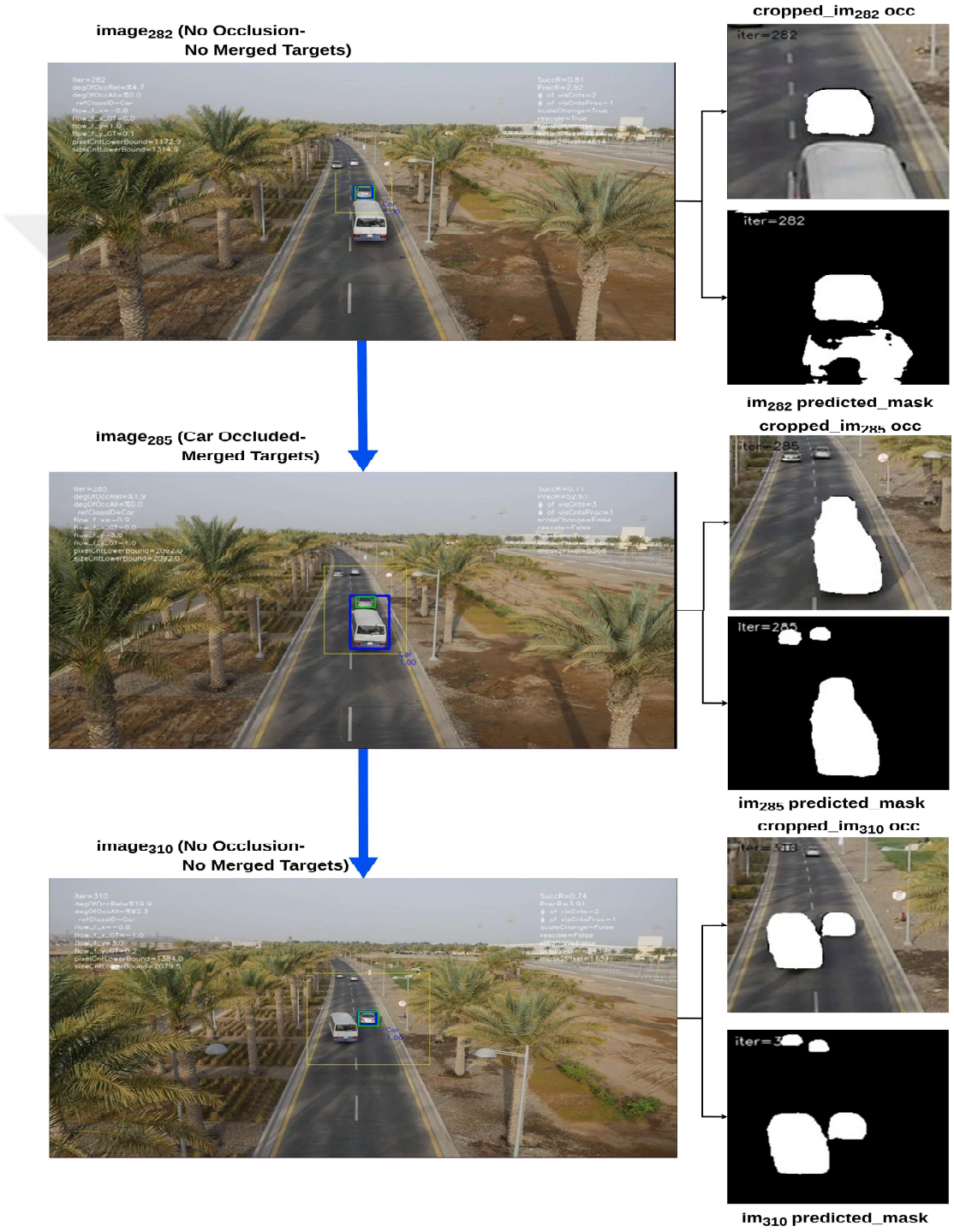


Figure 5.22. Merged target detection problem.

During the end of the next iterations, if the area of the estimated bounding box of the target at time t, $estimateState_t$, is larger than the product of the adaptive threshold and the reference value, the merged target condition keeps on, the counter for the condition is incremented, and $estimateState_t$ is updated as given in Equation (5.10). Otherwise, it is understood that the merged target condition ends, the counter is set to 0 for the re-initialization in case of a possible merged target condition, and $estimateState_t$ is update by equalizing to $estimateState_{t-1}$ to resume the tracking from the last know position of the target as shown in Figure 5.22. The adaptive coefficient is calculated by averaging the scale factors, found by the division of the areas of the estimated bounding boxes of the target, $estimateState_{t-1}$ and $estimateState_t$, up to the current iteration, and multiplying the average by a constant, 1.5. This constant is chosen to consider the maximum scale variation between consecutive frames. In UAV datasets, especially in the UAV123 dataset [2], it is not possible to face such a huge scale variation between two frames compared to the adaptive coefficient. If this adaptive coefficient is exceeded, it should be just an indication of a merged-target condition. This is the end of a tracking iteration if degreeOfOccAll is smaller than 30 %.

The functionality of the path that will be followed during Partial Occlusion (30% < & <100%) state, shown in Figure 5.4, is explained in the following paragraphs. Instead of explaining all processes on this path, just the differences from the path, followed during degreeOfOccAll, is smaller than 30 %, are elaborated. The initialization, ROI image sampling, and the application of CNN models are the same until “Re-Identification Module Final PFCNNOcc”, as illustrated in Figure 5.20. The differences in “Re-Identification Module Final PFCNNOcc”for this path are itemized as;

- pixelCountLowerBound and sizeTargetLowerBound are set to 0 instead of an adaptive number, calculated in the path while the degreeOfOccAll is smaller than 30 %. The reason for taking these threshold values as 0 is the uncertainty about the percentage of the occlusion, the target faces.

- While the confusionScores for each pair of visible contours from im_{t-1} and im_t are calculated, the check conditions for contourClassIdx's are ignored, because the target is highly deformed depending on the percentage of the occlusion, and the "Classifier Model"would give misleading results accordingly.
- If the detected contours that does not satisfy the validity of the contoursVisible, contourFlowChecks, flowCheckValid flags or if no contour is detected, then the target lost condition is issued, degreeOfOccRel and degreeOfOccAll set to 100% and the "Full Occlusion ROI Dense Search Module Final PFCNNOcc" function is followed during next iterations.

The last difference is that "Particle Filter Module"is not utilized in this path. The predicted target state value is set to the pre-estimate location of the target, and also equalized with $estimateState_t$. This is the end of a tracking iteration if $30\% < degreeOfOccAll < 100\%$. In summary, the iteration starts with "Partial Occlusion", because degreeOfOccAll was between 30% and < 100%. Depending on the tracking process, explained up to now, the state of occlusion can transition to No Occlusion and Full Occlusion as different states. Also, it can stay at its own state, Partial Occlusion, as shown in Figure 5.3.

The functionality of the path that will be followed during Full Occlusion (= 100%) state, shown in Figure 5.17, is explained in the following paragraphs.

If the target is lost, and degreeOfOccAll is calculated as 100% at the end of the previous iteration, then, "Full Occlusion ROI Dense Search Module Final PFCNNOcc", shown in Figure 5.23, takes the role of detecting and tracking the lost target during the present iteration as depicted in Figure 5.17. It should be noted that during the full occlusion, when im_{t-2} and im_{t-1} are input images, $outputonim_{t-1}$ is shown in Figure 5.23 as an output of the method in Figure 5.17. Red circle is drawn on im_{t-2} to border a small part of the target or car, visible at time t-2. The car is fully occluded at im_{t-1}, and the full occlusion state occurs as displayed at $outputonim_{t-1}$. At that moment, time is incremented as, $t = t + 1$, and the full occlusion state starts.

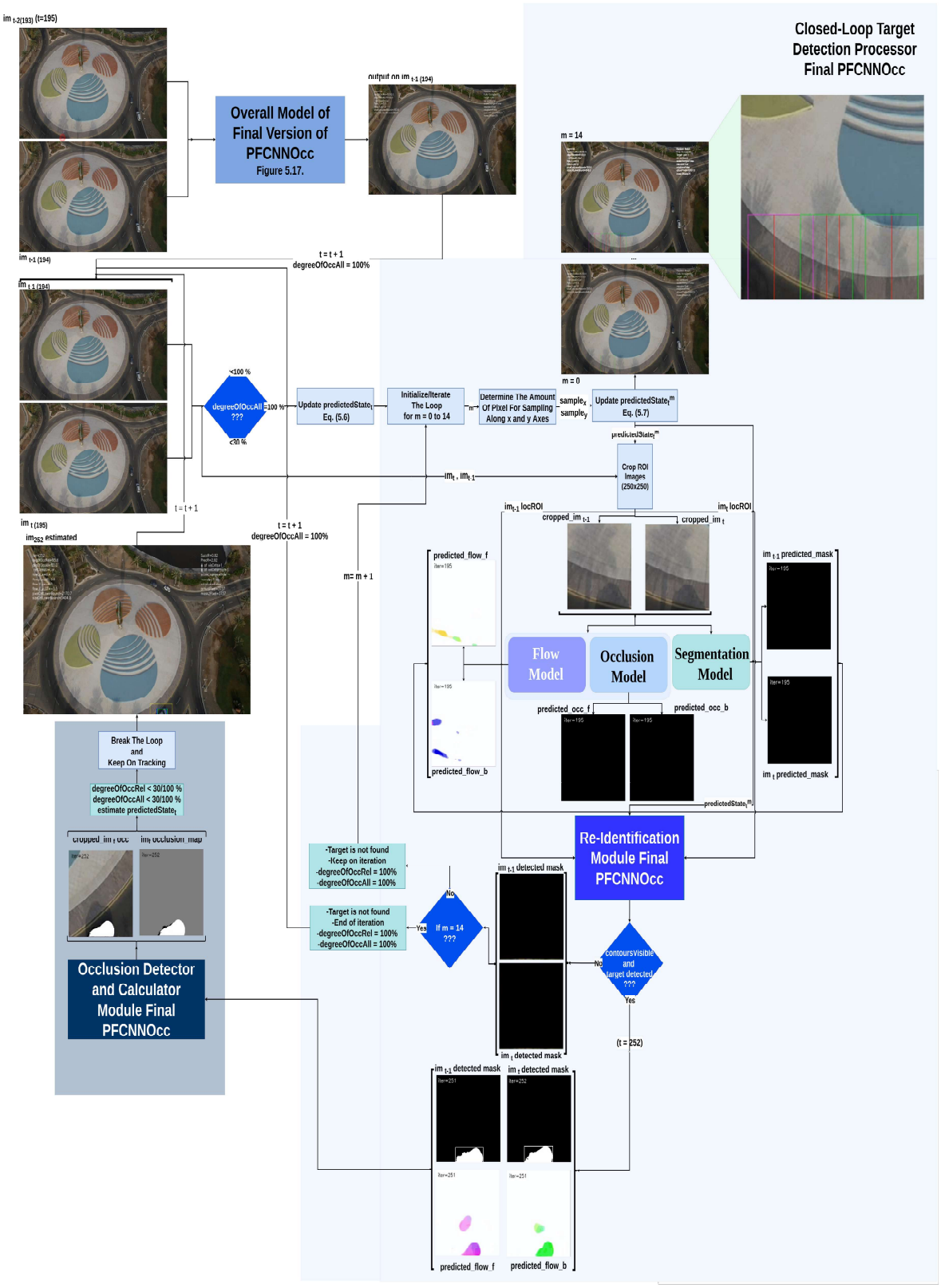


Figure 5.23. Full occlusion ROI dense search module final PFCNNOcc.

At the beginning of the full occlusion state, two images, im_{t-1} and im_t, are read, and $predictedState_t$ is updated as in Equation (5.6). Then, the "Closed-Loop Target Detection Processor Final PFCNNOcc"sub-module takes the role of processing the dense search method based on an iterative sampling loop around the target as shown in Figure 5.23. At this point, it is preferred to elaborate just the differences of the "Full Occlusion ROI Dense Search Module Final PFCNNOcc" in Figure 5.23 from the "Full Occlusion ROI Dense Search Module" in Figure 5.14, designed for the first version of PFCNNOcc that is explained in Section 5.3.1.1, and these differences are itemized as;

- The loop for dense search or sampling is initialized to enable to sample fifteen, 250X250 ROI's, $m, m = 0, 1, ..., 14$, using Equation (5.7), as shown with the different colors in Figure 5.23 instead of eight ROI's, sampled by the "Full Occlusion ROI Dense Search Module" in Figure 5.14 for the first version of PFCNNOcc. Sampling more ROI's around the predicted position of the occluded target in a dense manner increases the possibility of re-detection, especially, after a long-term occlusion. Hence, sampling more ROI means dense sampling. After a long-term full occlusion state occurs, it is impossible to predict the motion vector of the target. As long as the long-term occlusion continues, the error between the predicted motion vector and the actual motion vector accumulates. Hence, sampling densely alleviates the accumulated error and increases the possibility of detecting the target after the long-term full occlusion state vanishes.
- To re-detect the target in the sampled ROI's, the "Re-Identification Module Final PFCNNOcc", shown in Figure 5.20, is tailored instead of "Re-Identification Module Full Occlusion", designed for the first version of the PFCNNOcc. The details of the "Re-Identification Module Final PFCNNOcc"are elaborated under the explanations for the functionality of the path that will be followed during No Occlusion ($< 30\%$) state.

As illustrated in Figure 5.23, "Re-Identification Module Final PFCNNOcc"re-detects the target at t=252, from the "Full Occlusion ROI Dense Search Module Final PFCNNOcc", after a long-term full occlusion state starts at t=194. Even though the

long-term full occlusion state extends over 60 frames with a very large displacement, the target is detected successfully. The detected masks, im_{t-1} *detected mask* and im_t *detected mask*, are sent to the "Occlusion Detector and Calculator Module", designed for the first version of PFCNNOcc, to detect and calculate the occlusion and its degree. At the end of this process, the loop is broken to get rid of unnecessary, further ROI sampling and to change the Full Occlusion state to the "No Occlusion"or "Partial Occlusion"states, depending on the number of pixels of the targets in detected masks. The bounding boxes for the estimated values of the $predictedState_t$, the ground truth, and the ROI are drawn in blue, green, and yellow colors on the image, $im_{252}estimated$, respectively, as shown in Figure 5.23. Finally, a pair of new images is read for the next iteration, which follows the path to detect the target under the "No Occlusion"or "Partial Occlusion"state as shown in Figures 5.17 and 5.23.

5.4. Experiments

5.4.1. Experiments with the Training of the CNN Models

During the training of "Flow Model"and "Occlusion Model", EPE loss function is used for optical flow error, while the average F1 score function is utilized for occlusion error. Training is conducted for 100 epochs using the PyTorch library on two RTX 2080 TI GPUs. The batch size is set to 12, depending on the size of the CNN models and the images used during training, as well as the capacity of each GPU (11 GB). Adaptive Moment Estimation (ADAM) [90] is used as the optimizer with a learning rate and weight decay parameters of 10^{-5} and $4x10^{-4}$, respectively. The entire S-UAV dataset [10] is used for training, with 10% of the set utilized for validation. When the training stopped at the final epoch, validation losses for "Occ Model", F1 estimated moving average (F1-ema), and for "Flow Model", EPE-ema, are respectively, found to be 0.0169 and 0.1106.

The training of "Segmentation Model", shown in Figure B.6, is performed for 10 epochs using the PyTorch library on two RTX 2080 TI GPUs. The batch size is

set to 4 without any concern about the capacity of GPUs and the size of the CNN model, as well as the input images. Root Mean Squared Propagation (RMSProp) [96] is used as the optimizer with a learning rate and momentum parameters 10^{-5} and 0.999, respectively. The entire S-UAV dataset is used for training, with 10% of the set utilized for validation.

The re-training of "Segmentation Model", shown in Figure B.6, with the segmentation ground truth dataset of UAV123 [2], explained in Section 4.3, is performed for 150 epochs using the PyTorch library on two RTX 2080 TI GPUs. The batch size is set to 4 without any concern about the capacity of GPUs and the size of the CNN model as well as the input images. RMSProp [96] is used as the optimizer with a learning rate and momentum parameters of 10^{-5} and 0.999, respectively. Moreover, the optimizer has a scheduler to reduce the learning rate on a plateau. Gradient scaling and clipping are utilized to control gradient magnitude during learning. The entire segmentation ground truth of the UAV123 dataset [2] is used for training, with 10% of the set utilized for validation. When the training stopped at the final epoch, training seg loss ema is found to be 0.0718, validation segloss ema is 0.0003, and as a measure to show the success of the "Segmentation Model", the validation dice score is calculated as 0.9163. At the end of the training, it is observed that the learning rate is reduced to 10^{-8} by the scheduler.

The training of "Classifier Model", shown in Figure B.7, is conducted for 50 epochs using the PyTorch library on two RTX 2080 TI GPUs. The batch size is set to 4 without any concern about the capacity of GPUs and the size of the CNN model as well as the input images. ADAM [90] is used as the optimizer, and the learning rate is set to 10^{-3}. The entire S-UAV dataset is used for training, with 10% of the set utilized for validation.

The re-training of "Classification Model", shown in Figure B.7, with the classification ground truth dataset of UAV123 [2], explained in Section 4.3, is performed for 22 epochs using the Pytorch library on two RTX 2080 TI GPUs. The batch size is set

to 4 without any concern about the capacity of GPUs and the size of the CNN model, as well as the input images. ADAM [90] is used as the optimizer, and the learning rate is set to 10^{-4}. The entire UAV123 dataset is used for training, with 10% of the set, utilized for validation. When the training stopped at the final epoch, training class loss ema is 0.3835, validation class loss ema is 0.0066, and the percentage of total successful class predictions during training and validation is 99. This is a satisfactory result.

Hyperparameters, such as the number of epochs, the batch size, and the learning parameters for the optimizers (the learning rate, momentum parameters, scheduler parameters, etc.) for CNN models are decided depending on the training/validation scores and losses, derived at the end of many iterations. The most optimal hyperparameter set is utilized to train CNN models.

5.4.2. Experiments with the First Version of the PFCNNOcc

The first version of the novel PFCNNOcc algorithm, displayed in Figure 5.4 is tested using seven different scenarios from the S-UAV dataset [10], c_a_6_0_g, c_a_2_0_g, a_c_1_0_g, a_a_2_0_g, a_a_14_0_g, a_a_1_0_g, a_a_0_0_g. According to the scenario naming policy, presented in Figure 4.2, the scenarios or videos simulate the environments, explained in the following items.

- Urban areas with heavy traffic and trees.
- Urban areas with industrial facilities.
- Rural areas, populated with forest.
- Sunny weather.
- All visual problems, especially partial and full occlusion.
- Long-term, full occlusion combined with other visual problems.

Tests are run using the PyTorch library on two RTX 2080 TI GPUs, and the speed of the processing is 0.5 fps. Before the iteration starts, there are several parameters that are used during target tracking. First of all, imageID, trackID and catID are set

to enable to start tracking the target or vehicle that is determined before the process initiates. The number of particles, j, tailored by the "Particle Filter Module"is chosen as 25, and the variance of the noise parameters for the process and measurement equations are;

- $q = 4$ for the position variables of x_t in the process equation,
- $r = 1$ for the dimension variables of x_t in the process equation,
- $P = 0.00001$ for z_t, measurement variable in the measurement equation.

The number of particles and the noise parameters are chosen considering the role of the "Particle Filter Module"in the method. "Particle Filter Module"is utilized to fine-tune the pre-estimated location and the size of the target. Therefore, the number of particles, j, is selected as 25, which is enough to sample around the pre-estimated location of the target, and it is a very low quantity compared to the number of particles utilized by the PFCNN, explained in Section 3.1.4.1. The selection of the noise parameters affects the perimeter of the area where the particles are spread, and the similarity between the measured feature and that of the ground truth inside the likelihood calculation. Hence, noise parameters are selected very small to sample the particles in a close proximity of the pre-estimated location of the target for fine-tuning purposes.

Size of ROI, $size_ROI$, is set as 250X250 by default because of the size of images that are used during the training of "Flow Model", "Occlusion Model", "Segmentation Model", and "Classifier Model".

At the beginning of the iteration, two images are read at the same time, hence, the batch size is selected as 1. The number of ROI search areas that are sampled in "Re-Identification Module Full Occlusion"after the "Full Occlusion"state occurs, $m = 8$, to cover or sample the area of uncertainty around the last known position of the target. The selection of the number of ROI, m, is carried out based on the possible duration of the long-term full occlusion in the dataset, and it is guaranteed that the selection is satisfying to sample the area where the target is seen again.

At the end of each iteration, the precision and success scores are calculated to evaluate the performance of the novel method. In the meantime, flow loss, occlusion loss, and segmentation loss are found and plotted using the ground truth data from the S-UAV dataset [10]. The success of the "Classifier Model"is measured by creating the confusion matrix at the end of the scenario. If the "Full Occlusion"state is in progress, no measurement is taken, evaluated, or plotted because of the missing target on the scene.

5.4.3. Experiments with the Final Version of the PFCNNOcc

The final version of the PFCNNOcc method, illustrated in Figure 5.17, is tested with 21 videos or image sequences, selected from the UAV123 10 fps dataset [2]. The selected image sequences are listed in Table 5.1. As realized, the image sequences, including boat and car sequences with very low resolution, recorded from the UAVs that are flying at high altitudes, are excluded. One of the important differences of the UAV123 10 fps dataset, compared to the S-UAV dataset [10] and the UAV123 30 fps dataset, is the magnitude of the motion vector of the cars or targets. One of every three frames is kept; hence, the displacement of the target between the frames becomes equal to three times of the displacement in the UAV123 30 fps. Therefore, the uncertainty in the target motion becomes larger, and the tracking process gets more challenging.

All these image sequences include more than one visual problem at the same time. Four image sequences, car1_2, car1_3, car_2, and car_7, include partial and full occlusion as well as additional visual problems.

Tests are run using the PyTorch library on two RTX 2080 TI GPUs, and the speed of the processing is 0.5 fps. Before the iteration starts, there are a number of parameters that are used during target tracking. The number of particles, j, tailored by "Particle Filter Module"is chosen as 15, and the variance of noise parameters for the process and measurement equations are;

- $q = 2$ for the position variables of x_t in the process equation,
- $r = 1$ for the dimension variables of x_t in the process equation,
- $P = 0.00001$ for z_t, measurement variable in the measurement equation.

The number of particles and the noise parameters are chosen considering the role of the "Particle Filter Module"in the method. "Particle Filter Module"is utilized to fine-tune the pre-estimated location and the size of the target. Therefore, the number of particles, j, is selected as 15, which is enough to sample around the pre-estimated location of the target, and it is a very low quantity compared to the number of particles utilized by the PFCNN, explained in Section 3.1.4.1. The selection of the noise parameters affects the perimeter of the area where the particles are spread, and the similarity between the measured feature and that of the ground truth inside the likelihood calculation. Hence, noise parameters are selected very small to sample the particles in a close proximity of the pre-estimated location of the target for fine-tuning purposes.

Size of ROI, $size_ROI$, is set as 250X250 by default because of the size of images that are used during the training of "Flow Model", "Occlusion Model", "Segmentation Model", and "Classifier Model". During an iteration, two images are read at the same time, hence, the batch size is selected as 1. The number of ROI search areas that are sampled in "Re-Identification Module Full Occlusion Final PFCNNOcc"after the "Full Occlusion"state occurs, $m = 15$, to cover or sample the area of uncertainty around the last known position of the target. The selection of the number of ROI, m, is carried out based on the possible duration of the long-term full occlusion in the dataset, and it is guaranteed that the selection is satisfying to sample the area where the target is seen again after the full occlusion is ended. At the end of each iteration, the precision and success scores are calculated to evaluate the performance of the novel method. If the "Full Occlusion"state is in progress, no measurement is taken, evaluated, or plotted because of the missing target on the scene.

Table 5.1. Selected image sequences from UAV123 10 fps dataset to test the final version of PFCNNOcc.

Name Of Sequence	Start Frame	Stop Frame	Is Tested?	Scale Variation	Aspect Ratio Change	Low Resolution	Fast Motion	Full Occlusion	Partial Occlusion	Out-Of-View	Illumination Variation	Viewpoint Change	Camera Motion	Similar Object
boat1	1	301	NO											
boat2	1	267	NO											
boat3	1	301	NO											
boat4	1	185	NO											
boat5	1	169	NO											
boat6	1	269	NO											
boat7	1	179	NO											
boat8	1	229	NO											
boat9	1	467	NO											
car1_1	1	250	YES											
car1_2	1	292	YES											
car1_3	1	335	YES											
car2	1	441	YES											
car3	1	573	YES											
car4	1	449	YES											
car5	1	249	YES											
car6_1	1	163	YES											
car6_2	1	441	YES											
car6_3	1	383	YES											
car6_4	1	325	YES											
car6_5	1	313	YES											
car7	1	345	YES											
car8_1	1	453	YES											
car8_2	1	407	YES											
car9	1	627	YES											
car10	1	469	YES											
car11	1	113	NO											
car12	1	167	NO											
car13	1	139	NO											
car14	1	443	NO											
car15	1	157	NO											
car16_1	1	139	YES											
car16_2	1	527	YES											
car17	1	353	YES											
car18	1	403	YES											

5.4.4. Experiments for the Ablation Study with the Final Version of the PFCNNOcc

To achieve the ablation study of the final version of the PFCNNOcc, three different versions of the PFCNNOcc are created based on the PFCNNOcc. These versions and their features are explicated in the following items.

- PFCNNOccWOM (PFCNNOcc without Merged Target Detection): This version is created by disintegrating the "Merged Target Detection"function from PFCNNOcc, elaborated in Section 5.3.1.2.
- PFCNNOccWOOccCalc (PFCNNOcc without Occlusion Calculator): This is a version of PFCNNOcc without the "Occlusion Calculator", which is a sub-function of the "Occlusion Detector and Calculator"module, explained in the Section 5.3.1.2. It runs without the calculation of the degreeOfOccAll, and it makes the method to be unaware of the occlusion state.
- UDSCNNOcc (Uniform Dense Sampling CNN Occlusion): This version is formed based on the PFCNNOcc by integrating a uniform dense sampling method, designed in the context of this thesis, in the place of the "Particle Filter Module", shown in Figure 5.13. This method samples 16 bounding boxes on the rough estimate of the target from "Occlusion Detector and Calculator"module. Sampling is achieved by creating 1 pixel space between bounding boxes uniformly along x and y axes in the image coordinates. The width and height of the bounding boxes are sampled in the same manner that "Particle Filter Module"does. The similarity scores for samples are calculated by multiplying the cosine distance between the features of the samples and the reference feature with class probabilities, computed as in Equation (3.14). The features of the samples are extracted from the "Classifier Model", explained in Section 5.2 and Appendix B.

Tailoring the "Re-Identification Module Final PFCNNOcc "and the "Occlusion Detector "sub-function of the "Occlusion Detector and Calculator "module for the ablation study is impossible in terms of their vitality for the target and occlusion

detection purposes. Without these modules, any version of the PFCNNOcc can not run an iteration successfully.

The procedures and the parameters/hyperparameters, tailored during experiments for the ablation study, are the same as the ones used for the PFCNNOcc, explained in Section 5.4.3.

5.5. Discussion

The first version of the PFCNNOcc method, elaborated in Section 5.3.1.1, is tested with the selected videos from the S-UAV dataset [10], [11], and the simulation results are presented and examined in Section 6.3.1. Depending on the precision and success scores, the PFCNNOcc is very robust in estimating the location and the size of the bounding box for the target during tracking. The secrets of the success of the method, lying behind, are elaborated in the following items.

(i) The loss results of the CNN models show that the predictions for the flow and occlusion maps, as well as segmentation masks, have very high precision. The success of these models increases the accuracy in the estimation of the location and the size of the target, as well as in the classification and identification.

(ii) While predicted flow vectors provide the correct displacement of the targets, they contribute to the highly precise estimation of the location of the target with very high kinematics in the next frame of the image sequence. Having an averaged motion vector, calculated by using predicted flow vectors from the previous iterations, provides an opportunity to predict the possible locations of the target correctly under short-term or long-term partial or full occlusion states with varying frequencies.

(iii) A precise segmentation map, predicted on a pixel basis, improves the correctness of the predicted location and, especially, the size of the target on the scene, even under severe scale and aspect ratio variations.

(iv) The forward and backward consistency of the segmented pixels is checked using forward and backward predicted flow vectors, and this process improves the detection process of the occlusion state of the pixels as well as the classification of them as already-occluded, will-be-occluded, and non-occluded. Then, the degree of occlusion is calculated precisely considering the occlusion state of these pixels.

(v) Dense ROI search algorithm of the "Full Occlusion ROI Dense Search Module"contributes to the successful re-detection of the target with very high kinematics after short-term or long-term full occlusion with varying frequencies.

(vi) Classification scores in confusion matrices for cars, estimated by the "Classifier Model", are very high, but are low for the trucks. The reason for the degradation in classification scores is evaluated as the insufficient number of images for the trucks in the S-UAV dataset [10], [11], used to train the "Classifier Model".

The final version of the PFCNNOcc, explained in Section 5.3.1.2, is designed based on the first version of the PFCNNOcc to fill the gap that is comprised of visual problems in the UAV123 dataset [2] that are not observed in the synthetic S-UAV dataset [10], [11]. These visual problems are elaborated as;

- Sudden and large-scale variations,
- Very small vehicles with low resolution as well as very large vehicles whose bounding size is even bigger than the size of ROI (250x250), used to sample the region at the beginning of the iteration,
- Small motion vectors between the sequential images because of the relative motion of the vehicles, depending on the camera motion,
- Merging or the occlusion of the target by the other vehicles during tracking.

To overcome these problems in the final version of the PFCNNOcc, first of all, the "Segmentation Model"and the "Classifier Model"are trained with the ground truth, generated from the UAV123 dataset. Then, the novel functions stated below are evolved or newly designed for the final version of the PFCNNOcc.

- Re-Identification Module Final PFCNNOcc.
- Occlusion Detector and Calculator Module Final PFCNNOcc.
- Full Occlusion Dense Search Module Final PFCNNOcc.
- Merged-Target Condition Detection.

The novel "Re-Identification Module Final PFCNNOcc"evolves the "Re-Identification Module"in terms of pre-filtering the contours as a target candidate, and a revised matching algorithm, based on the physical attributes of the target. "Occlusion Detector and Calculator Module Final PFCNNOcc"is enhanced with the addition of the logic that decides the reason for the pixel reduction of the target during tracking as: 1. Occlusion, 2.Scale variation. The "Full Occlusion Dense Search Module Final PFCNNOcc"is also revised indirectly by the evolution of the "Re-Identification Module Final PFCNNOcc"and the "Full Occlusion Dense Search Module Final PFCNNOcc", and its dense ROI search algorithm is designed to cover a larger area. "Merged Target Detection"function checks if the target is merged or occluded with another vehicle that can be detected by "Segmentation Model", and it is implemented as a novel approach only in the final version of the PFCNNOcc. After the occlusion or merged-target condition ends, the discrimination capability of the function, separating the target from other vehicles on the run, helps the PFCNNOcc to continue tracking precisely.

The final version of the PFCNNOcc is tested with 21 videos from the UAV123 10 fps dataset [2], and the simulation results are presented and examined in Section 6.3.2. As seen from the simulation results, the final version of the PFCNNOcc outperforms all SOTA trackers via the contribution of these novel modules or functions in terms of precision and success scores against all visual problems, especially short and long-term partial and full occlusion, coupled with a variety of others. Even the Siamese Network-based CNN methods are not able to achieve such high tracking scores.

An ablation study is carried out by testing and comparing the performance of the PFCNNOcc and its ablation versions, namely PFCNNOccWOM, PFCNNOccWOOccCalc, and UDSCNNOcc. Depending on the performance results, elaborated in Sec-

tion 6.3.3, “Occlusion Calculator” and “Merged Target Detection” functions evolve the tracking performance of the PFCNNOcc step by step. PFCNNOccWOOccCalc shows the lowest tracking performance because of the lack of the “Occlusion Calculator” function, which is responsible for calculating the degree of overall occlusion during tracking. Without updating the degree of overall occlusion, the PFCNNOccWOOccCalc is unaware of the occlusion transitions. When the target is under full occlusion, the PFCNNOccWOOccCalc is not able to switch to the “Full Occlusion Dense Search Module Final PFCNNOcc” function to search for the occluded target. The integration of the “Occlusion Calculator” function boosts the precision and success scores of the PFCNNOcc significantly.

The importance of the “Merged Target Detection” function is easily discerned from the performance differences between the PFCNNOcc and the PFCNNOccWOM. The PFCNNOcc gains a very vital performance evolution that drives it to the top of the SOTA with the invention of the “Merged Target Detection” function.

While the PFCNNOcc outperforms the UDSCNNOcc and the SOTA trackers by the integration of the “Particle Filter” module to fine-tune the rough estimate of the target state variables in terms of the overall precision score, the uniform dense search function of the UDSCNNOcc provides a slightly higher success score compared to that of the PFCNNOcc. These results prove that while the PFCNNOcc provides more precise location estimates of the bounding boxes for the target, the UDSCNNOcc has better estimates for the size of the target, depending on the success results. It is thought that the performance difference between the UDSCNNOcc and the PFCNNOcc in terms of the success scores occurs because of the insufficient number of particles, tailored by the PFCNNOcc to sample the region around the rough estimate location of the target. The performance of the PFCNNOcc may increase if the number of particles is increased. Modeling the process equation with a non-linear model instead of a random walk model can boost the performance of the PFCNNOcc advantageously.

5.6. Conclusion

The PFCNNOcc is designed to achieve a robust target tracking in low-altitude flying UAVs against all visual problems, especially against short or long-term partial and full occlusion coupled with aspect ratio change, background clutter, camera motion, fast motion, illumination variation, low resolution, out-view, similar object, scale variation, viewpoint change. The PFCNNOcc combines CNN models, trained with the novel synthetic S-UAV dataset [10], [11], and the UAV123 dataset [2] to predict the optical flow and occlusion maps, segmentation masks and classification scores, and the novel functions, named as;

- Re-Identification Module Final PFCNNOcc,
- Occlusion Detector and Calculator Module Final PFCNNOcc,
- Full Occlusion Dense Search Module Final PFCNNOcc,
- Merged Target Detection,

and these are the contributions of the PFCNNOcc to the literature. As far as is known that the PFCNNOcc is the first method in literature, predicting and utilizing the optical flow and segmentation to estimate the state variables of the target as well as the occlusion state and the degree of it during tracking at UAV videos. The crucial functions that pave the way to success are summarized in the following items.

(i) While predicted flow vectors provide the correct displacement of the targets, they contribute to the highly precise estimation of the location of the target with very high kinematics under short-term or long-term partial or full occlusion states with varying frequencies.

(ii) A precise segmentation map, predicted on a pixel basis, improves the correctness of the predicted location and, especially, the size of the target on the scene, even under severe scale and aspect ratio variations.

(iii) The forward and backward consistency check using forward and backward predicted flow vectors improves the precise detection of the occlusion and the clas-

sification of pixels as already-occluded, will-be-occluded, and non-occluded.

(iv) Dense ROI search algorithm contributes to the successful re-detection of the target with very high kinematics after short-term or long-term full occlusions with varying frequencies.

(v) Classification scores in confusion matrices for cars are very high, but are low for the trucks. The reason for the degradation in classification scores is evaluated as the insufficient number of images for the trucks in the S-UAV dataset [10], [11], used to train the "Classifier Model".

PFCNNOcc outperforms all SOTA trackers in terms of precision and success scores against all visual problems, faced in 21 scenarios, summarized in Table 5.1, and even the Siamese Network-based CNN methods are not able to achieve such high tracking scores. The results of the ablation study also prove that "Occlusion Calculator" and "Merged Target Detection" functions evolve the tracking performance of the PFCNNOcc step by step to the top of the SOTA trackers.

For future work, the PFCNNOcc will be evolved to track multiple targets simultaneously at the UAV videos, and it is considered that having the target location, size, and motion vector information of all targets at the same time would ease the occlusion problem via solving the relative motion problem. Another future study may be to increase the number of samples and their variety in the S-UAV dataset to adapt it to the real-time UAV scenarios in terms of the resolution in size and the visibility of the target and the magnitude of the optical vectors of the vehicles.

6. SIMULATION RESULTS AND DISCUSSIONS

6.1. Simulation Results of PFCNN

The PFCNN is tested with the image sequences, labelled "YES"in Table 3.1, from the UAV123 30fps dataset [2], and the experimental setup, utilized during tests, is elaborated in Section 3.1.4. As demonstrated in the following Sections, namely, 6.1.1, 6.1.2, and 6.1.3, the PFCNN's precision and success results are plotted based on three distinct approaches: overall comparison, attribute-based comparison, and scenario-based comparison.

The tracking results on boat and car videos or situations in the UAV 123 dataset are used for attribute-based and scenario-based comparisons. Based on the precision and success metrics, attribute-based results present the analysis of how well a method tracks a target under visual problems, such as scale change, aspect ratio variation, low resolution, fast motion, out-of-view, viewpoint change, illumination variation, camera motion, and similar objects.

The robustness and weaknesses of a tracking approach can be assessed for a particular visual problem, and the flaws can be improved while focusing on these particular problems. The purpose of scenario-based comparison is to demonstrate the resilience of a tracking method to one or more visual problems in a scenario simultaneously, because a video may include a complicated setting of multiple visual issues rather than just one. Even though the solo performance of SOTA methods against each visual challenge is highly distinguishable, the combination of multiple visual challenges can occasionally drag the performance of these methods.

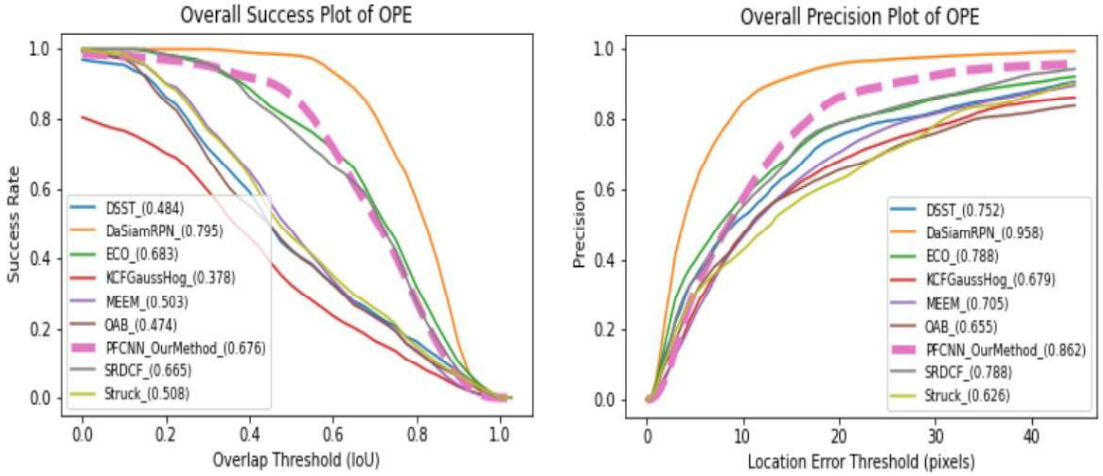


Figure 6.1. Overall precision and success plots of the SOTA trackers and the PFCNN.

6.1.1. Overall Comparison

Figure 6.1 displays the overall precision and success performances of the PFCNN [6] in comparison to the SOTA techniques. After DaSiamRPN [38], the PFCNN has the second-best overall precision and success scores [38]. It outperforms the best SOTA trackers like DSST [20], SRDCF [21], MEEM [24], and STRUCK [22, 23] in [4, 12, 13]. In particular, the overall success score of the PFCNN is nearly identical to that of the ECO [28].

The top-performing overall precision and success outcomes for the situations in Table 3.1 are a result of the innovative scale estimation system, the reference model calculator mechanism, the robust sampling approach, and the precise visual features of the PFCNN. The PFCNN produces extremely promising results for the uncommon partial occlusion and out-of-view sequences like car3, *car*8_1, and car10, even though not built for scenarios that include occlusion and out-of-view visual difficulties. Furthermore, the PFCNN demonstrated remarkable resilience to changes in scale, aspect ratio, fast motion, camera motion, viewpoint, and similar objects on the scene.

6.1.2. Attribute-Based Comparison

Figure 6.2 displays the attribute-based precision comparison between the PFCNN and the SOTA trackers. Table 3.1 displays the plots for each attribute, which are computed based on the scenario-attribute relation.

The PFCNN surpasses the other top-performing SOTA trackers in [4], [12,13] and has the second-best precision score after DaSiamRPN [38] for scenarios that include scale change and aspect ratio variation. Owing to the particle rescaling and unification modules, which estimate the scale along both axes, the PFCNN is very successful. With the best-performing SOTA trackers, the PFCNN attains nearly full precision scores for low-resolution scenarios.

For fast motion sequences, the PFCNN surpasses the other top-performing SOTA trackers in [4], [12, 13] by a significant margin and attains the second-best precision score behind DaSiamRPN [38]. The particle filter in PFCNN uses a random walk model with a large uncertainty sampling region to improve the estimation scores under nonlinear motion. Thus, PFCNN outperforms all the top-performing SOTA trackers, including ECO [28], [20] in [4], [12, 13], and is the second-best performing precision score for out-of-view, camera motion, viewpoint-change, and similar object attributes, after DaSiamRPN [38]. It should be mentioned that a very strong reference feature modeling and a CNN with proper training are essential for tracking against attributes like fast motion, viewpoint changes, and camera motion. A well-trained CNN and extremely strong reference feature modeling are essential for high-accuracy tracking against these four visual problems. The innovative unification module is in charge of updating the reference model for the parts of the vehicle at each iteration depending on how much each part represents its own class. Overall, the PFCNN outperforms the SOTA trackers in terms of precision scores, achieving the top two results against 80 % of the visual challenges.

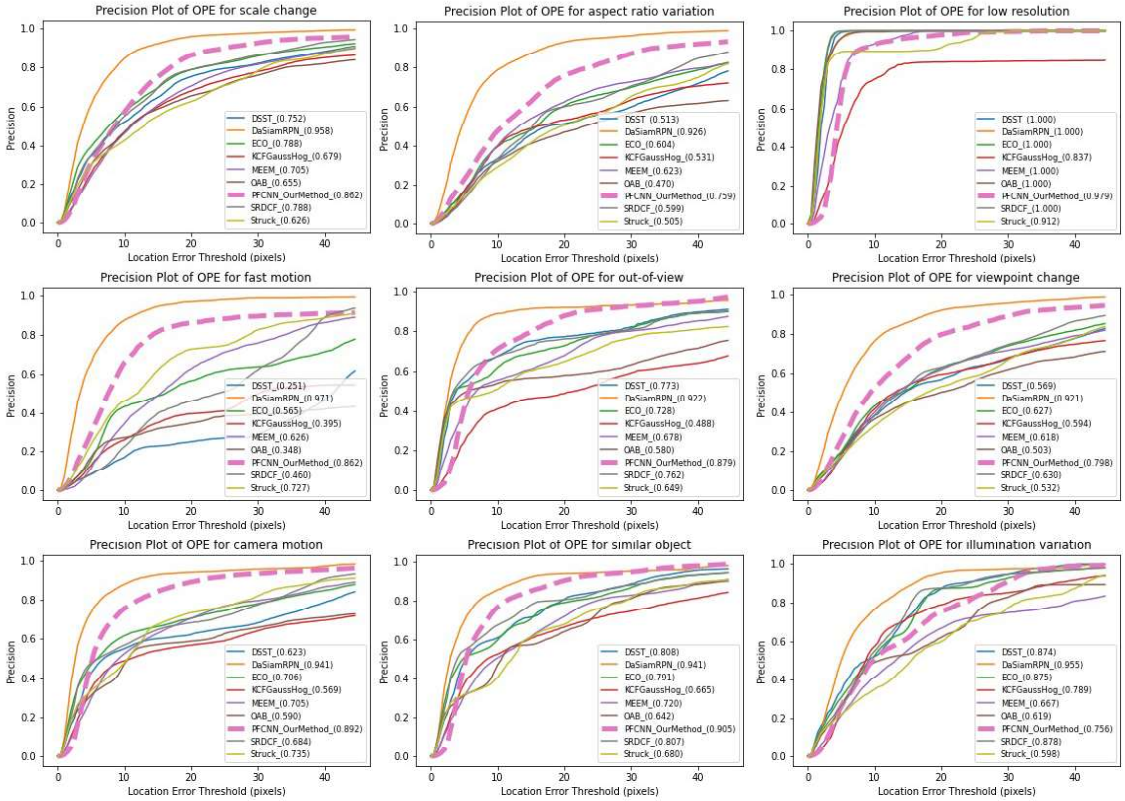


Figure 6.2. Attribute-based precision plots of the SOTA trackers and the PFCNN.

Figure 6.3 displays the attribute-based success comparison between the PFCNN and the SOTA trackers. Table 3.1 displays the plots for each attribute, which are computed based on the scenario-attribute relation. Following DaSiamRPN [38], the PFCNN attains the top two best success scores for the scenarios or videos, including scale change and aspect ratio variation as justified in the above paragraphs.

The PFCNN surpasses the other top-performing SOTA trackers in [4], [12,13] by a significant margin, even though it achieves a very close success score to ECO [28] under scale change. The PFCNN, which estimates the scale along both axes for adaptation to extreme scale shift, aspect ratio fluctuation, rotation, and fast motion, is supplied by the particle rescaling and unification modules. For fast motion sequences, the PFCNN outperforms the other top-performing SOTA trackers in [4], [12, 13] by a significant margin and earns the second-best success score behind DaSiamRPN [38]. This is due to the particle filtering with parts-based tracking mentality that is capable of tracking

a target under highly non-linear motions of the target. PFCNN demonstrates its resilience against the most advanced trackers in [4], [12, 13]. It achieves the top-3 best score when it comes to illumination variation problems, but it produces the top-4 best success scores when it comes to scenarios with similar objects and out-of-view problems. For 60% of the visual difficulties, the PFCNN achieves the top-2 best success scores, and for 70% of them, it performs the top-3 best scores.

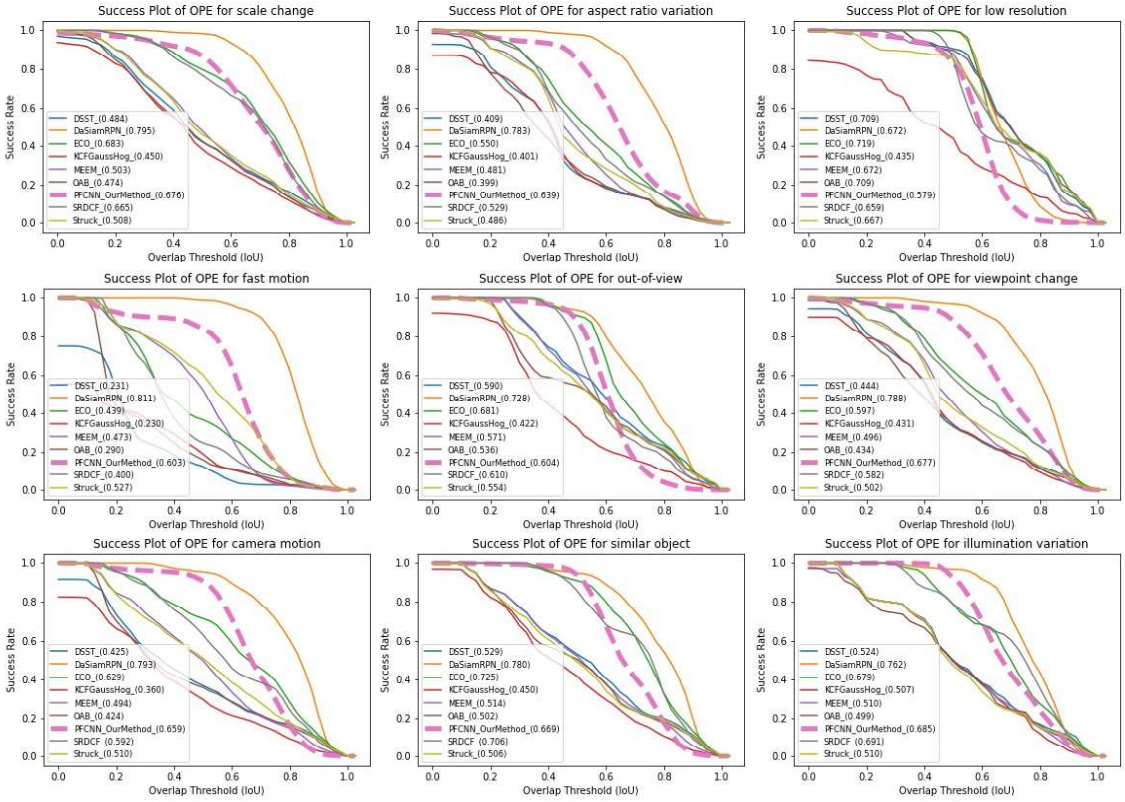


Figure 6.3. Attribute-based success plots of the SOTA trackers and the PFCNN.

All in all, thanks to its scale estimation scheme at the particle rescaling and unification modules, the reference model calculator mechanism of the unification module, the robust sampling strategy of the particle filter, and the precise visual features provided by the CNN model, the PFCNN outperforms the best-performing trackers in the literature by achieving the top-2 best precision and success scores at 70 % of the attributes and the top-3 best scores at 75 % of them.

6.1.3. Scenario-Based Comparison

After downloading the tracking results of the SOTA trackers from the working group websites [4], [12], the results of the PFCNN are compared. For the 17 instances that are marked with 'YES' in Table 3.1, the accuracy and success rates of every approach are displayed in Figures 6.4-6.7.

According to the precision findings in Figures 6.4 and 6.5, the PFCNN provides better results for the boat1, boat3, car6-1, and car6-4 than the finest SOTA trackers, including DaSiamRPN [38], ECO [28], and SRDCF [21]. These situations include changes in scale, aspect ratio, perspective, camera movements, and other related issues. A significant variance in the target visual model is also observed in these videos, demonstrating the PFCNN's resilience to these visual issues. PFCNN gets a precision score between the top 5 performing trackers with high precision scores for boat2, boat5, car1-1, car6-3, car3, car5, car8-2, car10, and car16-2.

Particularly, the visual sequences for car 6-3, car 8-2, and car 16-2 exhibit extreme size shifts, aspect ratio variations, and rotations exceeding 90 degrees, in addition to camera movements and shifts in viewpoint. Very high non-linear motion changes and abrupt, big moves are also features of car16-2. Together with the proposed particle rescaling (Section 3.1.3.2) and unification modules (Section 3.1.3.5), the particle filter scheme elevates the performance of the PFCNN to the top of the list of the best trackers in the literature [4], [12, 13].

Only a few trackers are able to obtain high precision scores for the car8-1 and car18 picture sequences, which are among the most difficult movies. With the exception of extreme occlusion, car8-1 exhibits all visual issues, including low resolution and fast motion. The PFCNN surpasses the other top SOTA trackers by a significant margin and receives the second-highest score behind DaSiamRPN [38]. Only a few number of trackers can even maintain high-precision target tracking in car18 due to extreme scale change, aspect ratio variation, camera motion, view-point change, and fast motion.

PFCNN beats other CNN-based trackers like ECO [28], SRDCF [21], and others, and is the second-best tracker after DaSiamRPN [38] with nearly the same score. The PFCNN achieves substantially higher precision scores than the top-performing trackers in [12], including DSST [20], SRDCF [21], and MEEM [24]. Due to the lack of comparable target samples in the training set, the PFCNN receives mediocre results for the boat4 and boat6 sequences.

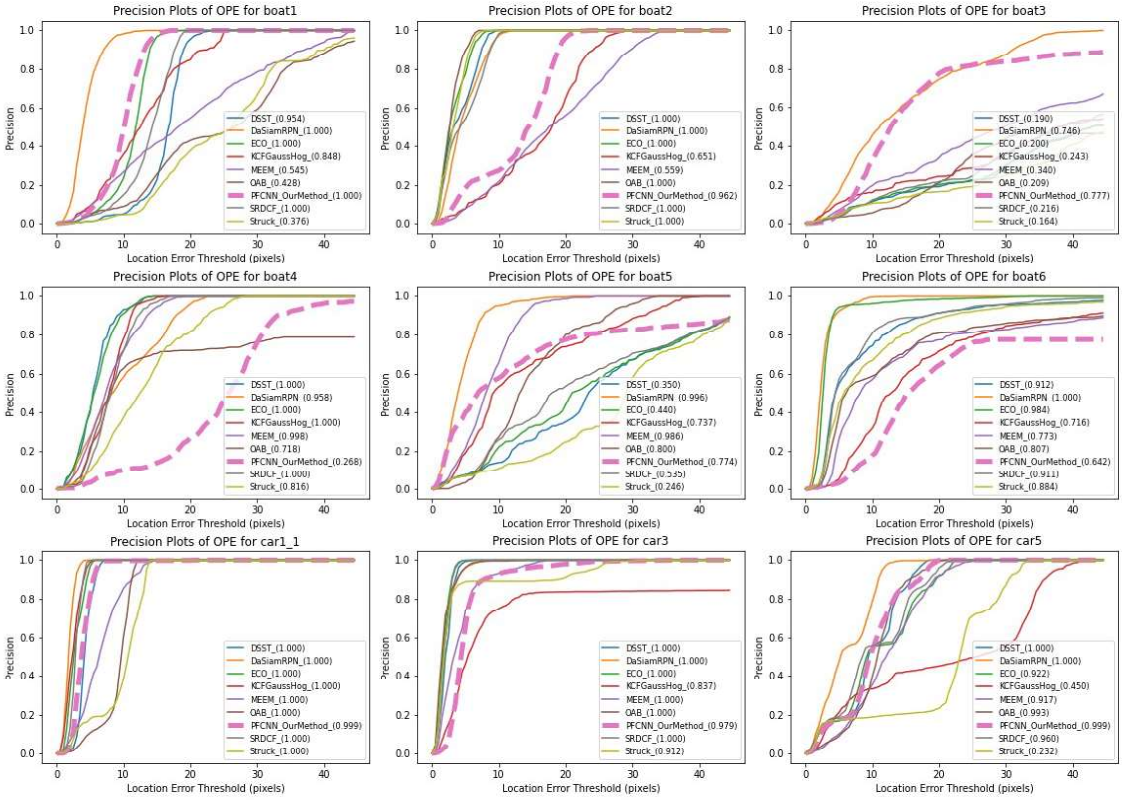


Figure 6.4. Scenario-based precision plots of the SOTA trackers and the PFCNN.

Therefore, when compared to the best SOTA trackers in [4], [12,13], the PFCNN earns the highest or top-5 performing precision scores for 90 % of very tough image sequences. The PFCNN outperforms the best SOTA trackers, including DaSiamRPN [38], ECO [28], SRDCF [21], and DSST [20], and generates the best success scores for boat-2 and car6-1, according to the success results in Figures 6.6 and 6.7.

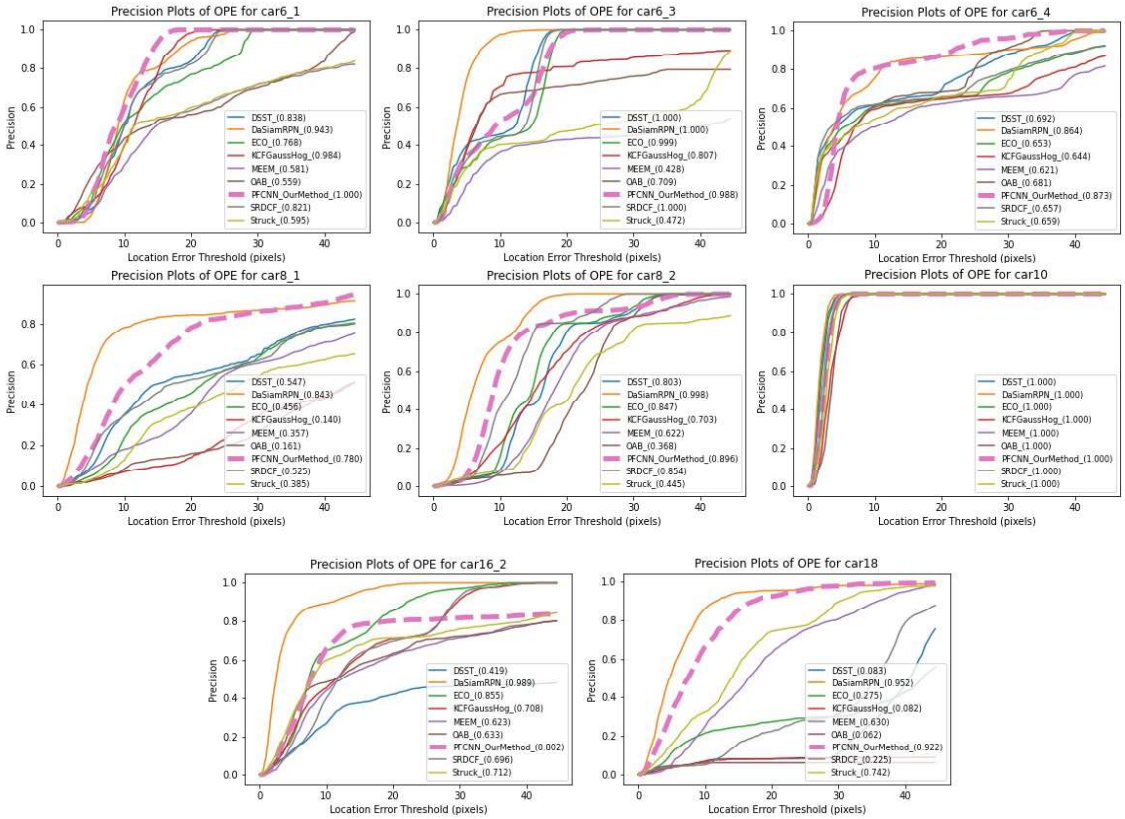


Figure 6.5. Scenario-based precision plots of the SOTA trackers and the PFCNN (cont.).

A significant scale shift is seen in these sequences, and the unification module estimates the scale parameter, sP, quite effectively because it uses its own scale estimator to assess the rescaled samples of the united vehicle. .The PFCNN outperforms the other top-performing state-of-the-art trackers in [4], [12, 13] and earns the top-2 best success scores after DaSiamRPN [38] for boat1, boat3, boat5, car5, car6-3, car6-4, car16-2, and car18. The present visual model is severely altered by extremely difficult visual problems, such as scale change, aspect ratio variation, rotation combined with fast motion, camera motion, and viewpoint changes, particularly during the boat3, boat5, car6-4, car16-2, and car-18 sequences. The PFCNN has a very high success rate and can track the cars reliably. Except the DaSiamRPN [38], it performs better than all SOTA trackers. Due to the lack of comparable target samples in the training set, the PFCNN attains poor success score results for the boat6 sequence.

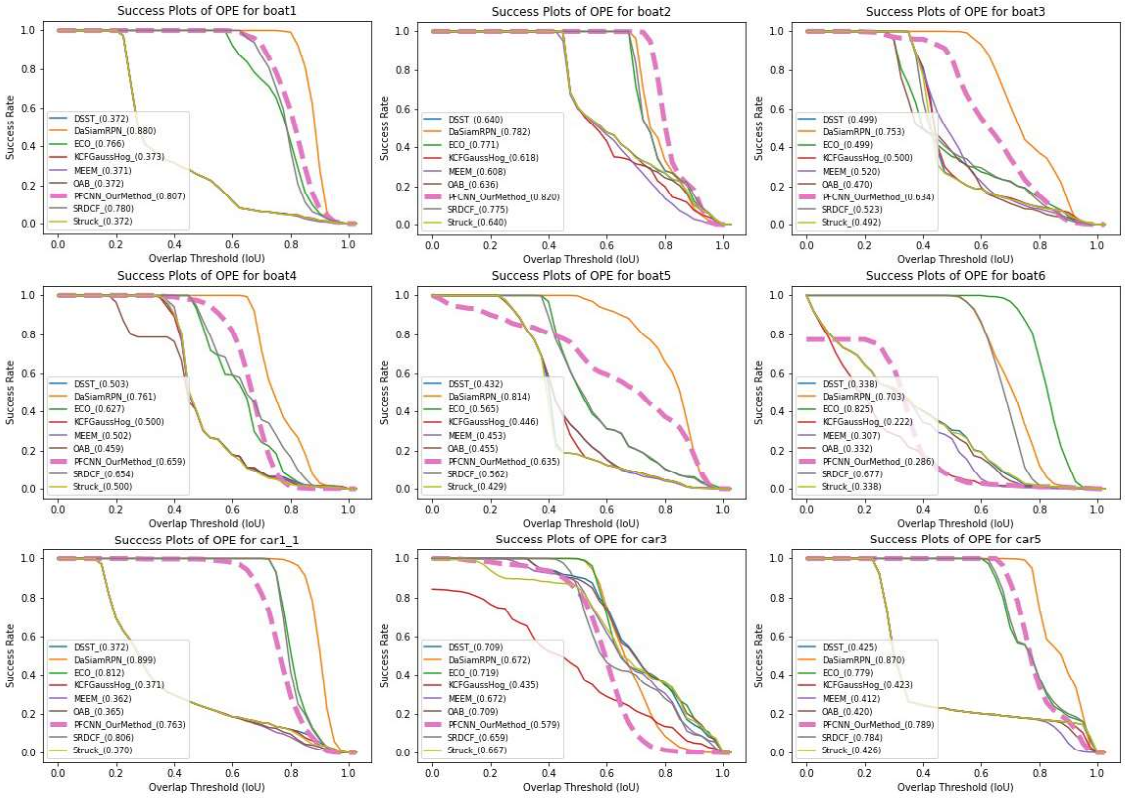


Figure 6.6. Scenario-based success plots of the SOTA trackers and the PFCNN.

Additionally, when visual issues such as scale change, aspect ratio variation, extremely moderate partial occlusion, out-of-view, camera motion, view-point shift, and similar items are present, the PFCNN performs well in the top three for car8-1. The PFCNN displays moderate success scores for boat6, car1-1, car3, car8-2, and car10, with effective tracking from start to finish of the video. The majority of the trackers attain good success scores, and these videos don't contain any particularly difficult visual issues.

According to these findings, when compared to the top SOTA trackers in [4], [12, 13], the PFCNN gets the highest or at least top-4 performance success ratings for 80 % of the very tough cases.

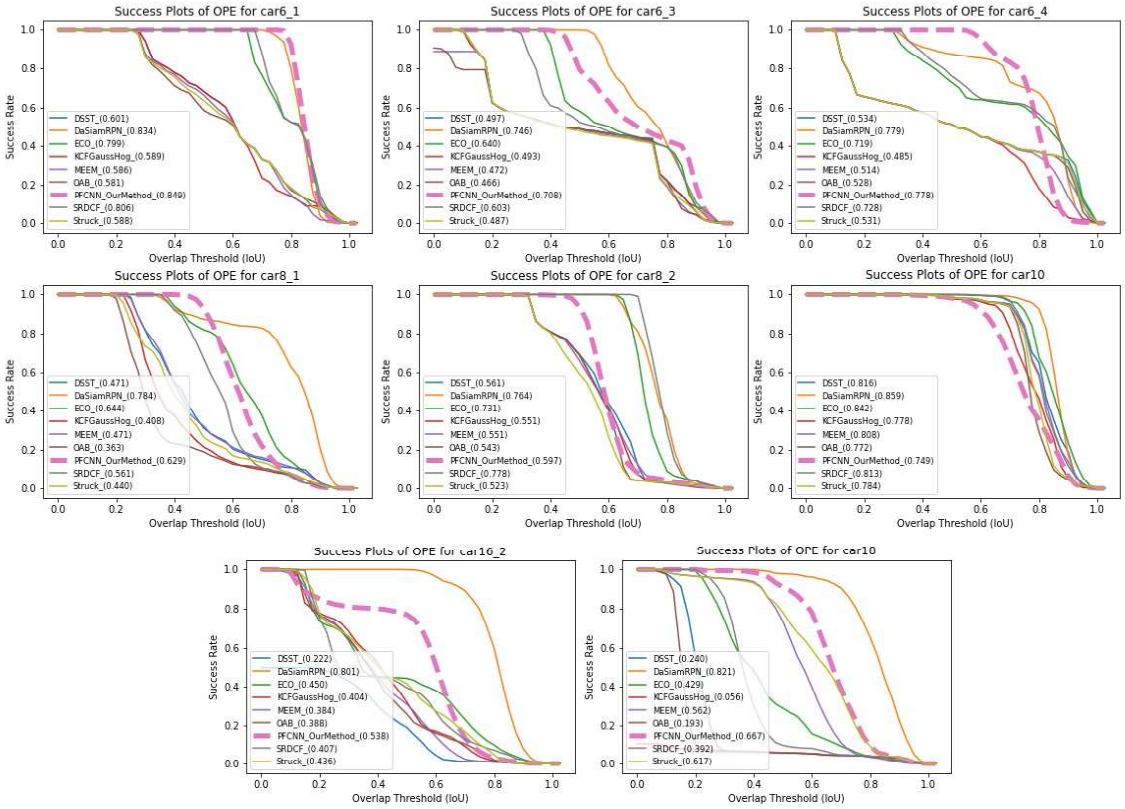


Figure 6.7. Scenario-based success plots of the SOTA trackers and the PFCNN (cont.).

In fact, an object can be tracked while being divided into $N \geq 2$ parts using the proposed method. However, the selection of $N = 2$ was made for two key reasons: The first is the ease with which the tracking result can be presented following the unification of the two parts. Second is the resolution of the objects in the UAV 123 dataset. The dataset has low resolution and small object or vehicle sizes. Thus, the attributes of the object become useless for tracking and training if it is divided into more than two parts. The resolution of the objects permits tracking with only two parts, even though the proposed method is convenient to train and test with multi-part processing. In [97], the video tracking results for each scenario using the PFCNN at a speed of 30 frames per second are shared. The estimates are shown by the blue bounding boxes, while the ground truth is shown by the green ones. Videos are annotated with the abbreviations "GT" and "Est" for ground truth and estimated, respectively, to highlight the salient characteristics of these annotations. To determine which frame is being processed at

any given moment, videos also carry the number of frames.

Another important feature is the visual problem or problems that are observed in videos. The annotations for scale change (SC), aspect ratio variation (ARV), low resolution (LR), fast motion (FM), partial occlusion (PO), out-of-view (OOV), illumination variation (IV), viewpoint change (VC), camera motion (CM), similar object (SO) are added into video with their abbreviations in red color to increase the situational awareness of the viewer.

6.2. Simulation Results of Auxiliary Particle Filter with CNN

To show the tracking performance of APF and the degree of analogy between APF and PFCNN, both methods are compared based on their corresponding precision and success scores, and these scores were constructed from three different points of view: 1. Overall, 2. Attribute-Based, and 3. Scenario-Based, and the details are given in the following subsections.

6.2.1. Overall Comparison

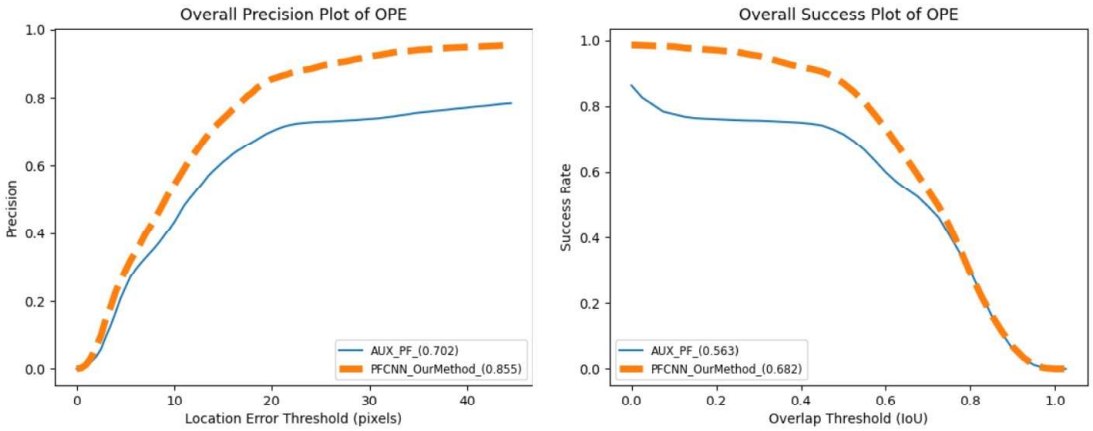


Figure 6.8. Overall precision and success Plots: PFCNN versus APF.

Overall precision and success scores for both methods, derived via the combination of the precision and success scores of sixteen videos, were shown in Figure 6.8.

Based on the results, PFCNN [6] outperforms APF in terms of both precision and success scores. These results can be evaluated as the average performance of methods against all visual problems faced during all videos. Hence, it can be concluded that while PFCNN and APF mimic the sampling scheme of the particle filter that uses the optimal importance density, PFCNN outperforms APF, when used for target tracking under challenging scenarios, and when target and camera motion are non-linear with high dynamics.

6.2.2. Attribute-Based Comparison

During the analysis, performance differences for a total of nine visual challenges were identified. These visual challenges can be listed as follows: aspect ratio variation, camera motion, fast motion, illumination variation, low resolution, out-of-view objects, scale change, similar objects, and viewpoint change.

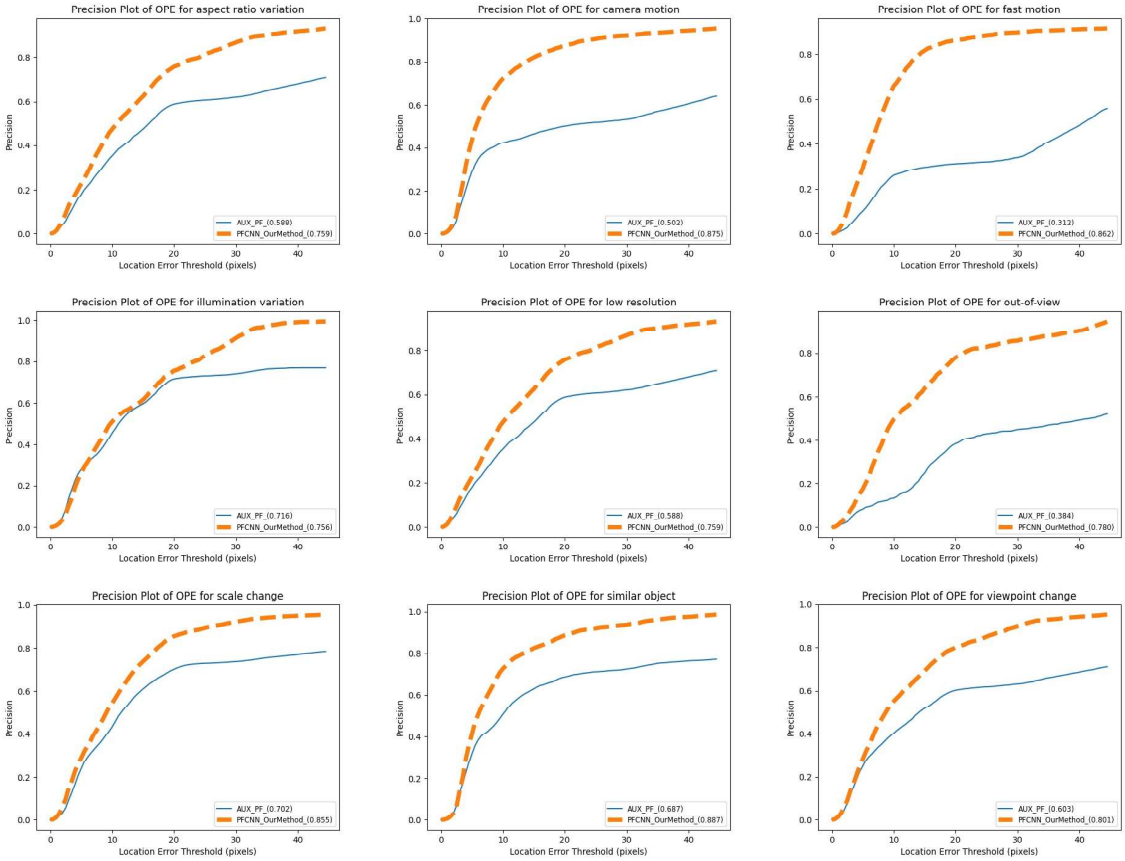


Figure 6.9. Attribute-Based Precision Plots PFCNN versus APF.

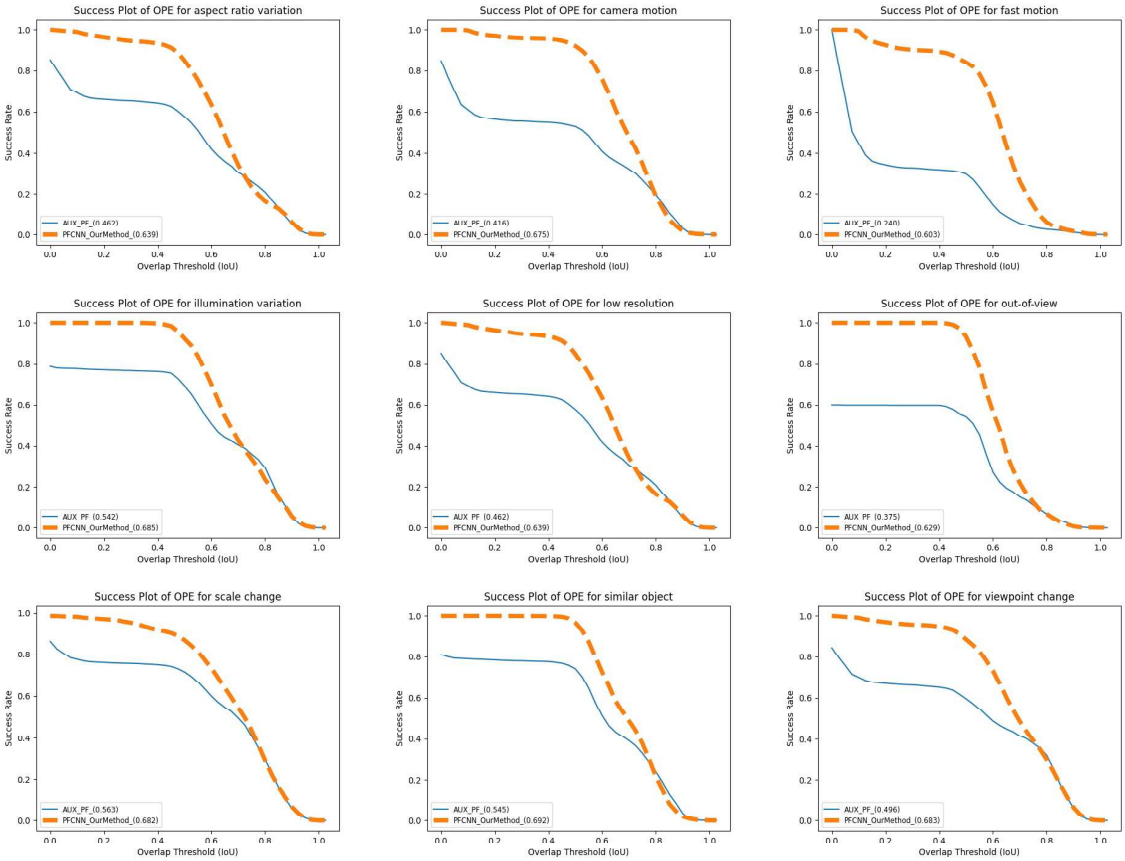


Figure 6.10. Attribute-based success plots PFCNN versus APF.

The precision scores, obtained, are presented in Figure 6.9, and the success scores are shown in Figure 6.10. When both precision and success scores are considered, it is easy to see that PFCNN method gives more robust or successful results compared to APF for all visual problems. In particular, significant performance differences were observed against issues such as fast motion, camera movement, and out-of-view problems, and it is determined that the PFCNN method is able to achieve target tracking with much higher performance than APF.

The primary reason for the significant performance difference is believed to be that PFCNN is more robust than APF in response to nonlinear, dynamic target or camera movements. This performance discrepancy is thought to stem from the parent particle sampling logic, named as “Presampling”, which is explained in detail in Section 3.2.2.1, as well as the likelihood support for belonging to the reference class used in the PFCNN method.

Specifically, as indicated in Equation3.35, after the auxiliary or parent particles are sampled with low-variance noise, their weights are updated based on the obtained observation. Then, resampling is performed according to these updated weights, and the target prediction is made in the second stage. During this process, the updated auxiliary particle weights cause the main particles to be sampled in a region of low uncertainty around the observation. This results in successful target tracking in environments where the target movement is linear, the target motion is low, and the observation model does not change significantly. On the other hand, in situations with high target or camera movement, the main particles increase their distance from the reference observation, significantly reducing the tracking performance.

In summary, based on the performance results and conclusions drawn from this section, and considering Equations 3.18, 3.38, 3.39, the PFCNN method [6] demonstrates superior tracking performance in all visual problems compared to APF, especially in videos with high dynamic motion. This superiority is due to the class probabilities it includes into the weight update calculation while mimicking the best importance sampling steps.

6.2.3. Scenario-Based Comparison

This section presents evaluations of the precision and success scores obtained for each scenario, as shown in Figures 6.11, 6.12, 6.13, and 6.14. When examining the graphics, it can be observed that both methods delivered very close, even identical results in scenarios where linear motion was high and visual model change was low. However, in scenarios with nonlinear, high dynamic motion and significant visual model change (boat6, car8-1, car16-2, car18), PFCNN [6] demonstrated a significantly higher performance compared to the APF. The reason for the performance difference is explained in detail in Section 6.2.2.

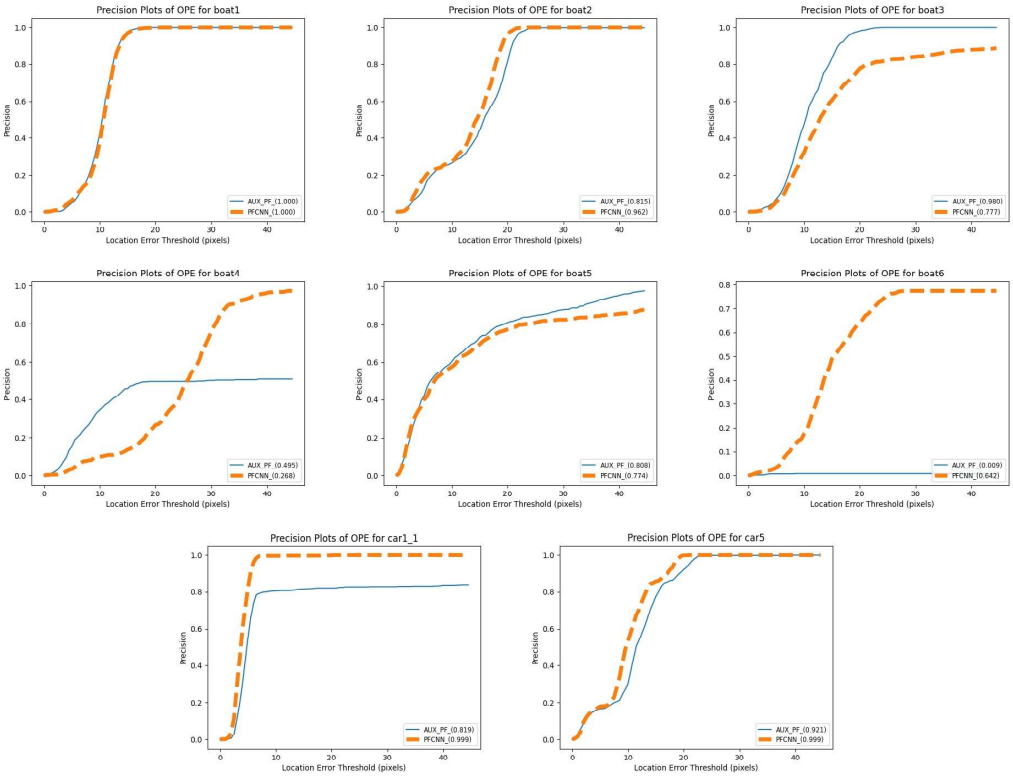


Figure 6.11. Scenario-based precision plots PFCNN versus APF.

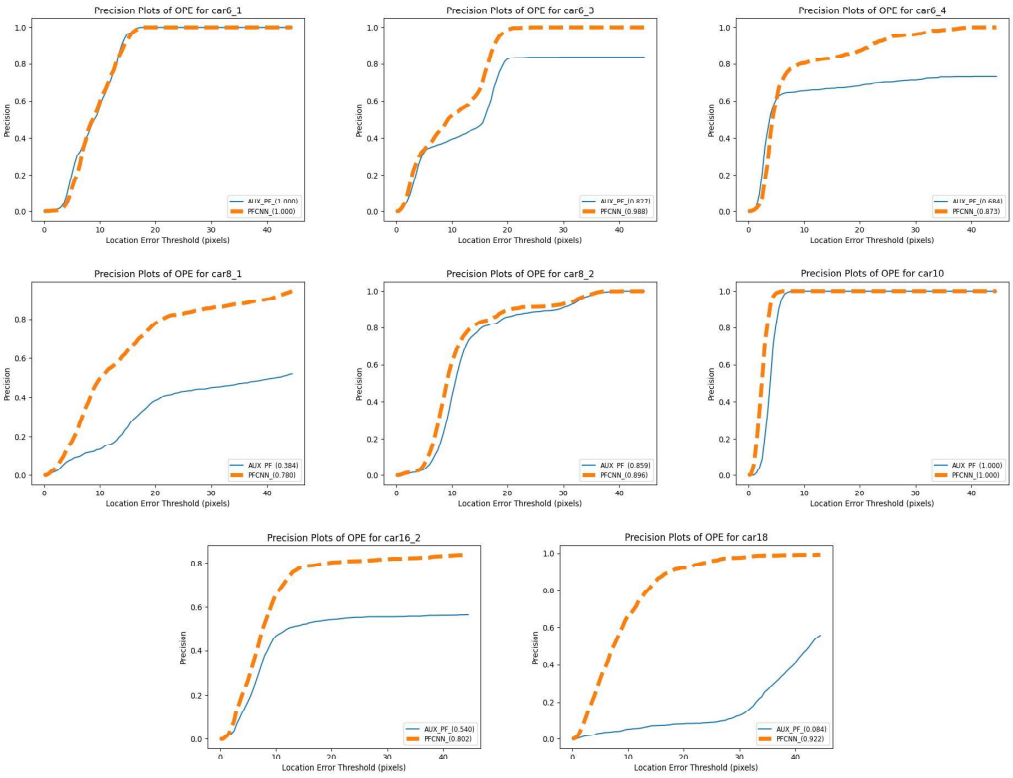


Figure 6.12. Scenario-based precision plots PFCNN versus APF (cont.).

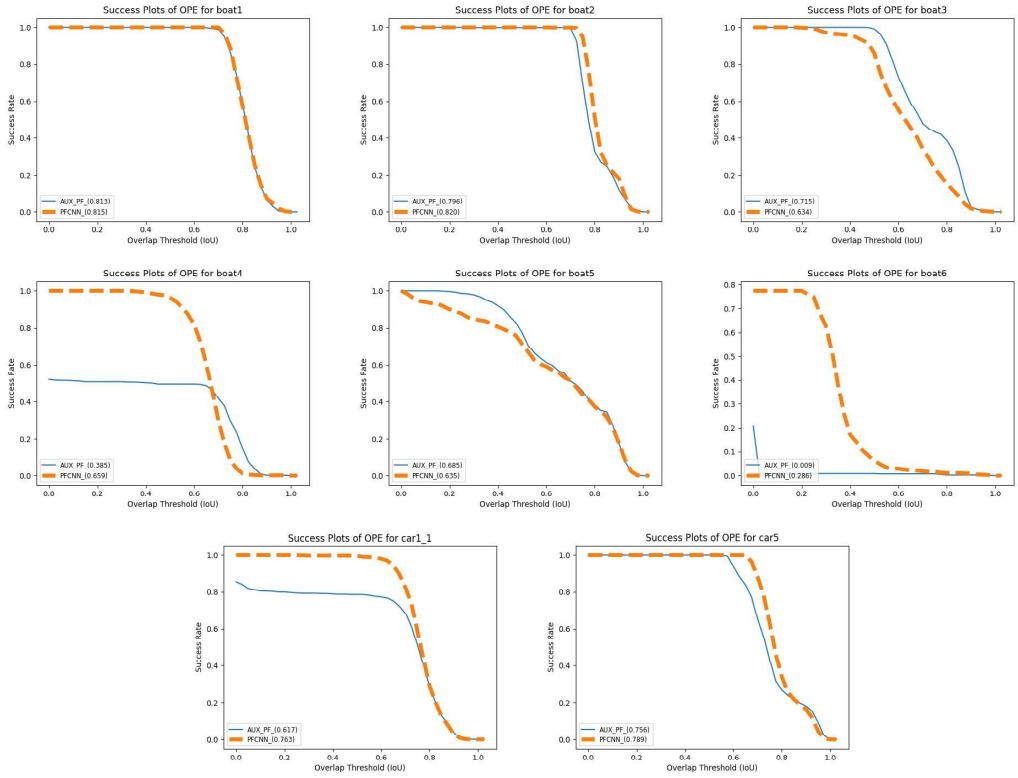


Figure 6.13. Scenario-based success plots PFCNN versus APF.

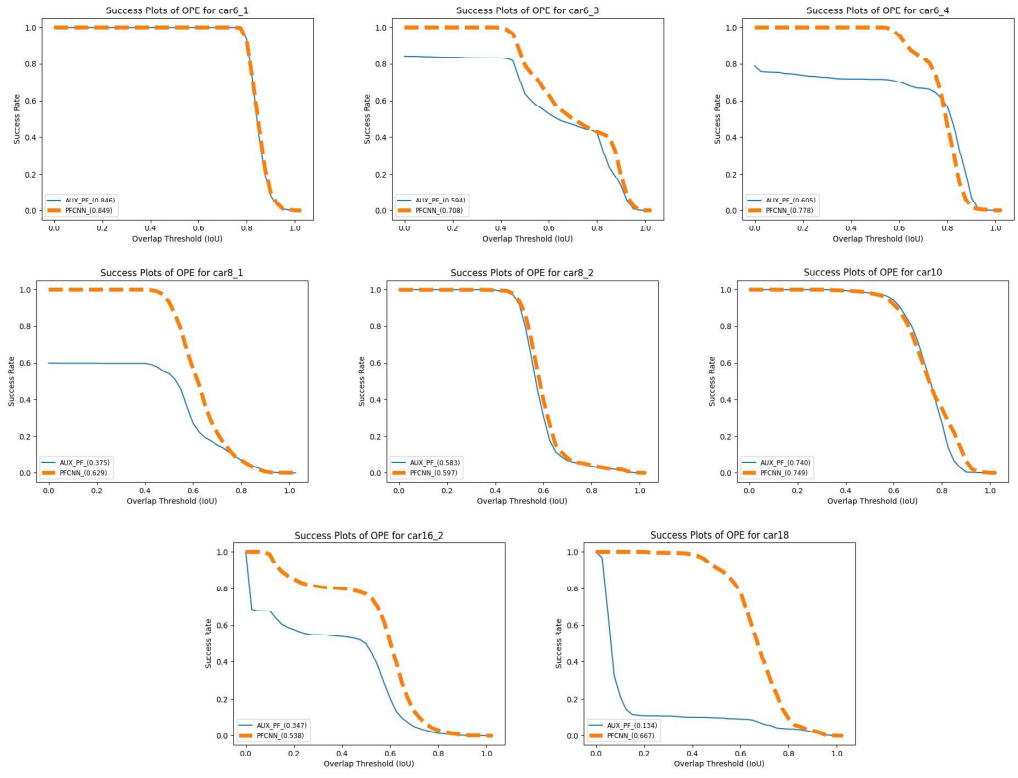


Figure 6.14. Scenario-based success plots PFCNN versus APF (cont.).

6.3. Simulation Results of PFCNNOcc

6.3.1. Simulation Results of the First Version of the Tracking Model

Simulation results of the first version of the novel tracking model, proposed in this thesis, shown in Figure 5.4, are evaluated based on the precision, success, and classification scores as well as flow, occlusion, and segmentation losses. Moreover, degreeOfOccRel, degreeOfOccAll, and their average during tracking are calculated and plotted. As elaborated in Section 5.4.2, seven different scenarios from S-UAV dataset [10], c_a_6_0_g, c_a_2_0_g, a_c_1_0_g, a_a_2_0_g, a_a_14_0_g, a_a_1_0_g, a_a_0_0_g, are selected to test the tracking method. These scenarios are the most challenging videos, including short and long-term partial and full occlusion with a varying frequency, and they simulate urban and rural areas with different kinds of objects, buildings, trees, bridges, and traffic lights, to provide the occlusion condition. For each scenario, a snapshot of an image, annotated with the results derived at each frame during the tracking, is presented with the simulation results. The details of the information, annotated on the image, are presented in the following items.

- The information, shown on the top left side of the image before the tracking starts: The number of iterations (iter), waiting for image ID (the image ID that the tracking starts), waiting for category ID (the category ID of the target that will be tracked), waiting for track ID (the track ID of the target that will be tracked), the number of the targets, seen on the image at that moment (# of targets on the scene).
- The information, shown on the top left side of the image: The number of iterations (iter), degreeOfOccRel (degOfOccRel), degreeOfOccAll (degOfOccAll), reference class ID (refClassID), forward flow along the x and y axes, flow_f_x and flow_f_y.
- The information, shown on the top right side of the image: Success rate or score (SuccR), precision rate or score (PrecR), flow loss (Flow Loss), occlusion loss (Occ Loss), segmentation loss (Seg Loss), and the number of visible contours inside the ROI search are that satisfy the contourFlowCheck conditions, # of visCnts.

- Images, displayed on the bottom side of the image (From left to right): *predicted_flow_f* (flow_f), ground truth for flow_f (GT_flow_f), *predicted_occ_f* (occ_f), ground truth for occ_f (GT_occ_f), im_{t-1}*predicted_mask* (seg_1), ground truth for im_{t-1}*mask* (GT_seg_1), im_{t-1}*occlusion_map* (im1_occ_map).
- Rectangles/squares and annotations around the target: Green rectangle/square (ground truth for the bounding box), blue rectangle (estimated/predicted bounding box), estimated class ID name (blue), and class score (blue).

The scenario-based simulation results are shown and explained in the following paragraphs. The calculation of the results starts at the initial frame or image of the video when the tracked target shows up, and the calculation is stopped when the target is out of frame or the video is finished. Hence, the first iteration, shown in the plots for the results, is the initial frame in which the target appears, and the tracking starts. And the videos, showing the simulation results during tracking, are presented in the [98].

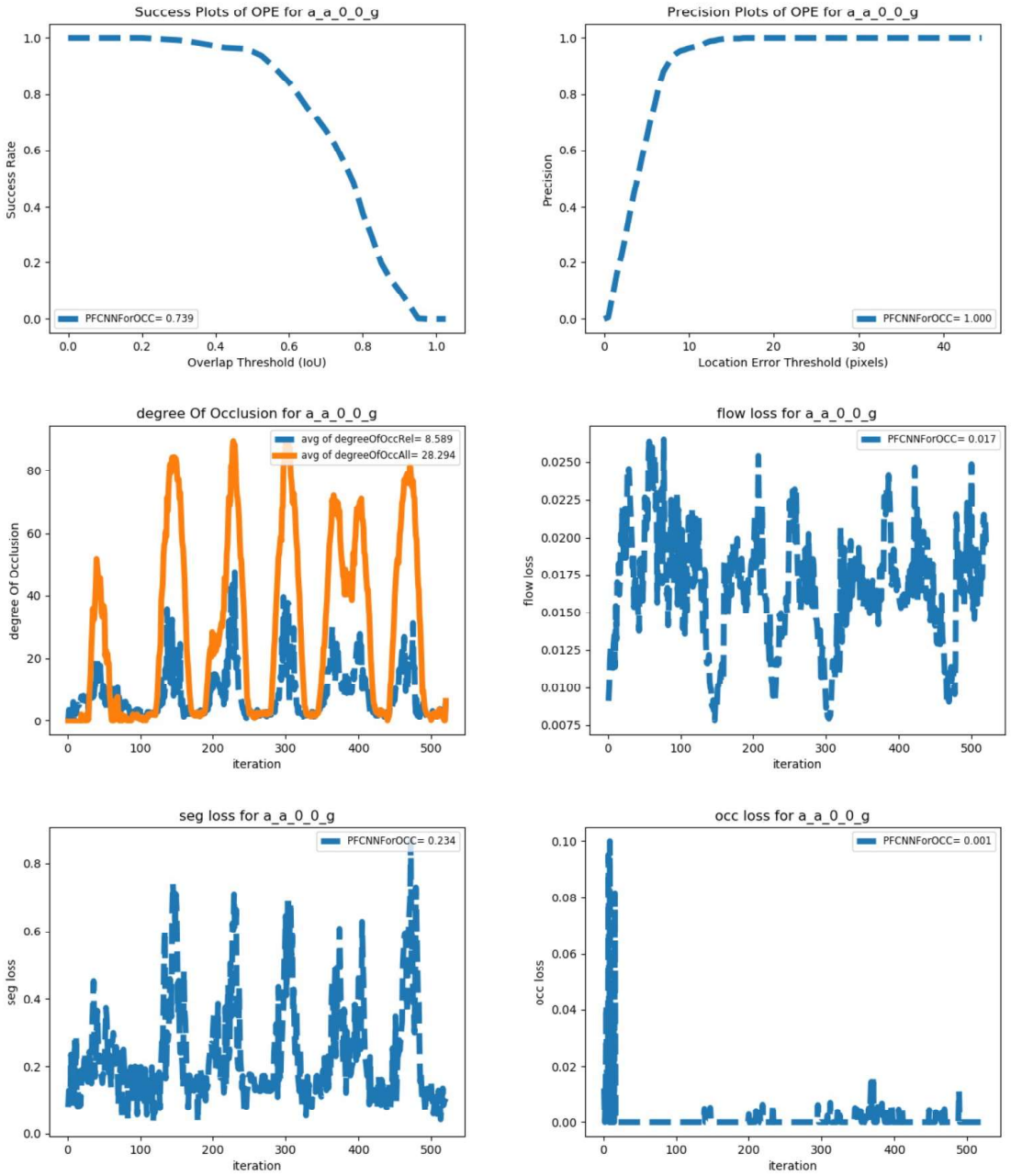
6.3.1.1. Scenario-Based Simulation Results of a_a_0_0_g. A snapshot, taken from the video output during tracking, is displayed in Figure 6.15, and the simulation results are shown in Figure 6.16. The car enters the frame at the 75th iteration, and the scenario stops at the 595th iteration. In this scenario, the car moves on the road in a city, populated with trees, traffic lights and buildings. These structures occlude the car partially and fully for a short period with high frequency. As plotted in Figure 6.16, the precision score is 1, which means that the distance between centers of the predicted/estimated bounding box and the ground truth bounding box is smaller than 20 pixels for all frames except the ones with full occlusion. This means a perfect score in terms of precision. The success score is 0.739. Hence, the method achieves very robust tracking performance under partial and full occlusion. The plot for the degree of occlusion shows the high performance of PFCNNOcc for detecting the 6 partial and full occlusions and the degree of it during tracking. The iterations that the degreeOfOccAll equals 100% represent the images in which the tracked car is being fully occluded. It can be easily observed from the correlation between the occlusion detection plot and the snapshot image, shown in Figure 6.17 and the video tracking results, shared in [98],

that at the iterations when the car is facing partial or full occlusion, it passes under trees with an almost constant time with high frequency. The frequency represents the number of times that the vehicle passes under trees, and the constant time stands for the duration of time between two trees that the vehicle navigates.



Figure 6.15. A snapshot from scenario a_a_0_0_g during tracking.

Even though the car rotates and the aspect ratio of the bounding box changes, PFCNNOcc continues to track with robust performance. When the plots for segmentation loss (seg loss), flow loss, and occlusion loss (occ loss) in Figure 6.16 are examined, the average values of losses are very small. Hence, it can be concluded that predictions of "Flow and Occlusion Model"and "Segmentation Model"are very similar to the ground truth data. The confusion matrix in Figure 6.16 shows that the classification score of the "Classifier Model"which estimates the ID of the vehicle as "car"is 99%. While the car moves only along the positive x-axis of the image coordinates, the x and y components of the vector of $predicted_flow_f$ are annotated on the output image, shown in Figure 6.15. It can easily be verified that flow_f_x is calculated as 1.9 pixels, and the color, representing the motion vector of the pixels in both flow_f and GT_flow_f, is blue. This means that the novel method calculates the flow vector of the car correctly with a robust performance during tracking. All numeric results, presented in Figure 6.16, can also be verified with the video results, shared in [98].



	Bus	Car	Lorry	Truck	Van	Class. Score
Bus	**0**	**0**	**0**	**0**	**0**	
Car	**0**	**521**	**0**	**0**	**1**	**521/522 = %99**
Lorry	**0**	**0**	**0**	**0**	**0**	
Truck	**0**	**0**	**0**	**0**	**0**	
Van	**0**	**0**	**0**	**0**	**0**	

Figure 6.16. Simulation results of scenario a_a_0_0_g.

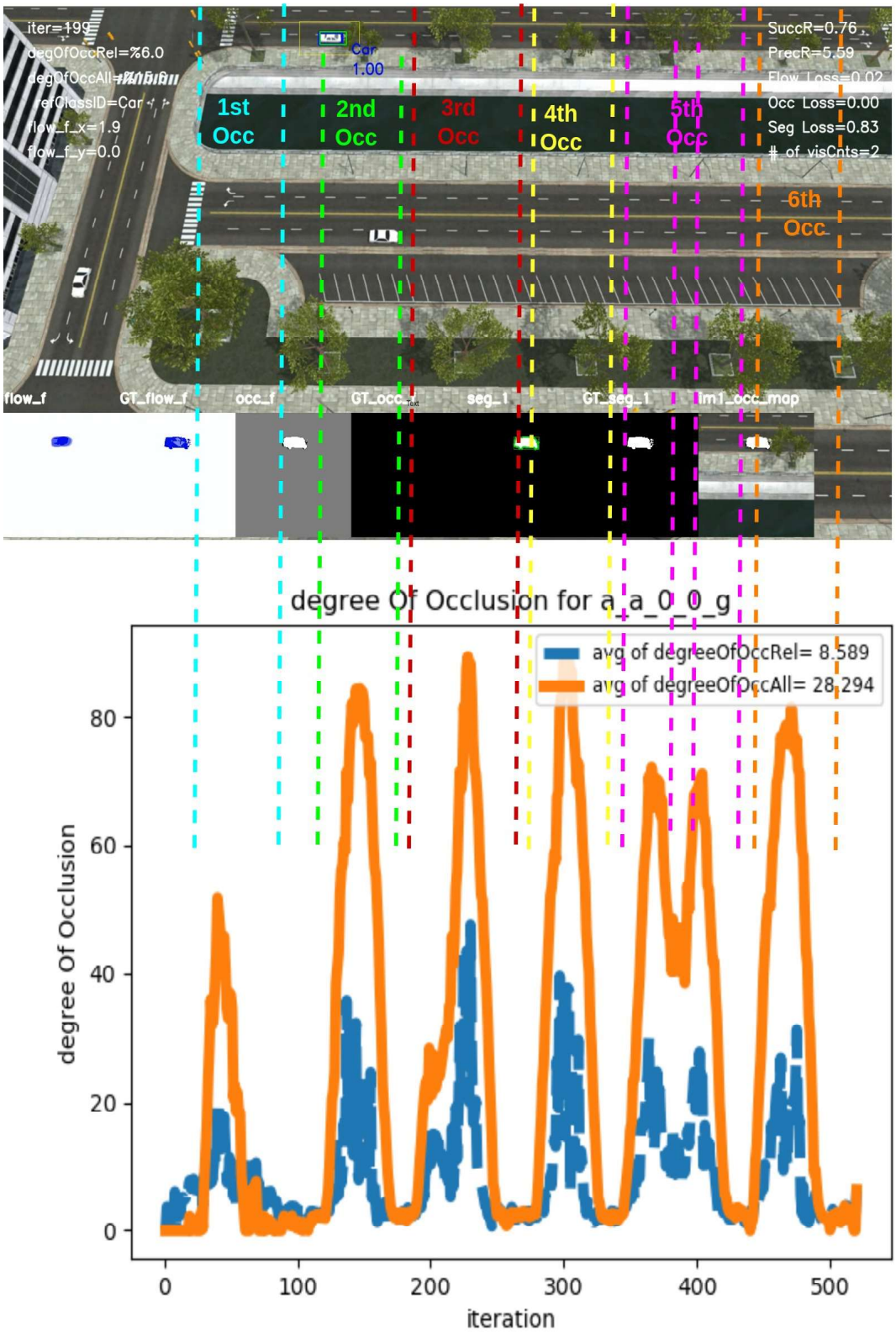


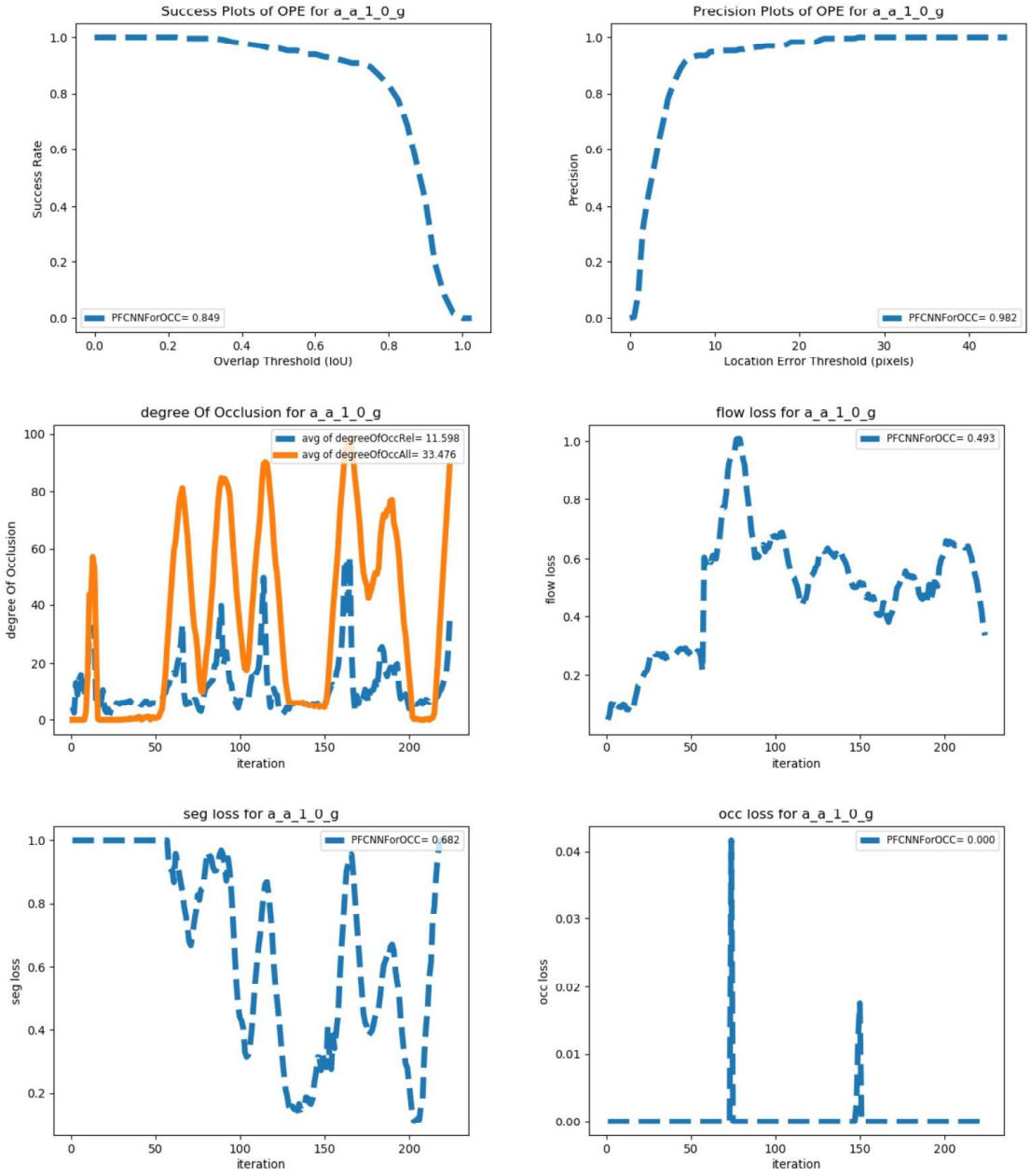
Figure 6.17. The correlation between the occlusion detection plot and the snapshot image.

6.3.1.2. Scenario-Based Simulation Results of a_a_1_0_g. A snapshot, taken from the video output during tracking, is displayed in Figure 6.18, and the simulation results are shown in Figure 6.19. The car enters the frame at the 66th iteration, and the scenario stops at the 290th iteration. In this scenario, the car moves on the road in a city, populated with trees and buildings. These structures occlude vehicles partially and fully for a short period with high frequency. As plotted in Figure 6.19, the precision score is 0.982. This means almost a perfect score in terms of precision. The success score is 0.849, which shows the robustness of the novel method to estimate the precise location and the size of the bounding box at every iteration. Hence, the method achieves very robust tracking performance under partial and full occlusion. The plot for the degree of occlusion shows the high performance of PFCNNOcc for detecting the occlusion and the degree of it during tracking. The iterations that the degreeOfOccAll equals 100% represent the images in which the tracked car is being fully occluded.



Figure 6.18. A snapshot from scenario a_a_1_0_g during tracking.

It can be easily observed from the plot and the video tracking results, shared in [98], that at the iterations when the car is facing partial or full occlusion, it passes under trees with a varying frequency. It means that the duration of time, the car travels between two consecutive trees, fluctuates randomly. Under these circumstances, PFCNNOcc continues to track with robust performance. When plots for segmentation loss (seg loss), flow loss, and occlusion loss (occ loss) in Figure 6.19 are examined, the



	Bus	Car	Lorry	Truck	Van	Class. Score
Bus	**0**	**0**	**0**	**0**	**0**	
Car	**17**	**207**	**1**	**0**	**1**	**207/226 = %92**
Lorry	**0**	**0**	**0**	**0**	**0**	
Truck	**0**	**0**	**0**	**0**	**0**	
Van	**0**	**0**	**0**	**0**	**0**	

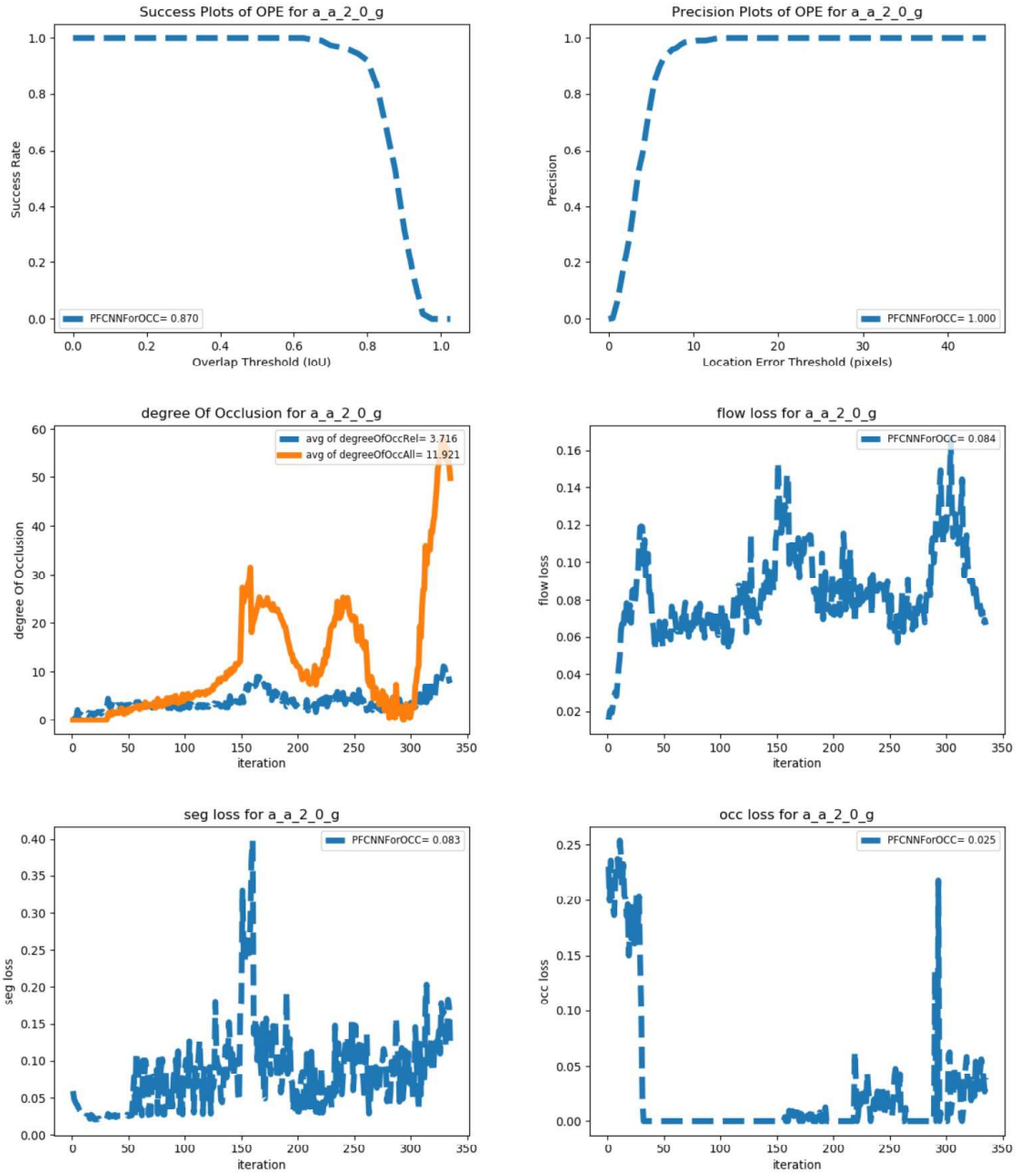
Figure 6.19. Simulation results of scenario a_a_1_0_g.

average values of losses are minimal. They slightly increase during the transition phase to the full occlusion state. Hence, it is concluded that predictions of the "Flow and Occlusion Model" and "Segmentation Model" are similar to the ground truth data. The confusion matrix in Figure 6.19 shows that the classification score of "Classifier Model" which estimates the ID of the vehicle as the car is 92%. While the car moves along the negative x-axis and the positive y-axis of the image coordinates, the x and y components of the vector of $predicted_flow_f$ are annotated on the output image, shown in Figure 6.18. It can easily be verified that flow_f_x and flow_f_y are calculated as -4.6 and 1.2 pixels, and the color, representing the motion vector of the pixels in both flow_f and GT_flow_f, is yellowish green (GT leri değiştirmeyi unutma). This means that the novel method calculates the flow vector of the car correctly with a robust performance during tracking. All numeric results, presented in Figure 6.19, can also be verified with the video results, shared in [98].

6.3.1.3. Scenario-Based Simulation Results of a_a_2_0_g. A snapshot, taken from the video output during tracking, is displayed in Figure 6.20, and the simulation results are shown in Figure 6.21.



Figure 6.20. A snapshot from scenario a_a_2_0_g during tracking.



	Bus	Car	Lorry	Truck	Van	Class. Score
Bus	**0**	**0**	**0**	**0**	**0**	
Car	**0**	**0**	**0**	**0**	**0**	
Lorry	**0**	**0**	**0**	**0**	**0**	
Truck	**2**	**328**	**6**	**1**	**0**	**1/337 = % 0.2**
Van	**0**	**0**	**0**	**0**	**0**	

Figure 6.21. Simulation results of scenario a_a_2_0_g.

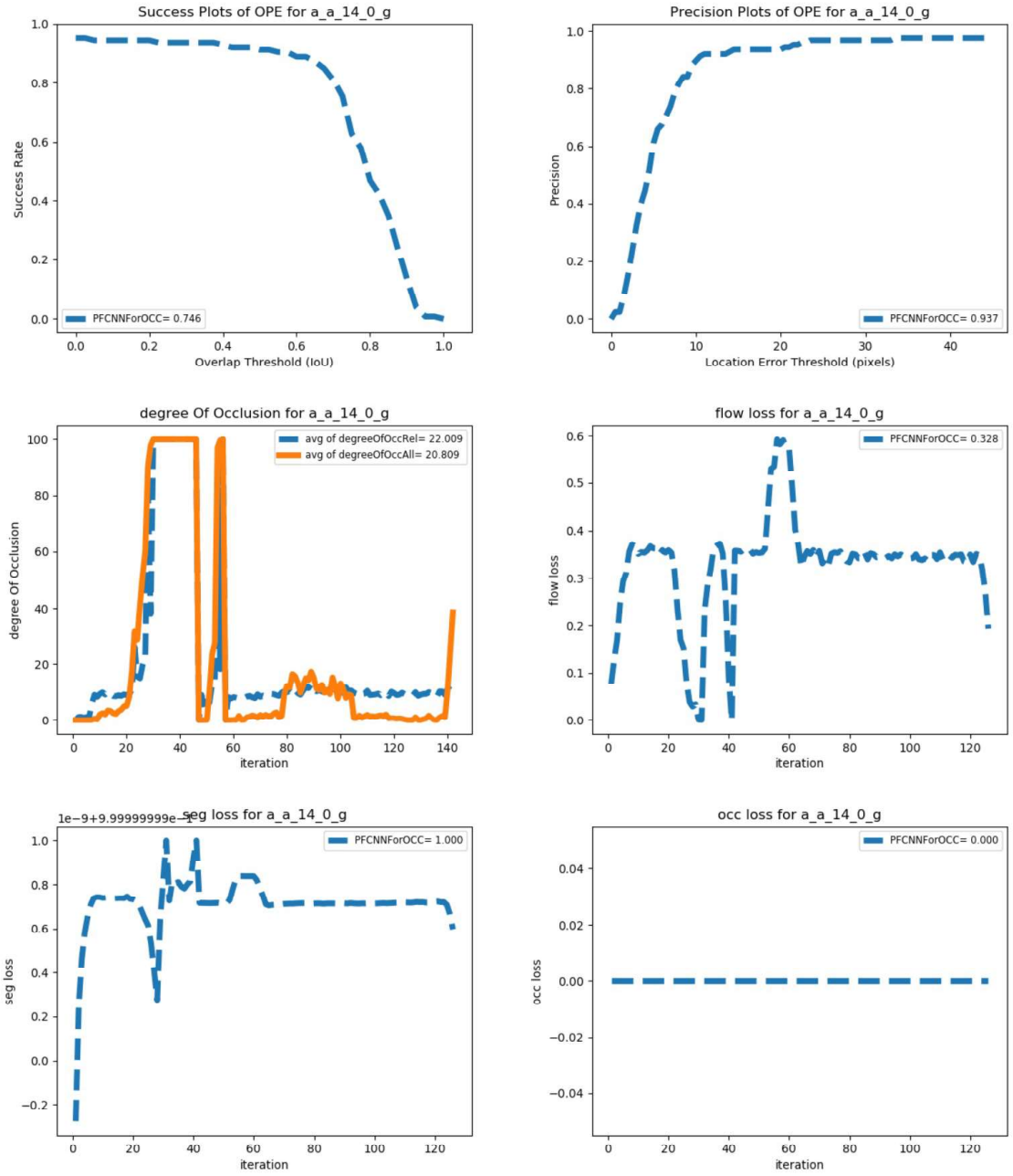
The truck enters the frame at the 5th iteration, and the scenario stops at the 340th iteration. In this scenario, the truck is driven on the road inside a city, populated with rare trees, buildings, and traffic lights. These structures, especially trees, occlude the truck partially for a short period with low frequency. It means that the vehicle is occluded by a few number of time by tree with a short duration of time. As plotted in Figure 6.21, the precision score is 1. This means a perfect score in terms of precision. The success score is 0.870 which shows the robustness of the proposed method to estimate the precise location and the size of the bounding box at every iteration. Hence, the proposed method achieves very robust tracking performance under partial occlusion. The plot for the degree of occlusion shows the high performance of PFCNNOcc for detecting the occlusion as well as its degree during tracking. The iterations that degreeOfOccAll is between 30% and 100% represent the images in which the tracked car is being partially occluded. It can be easily observed from the plot and the video tracking results, shared in the [98], that at the iterations when the truck is facing partial occlusion, it passes under trees with a constant low frequency. Under these circumstances, PFCNNOcc keeps tracking with robust performance. When plots for segmentation loss (seg loss), flow loss, and occlusion loss (occ loss) in Figure 6.21 are examined, the average values of all type of losses are minimal. Hence, it can be concluded that predictions of "Flow and Occlusion Model"and "Segmentation Model"are very similar to the ground truth data. The confusion matrix in Figure 6.21 shows that the classification score of the "Classifier Model"to estimate the ID of the vehicle as a truck is 0.2%. While the truck drives along the negative x-axis and the positive y-axis of the image coordinates, the x and y components of the vector of $predicted_flow_f$ are annotated on the output image, shown in Figure 6.20. Flow_f_x and flow_f_y are calculated as -0.6 and 3.1 pixels, and the color, representing the motion vector of the pixels in both flow_f and GT_flow_f, is light blue. This means that the novel method calculates the flow vector of the truck correctly with a robust performance during tracking. All numerical results, presented in Figure 6.21, can also be verified with the video results, shared in [98].

6.3.1.4. Scenario-Based Simulation Results of a_a_14_0_g. A snapshot, taken from the video output during tracking, is displayed in Figure 6.22, and the simulation results are shown in Figure 6.23.



Figure 6.22. A snapshot from scenario a_a_14_0_g during tracking.

The car enters the frame at the 8th iteration, and the scenario stops at the 150th iteration just before the target goes out of the frame. In this scenario, the car drives on the road inside a city, populated with rare trees, buildings, traffic lights, a road sign, as well as a bridge. While trees and the road sign occlude vehicles partially for a short period, the bridge fully occludes vehicles for a long period. As plotted in Figure 6.23, the precision score is 0.937. This means almost a perfect score in terms of precision. The success score is 0.746, which shows the robustness of the proposed method to estimate the precise location and the size of the bounding box at every iteration. Hence, the method achieves very robust tracking performance under partial and long-term full occlusion. The plot for the degree of occlusion shows the high performance of PFCNNOcc for detecting the occlusion and the degree of it during tracking. The iterations that the degreeOfOccAll equals 100% represent the images in which the tracked car is being fully occluded by the bridge with red color as seen from the plot and video tracking results, shared in [98]. Under these circumstances, PFCNNOcc keeps tracking with robust performance during the long-term full occlusion. When plots for segmentation loss (seg loss), flow loss, and occlusion loss (occ loss) in Figure 6.23 are



	Bus	Car	Lorry	Truck	Van	Class. Score
Bus	**0**	**0**	**0**	**0**	**0**	
Car	**1**	**122**	**0**	**0**	**0**	**122/123 = %99**
Lorry	**0**	**0**	**0**	**0**	**0**	
Truck	**0**	**0**	**0**	**0**	**0**	
Van	**0**	**0**	**0**	**0**	**0**	

Figure 6.23. Simulation results of scenario a_a_14_0_g.

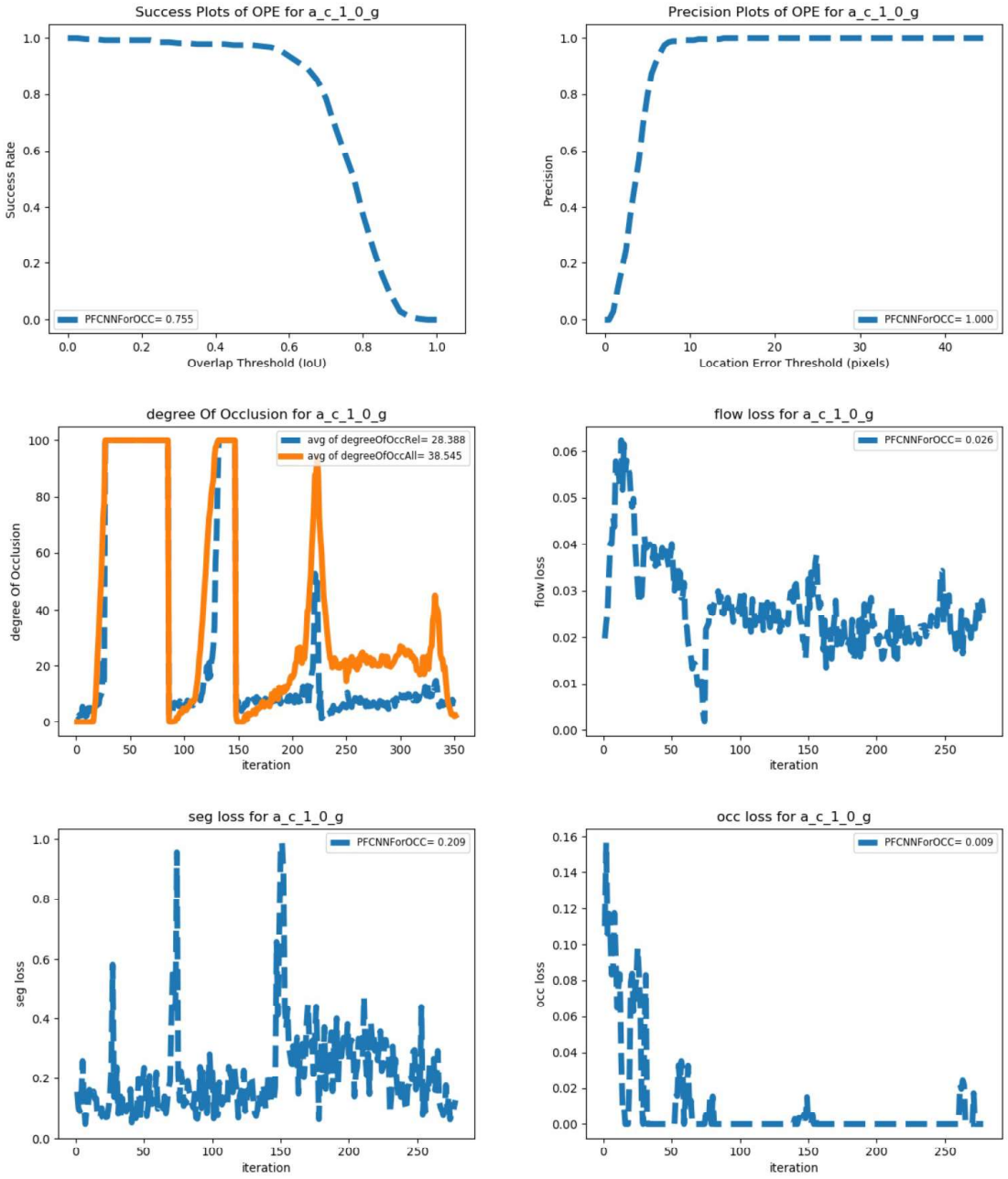
examined, the average values of all loss types are very small. Hence, it can be concluded that predictions of "Flow and Occlusion Model"and "Segmentation Model"are very similar to the ground truth data. The confusion matrix in Figure 6.23 shows that the classification score of "Classifier Model"to estimate the ID of the vehicle as the car is 0.99%. While the car moves along the positive x-axis of the image coordinates, the x and y components of the vector of predicted_flow_f are annotated on the output image, shown in Figure 6.22. Flow_f_x and flow_f_y are calculated as 4.1 and 0.0 pixels, and the color, representing the motion vector of the pixels in both flow_f and GT_flow_f, is blue. This means that the proposed method calculates the flow vector of the car correctly with a robust performance during tracking. All numerical results, presented in Figure 6.23, can also be verified with the video results, shared in [98].

6.3.1.5. Scenario-Based Simulation Results of a_c_1_0_g. A snapshot, taken from the video output during tracking, is displayed in Figure 6.24, and the simulation results are shown in Figure 6.25.



Figure 6.24. A snapshot from scenario a_c_1_0_g during tracking.

The car enters the frame at the 49th iteration, and the scenario stops at the 400th iteration. In this scenario, the car drives on the road inside an industrial facility, populated with buildings, factories, and containers. These structures occlude the target



	Bus	Car	Lorry	Truck	Van	Class. Score
Bus	**0**	**0**	**0**	**0**	**0**	
Car	**4**	**272**	**0**	**0**	**0**	**272/276 = %98**
Lorry	**0**	**0**	**0**	**0**	**0**	
Truck	**0**	**0**	**0**	**0**	**0**	
Van	**0**	**0**	**0**	**0**	**0**	

Figure 6.25. Simulation results of scenario a_c_1_0_g.

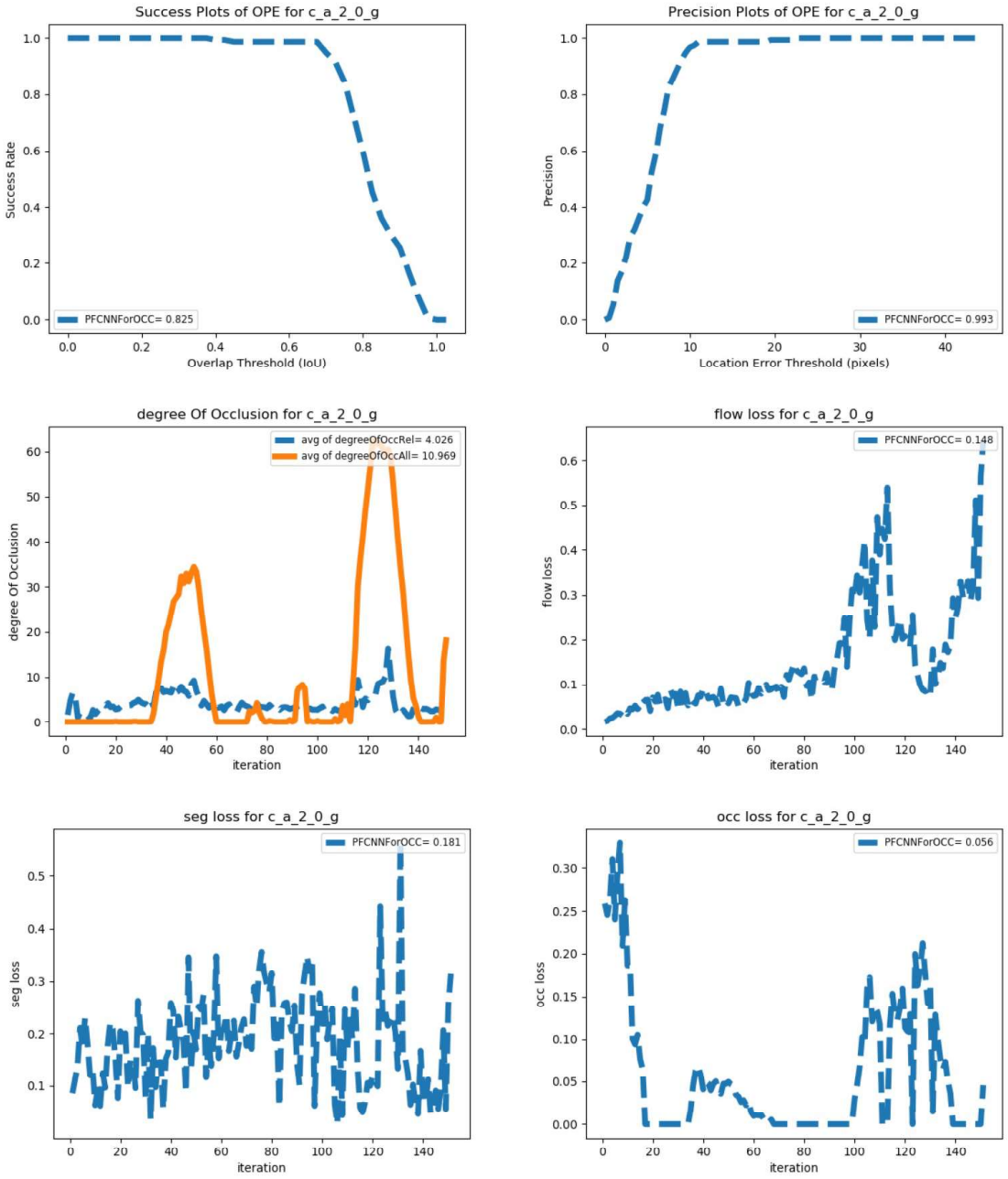
partially and fully for the long term. In the meantime, the car turns for several times which creates drastic aspect ratio variations. As plotted in Figure 6.25, the precision score is 1. This means a perfect score in terms of precision. The success score is 0.755, which shows the robustness of the proposed method to estimate the precise location and the size of the bounding box at every iteration. Hence, the method achieves very robust tracking performance under several long-term partial and full occlusion conditions. The plot for the degree of occlusion shows the high performance of PFCNNOcc for detecting the occlusion and the degree of it during tracking. The iterations that the degreeOfOccAll equals 100% represent the images in which the tracked car is being fully occluded by facilities three times. These can be concluded from the plot and video tracking results, shared in [98]. Under these circumstances, PFCNNOcc keeps tracking with robust performance under several long-term full occlusions using the novel "Re-Identification Module Full Occlusion". When plots for segmentation loss (seg loss), flow loss, and occlusion loss (occ loss) in Figure 6.25 are examined, the average values of all loss types are very small. Hence, it can be concluded that predictions of the "Flow and Occlusion Model"and "Segmentation Model"are very similar to the ground truth data. The confusion matrix in Figure 6.25 shows that the classification score of the "Classifier Model"to estimate the ID of the vehicle as the car is 0.98%. While the car moves along the positive x-axis and negative y-axis of the image coordinates under full occlusion, the x and y components of the vector of $predicted_flow_f$ are annotated on the output image, shown in Figure 6.22. In this situation, the motion vector is not calculated because of the full occlusion, and the flow vector is predicted using the last known value of the predicted flow vector from the previous iteration. All numerical results, presented in Figure 6.23, can also be verified with the video results, shared in [98].

6.3.1.6. Scenario-Based Simulation Results of c_a_2_0_g. A snapshot, taken from the video output during tracking, is displayed in Figure 6.26, and the simulation results are shown in Figure 6.27.



Figure 6.26. A snapshot from scenario c_a_2_0_g during tracking.

The truck enters the frame at the 49th iteration, and the scenario stops at the 200th iteration. In this scenario, the truck moves on the road inside a forest, populated with very dense trees and a number of road sign structures. These objects occlude the vehicle partially for a short term with a low frequency. It means that the truck will be occluded a few times with a short duration of time. The truck faces illumination variation because of the shade of the trees. Also, the close proximity of the vehicles increases the complexity of the tracking, thus in this case, the precision score is 0.993 as observed in Figure 6.27. This means an almost perfect score in terms of precision. The success score is 0.825, which shows the robustness of the proposed method to estimate the precise location and the size of the bounding box at every iteration. Hence, the method achieves robust tracking performance under several short-term partial occlusion conditions and illumination variation. The plot for the degree of occlusion shows the high performance of PFCNNOcc for detecting the occlusion and the degree of it during tracking. The iterations that the degreeOfOccAll is between 30% and 65% represent the images in which the tracked truck is being partially occluded by the trees two times, as seen from the plot and video tracking results, shared in [98]. Under these circumstances, PFCNNOcc keeps tracking with robust performance under several short-term partial occlusions using the novel “Re-Identification Module Full Occlusion”while the other vehicles drive within close proximity. When plots for segmentation loss (seg loss), flow



	Bus	Car	Lorry	Truck	Van	Class. Score
Bus	**0**	**0**	**0**	**0**	**0**	
Car	**0**	**0**	**0**	**0**	**0**	
Lorry	**0**	**0**	**0**	**0**	**0**	
Truck	**68**	**43**	**0**	**42**	**0**	**42/153 = % 28**
Van	**0**	**0**	**0**	**0**	**0**	

Figure 6.27. Simulation results of scenario c_a_2_0_g.

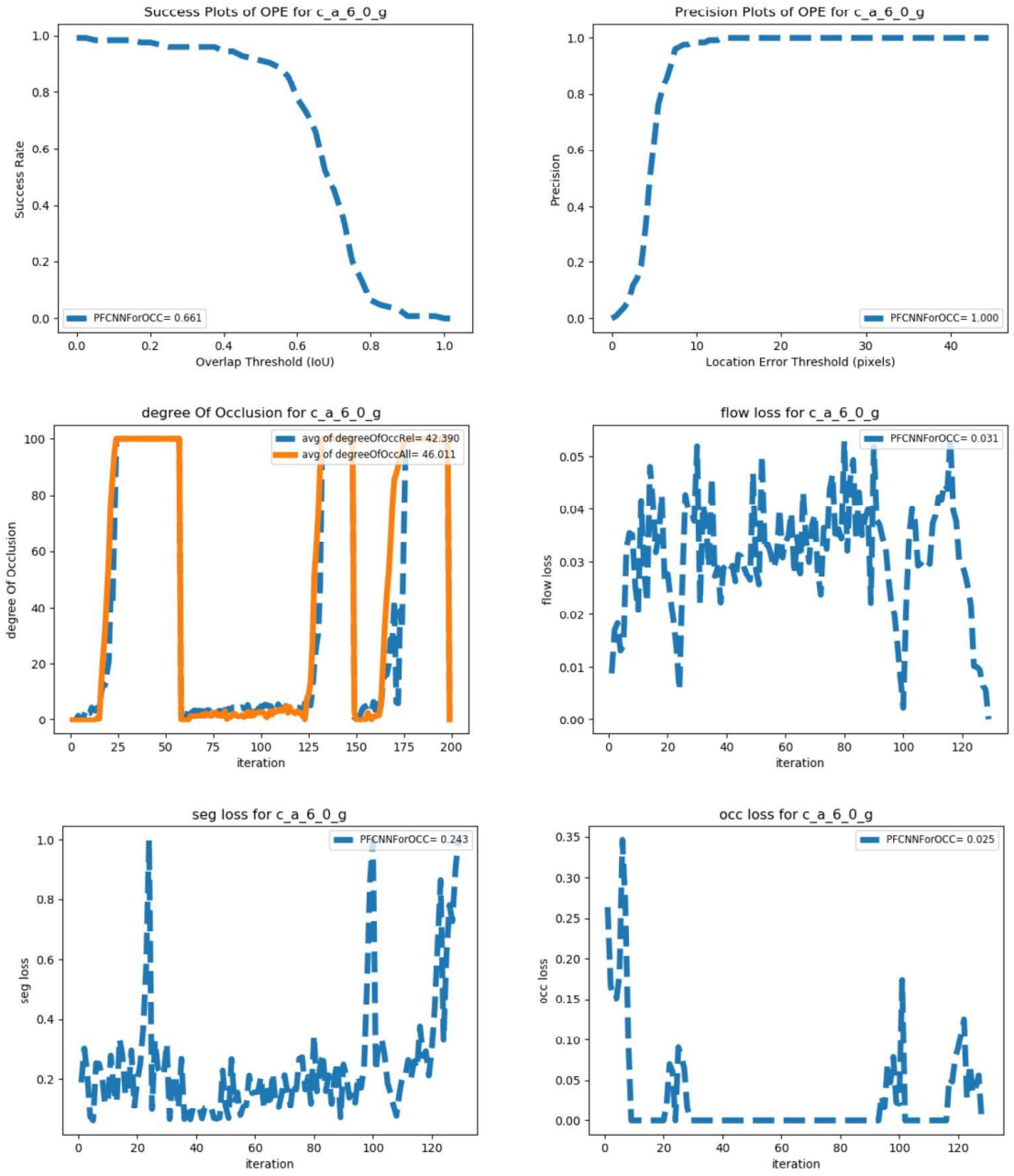
loss, and occlusion loss (occ loss) in Figure 6.27 are examined, the average values of all loss types are very small. Hence, it can be concluded that predictions of "Flow and Occlusion Model"and "Segmentation Model"are very similar to the ground truth data. The confusion matrix in Figure 6.27 shows that the classification score of "Classifier Model"to estimate the ID of the vehicle as the car is 0.28%. While the truck drives along the negative x-axis and the positive y-axis of the image coordinates, the x and y components of the vector of $predicted_flow_f$ are annotated on the output image, shown in Figure 6.26. Flow_f_x and flow_f_y are calculated as -2.4 and 1.8 pixels, and the color, representing the motion vector of the pixels in both flow_f and GT_flow_f, is light green. This means that the proposed method calculates the flow vector of the truck correctly with a robust performance during tracking. All numerical results, presented in Figure 6.27, can also be verified with the video results, shared in [98].

6.3.1.7. Scenario-Based Simulation Results of c_a_6_0_g_1. A snapshot, taken from the video output during tracking, is displayed in Figure 6.28, and the simulation results are shown in Figure 6.29. The car enters the frame at the 51st iteration, and the scenario stops at the 250th iteration. In this scenario, the car drives on the road inside a rural area, populated with trees and a road sign structure. The trees occlude the car fully for a long-term with low frequency. In the meantime, the car faces illumination variation because of the shade of the trees. As plotted in Figure 6.29, the precision score is 1. This means a perfect score in terms of precision. The success score is 0.661, which shows the robustness of the novel method to estimate the precise location and the size of the bounding box at every iteration. Hence, the method achieves very robust tracking performance under several long-term full occlusion conditions and illumination variations.



Figure 6.28. A snapshot from scenario c_a_6_0_g_1 during tracking

The plot for the degree of occlusion shows the high performance of PFCNNOcc for detecting the occlusion and the degree of it during tracking. The iterations that the degreeOfOccAll is 100% represent the images in which the tracked car is being fully occluded by the tree three times, as observed from the plot and video tracking results, shared in [98]. Under these circumstances, PFCNNOcc keeps tracking with robust performance under several long-term full occlusions using the novel "Re-Identification Module Full Occlusion". When plots for segmentation loss (seg loss), flow loss, and occlusion loss (occ loss) in Figure 6.29 are examined, the average values of all loss types are very small. Hence, it can be concluded that predictions of the "Flow and Occlusion Model"and "Segmentation Model"are very similar to the ground truth data. The confusion matrix in Figure 6.29 shows that the classification score of the "Classifier Model"to estimate the ID of the vehicle as the car is 0.99%. While the car moves along the negative x-axis and positive y-axis of the image coordinates under full occlusion, the x and y components of the vector of $predicted_flow_f$ are annotated on the output image, shown in Figure 6.28. In this situation, the motion vector is not calculated because of the full occlusion, and the flow vector is predicted using the last known value of the predicted flow vector from the previous iteration. All numerical results, presented in Figure 6.29, can also be verified with the video results, shared in [98].



	Bus	Car	Lorry	Truck	Van	Class. Score
Bus	**0**	**0**	**0**	**0**	**0**	
Car	**1**	**125**	**0**	**0**	**0**	**125/126 = %99**
Lorry	**0**	**0**	**0**	**0**	**0**	
Truck	**0**	**0**	**0**	**0**	**0**	
Van	**0**	**0**	**0**	**0**	**0**	

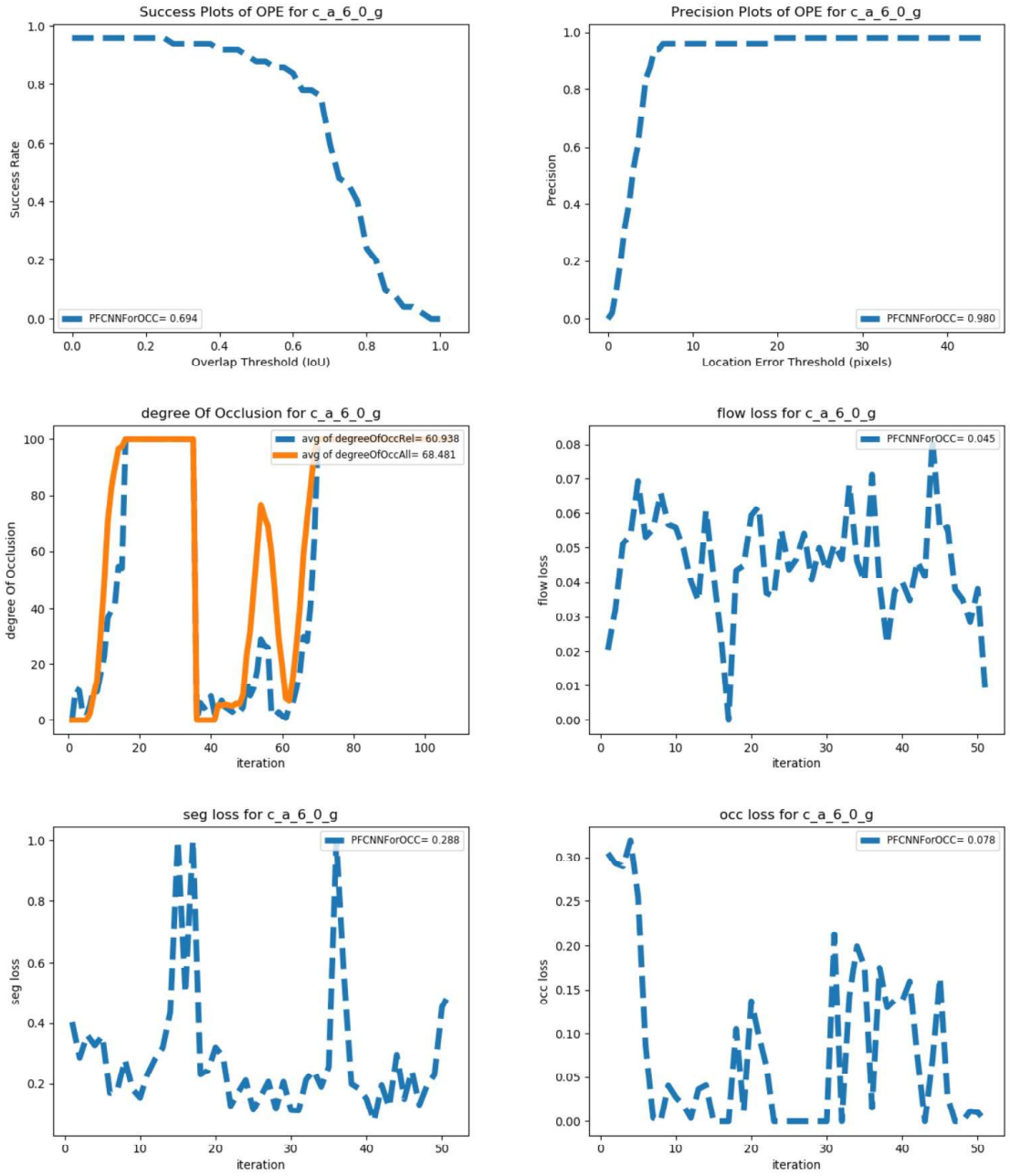
Figure 6.29. Simulation results of scenario c_a_6_0_g_1.

6.3.1.8. Scenario-Based Simulation Results of c_a_6_0_g_2. A snapshot, taken from the video output during tracking, is displayed in Figure 6.30, and the simulation results are shown in Figure 6.31.



Figure 6.30. A snapshot from scenario c_a_6_0_g_2 during tracking.

The car enters the frame at the 126th iteration, and the scenario stops at the 233rd iteration. In this scenario, the car drives on the road inside a rural area, populated with trees and a road sign structure. These objects occlude the vehicle partially for the short term and fully for the short and long term with high frequency. In the meantime, the car faces the illumination variation because of the shade of the trees. As plotted in Figure 6.31, the precision score is 0.980. This means an almost perfect score in terms of precision. The success score is 0.694, which shows the robustness of the proposed method to estimate the precise location and the size of the bounding box at every iteration. Hence, the method achieves robust tracking performance under several short-term and long-term full occlusion conditions and illumination variation. The plot for the degree of occlusion shows the high performance of PFCNNOcc for detecting the occlusion and the degree of it during tracking. The iterations that the degreeOfOccAll is between 30% and 100% represent the images in which the tracked car is being partially and fully occluded by the road sign structure and the trees, three times as observed from the plot and video tracking results, shared in [98]. Under these circumstances, PFCNNOcc keeps tracking with robust performance under several short-term



	Bus	Car	Lorry	Truck	Van	Class. Score
Bus	**0**	**0**	**0**	**0**	**0**	
Car	**3**	**43**	**1**	**0**	**2**	**43/49 = %87**
Lorry	**0**	**0**	**0**	**0**	**0**	
Truck	**0**	**0**	**0**	**0**	**0**	
Van	**0**	**0**	**0**	**0**	**0**	

Figure 6.31. Simulation results of scenario c_a_6_0_g_2.

and long-term, partial and full occlusions using the novel "Re-Identification Module Full Occlusion". When plots for segmentation loss (seg loss), flow loss and occlusion loss (occ loss) in Figure 6.31 are examined, the average values of all loss types are minimal. Hence, it can be concluded that predictions of the "Flow and Occlusion Model"and "Segmentation Model"are similar to the ground truth data. The confusion matrix in Figure 6.31 shows that the classification score of "Classifier Model"to estimate the ID of the vehicle as the car is 0.87%. While the car drives along the positive x-axis of the image coordinates, the x and y components of the vector of $predicted_flow_f$ are annotated on the output image, shown in Figure 6.26. Flow_f_x and flow_f_y are calculated as 5.2 and 0.0 pixels, and the color, representing the motion vector of the pixels in both flow_f and GT_flow_f, is blue. This means that the proposed method calculates the flow vector of the car correctly with a robust performance during tracking. All numerical results, presented in Figure 6.27, can also be verified with the video results, shared in [98].

6.3.2. Simulation Results of the Final Version of the Tracking Model and the State-Of-The-Art Trackers on the UAV123 10fps Dataset

Simulation results of the final version of the PFCNNOcc and the state-of-the-art trackers on the UAV123 10 fps dataset [2] are extracted and evaluated in three different comparison approaches in terms of precision and success scores; 1. Overall comparison, 2. Attribute-based comparison and 3. Scenario-based comparison. While the overall comparison, elaborated in Section 6.3.2.1, presents the final average performance of the methods against all visual visual problems, attribute-based evaluation, in Section 6.3.2.2, visualize the performance analysis of the methods depending on the all visual problems, referenced in this thesis. Scenario-based comparison puts forward the performance of the trackers based on each scenario or video in Section 6.3.3.3. And the videos, showing the simulation results during tracking for each scenario, are presented in [99], and they are comprised of the final tracking result, predicted forward and backward optical flow and occlusion maps, segmentation masks, and processed occlusion maps on the images and cropped ROI images. The details of the information, anno-

tated on the images in the final tracking result videos, are presented in the following items.

- The information, shown on the top left side of the image: The number of iterations (iter), degreeOfOccRel (degOfOccRel), degreeOfOccAll (degOfOccAll), reference class ID (refClassID), predicted forward flow along the x and y axes, flow_f_x and flow_f_y and the ground truth values for these vectors, flow_f_x_GT and flow_f_y_GT, and the threshold values, pixelCountLowerBound and sizeCountLowerBound.
- The information, shown on the top right side of the image: Success rate or score (SuccR), precision rate or score (PrecR), the number of visible contours inside the ROI search and the number of these that satisfy the contourFlowCheck conditions, # of visCnts and # of visCntsProc, the value of the flags to monitor the state of the scaleChange, rescale and rotation situation, and the reference number of the pixel count for the target, actualPixel, as well as the instantaneous number of pixel count for the target at that specific frame, mask2Pixel.
- Rectangles/squares and annotations around the target: Green rectangle/square (ground truth for the bounding box), blue rectangle/square (estimated/predicted bounding box), yellow rectangle/square (ROI), estimated class ID name (blue), and class score (blue).

6.3.2.1. Overall Comparison. Overall comparison between the final version of the PFCNNOcc and the SOTA trackers on the UAV123 10fps dataset [2] is shown in Figure 6.32 and Table 6.1. As understood from the simulation results, PFCNNOcc outperforms all SOTA trackers in terms of precision and success scores against all visual problems, faced in 21 scenarios, summarized in Table 5.1. The precision score of the proposed method is 0.943, and it means that PFCNNOcc can estimate the location of the bounding box with an error smaller than 20 pixels for 94.3% of the frames of all videos. Even the Siamese Network-based CNN methods are not able to achieve such a high precision tracking score, and SiamAPN [35] is able to reach a precision score of 0.863, which is the closest to that of the PFCNNOcc. The success score of the

PFCNNOcc is 0.763, and SiamAPN has the closest score with 0.756. This score verifies that the PFCNNOcc is very capable of tracking not only the location but also the size of the bounding box of the target compared to the SOTA trackers. The superiority of the final version of the PFCNNOcc against the SOTA arises from the novel functions of the PFCNNOcc, elaborated in the following items.

- Re-Identification Module Final PFCNNOcc.
- Occlusion Detector and Calculator Final PFCNNOcc.
- Full Occlusion Dense Search Module Final PFCNNOcc.
- Merged Target Detection.

Table 6.1. Overall performance of the final version of PFCNNOcc and the SOTA trackers. The best three performances are highlighted in red, green, and blue colors.

Trackers	Precision Score	Success Score
DSST	0.404	0.276
DaSiamRPN	0.799	0.674
ECO	0.775	0.562
MEEM	0.316	0.289
MUSTER	0.364	0.357
RTMDNet	0.823	0.565
SRDCF	0.417	0.433
SiamAPN	0.883	0.756
SiamAPN++	0.869	0.738
SiamRPN	0.830	0.691
Struck	0.353	0.296
StruckUAV	0.441	0.378
PFCNNOcc	0.943	0.763

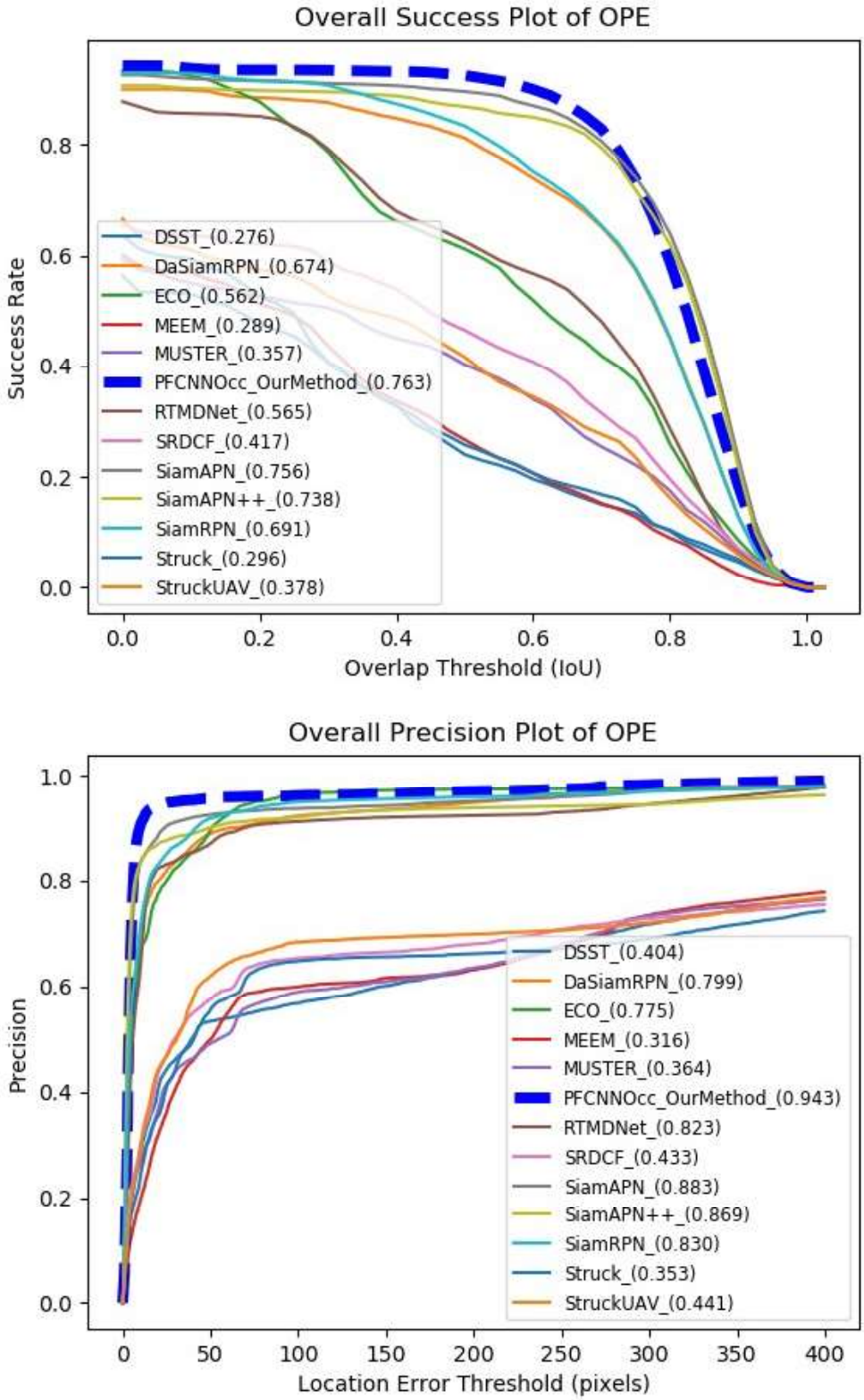


Figure 6.32. Overall precision and success plots of the SOTA trackers and the final version of PFCNNOcc.

"Re-Identification Module Final PFCNNOcc"utilizes the forward and backward optical flow predictions, and the segmentation masks of the consecutive images with a novel matching algorithm to re-detect the target at the beginning of each image pair and to find the location and the size of the target at the pixel level with high precision. In the meantime, the average motion vector for the target is also calculated. The location, size, and motion vector, calculated with high accuracy by the matching algorithm, pave the way to boost the tracking performance of the final version of the PFCNN. "Occlusion Detector and Calculator Final PFCNNOcc"is very vital to decide whether a pixel is non-occluded, will-be-occluded, and already-occluded, and to calculate the degree of occlusion by considering the possible scale and aspect ratio variations. This novel function increases the robustness of the method against partial and full occlusion, and hence, it makes a difference compared to the SOTA trackers. "Full Occlusion Dense Search Module Final PFCNNOcc"is a combination of the "Re-Identification Module Final PFCNNOcc"and "Occlusion Detector and Calculator Final PFCNNOcc"with a dense-search algorithm to re-detect the target after a short-term or a long-term occlusion with varying frequencies or periods. The overall performance of most of the SOTA trackers decreases because of the full occlusion condition, creating a huge uncertainty in terms of the location and size after the target vanishes. The amount of uncertainty is cumbersome for the trackers to handle after the target appears on the scene again at the end of a long-term full occlusion. Hence, "Full Occlusion Dense Search Module Final PFCNNOcc"has the key contribution to the best performance of the PFCNNOcc. The merged target detection function at the end of the iteration detects whether a target is occluded by another vehicle during tracking. After the occlusion or merged-target condition ends, the discrimination capability of the function that separates the target from other vehicles on the run helps the PFCNNOcc to continue tracking precisely. All these novel functions lead the final PFCNNOcc to the top of SOTA in terms of precision and success scores.

The threshold, $TH1$, is utilized to decide the transition from No occlusion to Partial occlusion, and the selection of this threshold is dependent on the deterioration level of the visual model of the target. Even if 25-30 % of the target is occluded, it is

verified that the feature of the visual model under the occlusion has changed slightly, and the similarity score between its feature and the reference feature becomes very high. It is observed that as the degree of occlusion increases, the visual model of the target changes, and the variance of the visual model of the target gets very high. Therefore, during tracking, the update of the visual model is stopped, and the particle filter is not used to estimate the target kinematics anymore because of the high variance in the visual model that the occlusion triggers. All in all, 30% is assumed to be the optimal threshold to decide the transition between No Occlusion and Partial Occlusion states in the proposed method.

6.3.2.2. Attribute-Based Comparison. In attribute-based comparison, the final PFCNNOcc and the SOTA trackers are compared depending on their performances against scale change (SC), aspect ratio variation (ARV), fast motion (FM), out-of-view (OOV), low resolution (LR), viewpoint change (VC), camera motion (CM), similar object (SO), partial occlusion (PO), full occlusion (FO), and illumination variation (IV). As seen from the precision results, shown in Figures 6.33, 6.34 and summarized in Table 6.2, the final PFCNNOcc outperforms the SOTA trackers against 9 visual problems and achieves the second highest performance against full occlusion with a very high precision score of 0.920.

The novel functions, “Re-Identification Module Final PFCNNOcc”and “Occlusion Detector and Calculator Final PFCNNOcc”, provide the superiority of the PFCNNOcc against the partial occlusion, even though the SOTA methods, such as ECO [28], RT-MDNet [25], SiamAPN [35], SiamAPN++ [39], are also designed based on CNNs. “Full Occlusion Dense Search Module Final PFCNNOcc”function makes the performance of the final PFCNNOcc against full occlusion almost perfect via its dense-search algorithm, and carries it to the top of the SOTA trackers with SiamAPN [35]. The novel final PFCNNOcc and SiamAPN [35] are by far the best methods against full occlusion.

The final PFCNNOcc achieves the top performances in terms of average precision score and average success score, as 0.8875 and 0.663 against both partial occlusion and full occlusion, respectively. The main contribution of these functions results from the optical flow predictions and the segmentation masks that are used to segment the target, while the occlusion state of each segmented pixel is decided and marked.

Attribute-based success scores of the final version of the PFCNNOcc and the SOTA methods are plotted in Figures 6.35, 6.36 and summarized in Table 6.3, and it is easily realized that the final PFCNNOcc outperforms the SOTA trackers against 8 visual problems and achieves the second highest performance against full occlusion and viewpoint change with high success scores. The robustness in success scores is directly related to the correctness of not only the location but also the size of the target, estimated at the end of each iteration.

The novel functions, "Re-Identification Module Final PFCNNOcc"and "Occlusion Detector and Calculator Final PFCNNOcc", provide the superiority of the PFCNNOcc against the partial occlusion, even though the SOTA methods, such as ECO [28], RT-MDNet [25], SiamAPN [35], SiamAPN++ [39], are also designed based on CNNs. "Full Occlusion Dense Search Module Final PFCNNOcc"function makes the performance of the final PFCNNOcc against full occlusion almost perfect via its dense-search algorithm, and carries it to the top of the SOTA trackers with SiamAPN [35]. The novel final PFCNNOcc and SiamAPN [35] are by far the best methods against full occlusion. The final PFCNNOcc achieves the top performances in terms of average precision score and average success score, as 0.8875 and 0.663 against both partial occlusion and full occlusion, respectively. The main contribution of these functions results from the optical flow predictions and the segmentation masks that are used to segment the target, while the occlusion state of each segmented pixel is decided and marked.

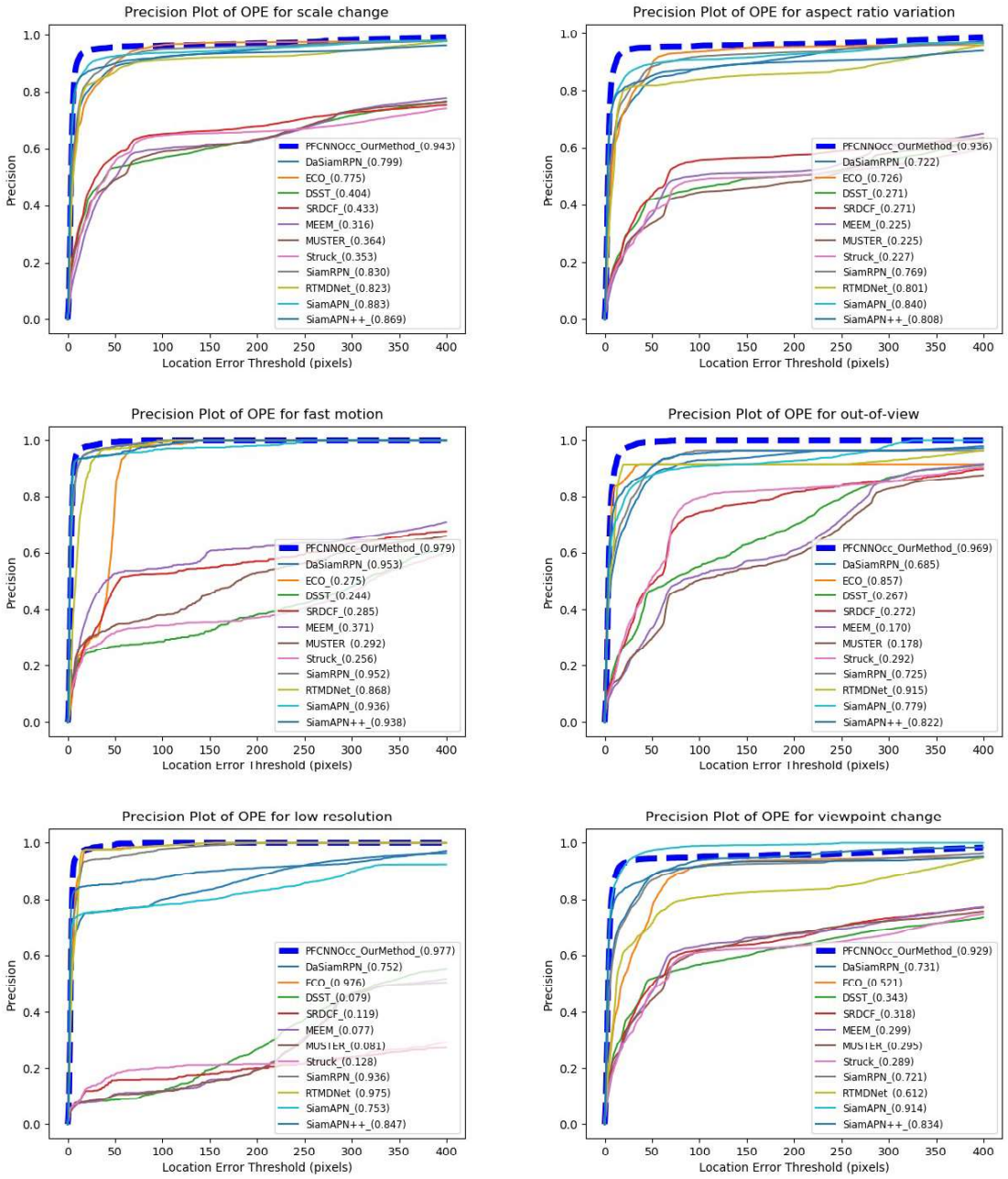


Figure 6.33. Attribute-based precision plots of the SOTA trackers and the final version of PFCNNOcc.

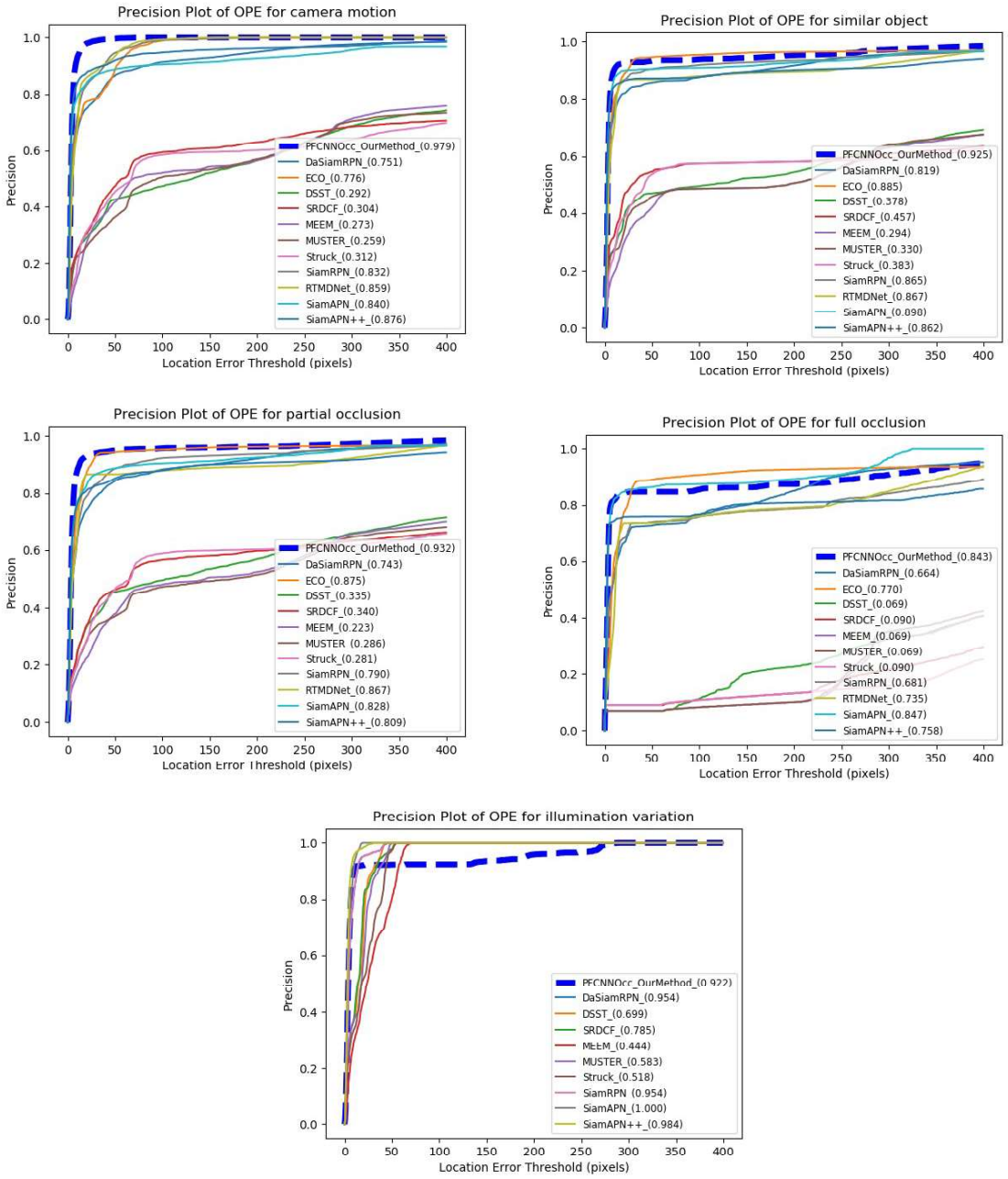


Figure 6.34. Attribute-based precision plots of the SOTA trackers and the final version of PFCNNOcc. (cont.)

Table 6.2. Attribute-based precision results of the SOTA trackers and the final version of PFCNNOcc. The best three performances are, respectively highlighted in red, green, and blue colors.

Attribute-Based Precision Scores											
Trackers	PO	FO	SC	ARV	LR	FM	OOV	IV	VC	CM	SO
DaSiamRPN	0.743	0.664	0.799	0.722	0.752	0.953	0.685	0.954	0.731	0.751	0.819
ECO	0.875	0.770	0.775	0.726	0.976	0.275	0.857	-	0.521	0.776	0.885
DSST	0.335	0.069	0.404	0.271	0.079	0.244	0.267	0.699	0.343	0.292	0.378
SRDCF	0.340	0.090	0.433	0.271	0.119	0.285	0.272	0.785	0.318	0.340	0.457
MEEM	0.223	0.0.069	0.316	0.225	0.077	0.371	0.170	0.444	0.299	0.223	0.294
MUSTER	0.286	0.069	0.364	0.225	0.081	0.292	0.178	0.583	0.295	0.286	0.330
Struck	0.281	0.090	0.353	0.227	0.128	0.256	0.292	0.518	0.289	0.312	0.383
SiamRPN	0.790	0.681	0.830	0.769	0.936	0.952	0.725	0.954	0.721	0.832	0.865
RTMDNet	0.867	0.735	0.823	0.801	0.975	0.868	0.915	-	0.612	0.859	0.867
SiamAPN	0.828	0.847	0.883	0.840	0.753	0.936	0.779	1.000	0.914	0.840	0.898
SiamAPN++	0.809	0.758	0.869	0.808	0.847	0.938	0.822	0.984	0.834	0.876	0.862
PFCNNOcc	0.932	0.843	0.943	0.936	0.977	0.979	0.969	0.922	0.929	0.979	0.925

The final PFCNNOcc achieves the best precision and success scores compared to the Siamese network-based trackers, SiamAPN [35] and SiamAPN++ [39], against scale change and aspect ratio variation by means of the “Segmentation Model”, utilized to segment the targets on the scene. The pixel-wise segmentation of the targets boosts the detection capability of the “Re-Identification Module Final PFCNNOcc”function. Whenever the target faces scale change and aspect ratio variation, this function can localize the target as well as its dimensions with very high precision, and provides the final PFCNNOcc invariance to these visual problems. In the meantime, the “Occlusion Detector and Calculator Final PFCNNOcc”function controls the presence of the scale change and aspect ratio, considering the possible occlusion state.

The precision and success scores of the final PFCNNOcc for the videos, including low resolution and illumination variation problems, show the robustness of the CNN models. While the “Segmentation Model”produces high precision segmentation masks, and “Classifier Model”extracts accurate features and class predictions for the vehicles on the scene, the method estimates the location and the size of the target as perfectly as possible by means of the novel functions of the final PFCNNOcc. While the final PFCNNOcc achieves the SOTA precision and success scores against low resolution, it is beneath the top-5 performing methods, such as SiamAPN [35], SiamAPN++ [39], SiamRPN [37], DaSiamRPN [38], and itself against the illumination variation problem. As seen from the results, the precision score of PFCNNOcc is very close to those of Siamese-based methods, and outperforms the methods by far. It is thought that the small gap between PFCNNOcc and Siamese-based methods can be closed by fine-tuning the CNN methods with the S-UAV and UAV123 datasets that are augmented with more images, and illumination variation is applied with different levels.

The final PFCNNOcc is by far the best method among the SOTA methods against the out-of-view problem in terms of the precision and success scores. This superiority of the proposed method results from the novel functions, “Re-Identification Module Final PFCNNOcc”, “Occlusion Detector and Calculator Final PFCNNOcc”, and “Full Occlusion Dense Search Module Final PFCNNOcc”, that are also key functions to

tracking the target under the partial and full occlusion state. When the target is out of view or the scene, it means that the full occlusion state occurs. Then, “Full Occlusion Dense Search Module Final PFCNNOcc”function handles the problem, and whenever the target is seen after a short-term or long-term full occlusion, it can re-detect the target and hand over the tracking process to other novel functions successfully. These are also verified by the precision and success scores, derived against partial occlusion, full occlusion, and out-of-view.

For the videos where the target shows a nonlinear motion with a large displacement vector as well as drastic variations in its visual model because of fast motion, view point change and camera motion variation, the final PFCNNOcc outperforms the SOTA trackers using the dense ROI sampling scheme of the “Re-Identification Module Final PFCNNOcc”function, precise predictions of “Flow Model”, “Segmentation Model”and “Classifier Model”, the reference feature update mechanism, utilized at the end of each iteration if the degree of occlusion is smaller than 30%, and the strategy to decide whether the rescaling is required depending on the size of the target from the previous iteration. Even the methods, using Siamese-based networks, can not achieve the high precision and success scores that the final PFCNNOcc has.

The proposed method is also very robust at image sequences that include similar objects or vehicles, and achieves the SOTA precision and success scores with an important performance difference. The robustness against this visual problem results from the matching algorithm of the “Re-Identification Module Final PFCNNOcc”function as well as the merged-target detection function. The second-best method, SiamAPN [35], can not perform a higher precision score than the proposed method, even though it is designed with a Siamese-based network architecture, searching for a reference or a template frame in the parallel running networks. The final PFCNNOcc achieves the top-3 performing success score that is almost the same as SiamAPN [35].

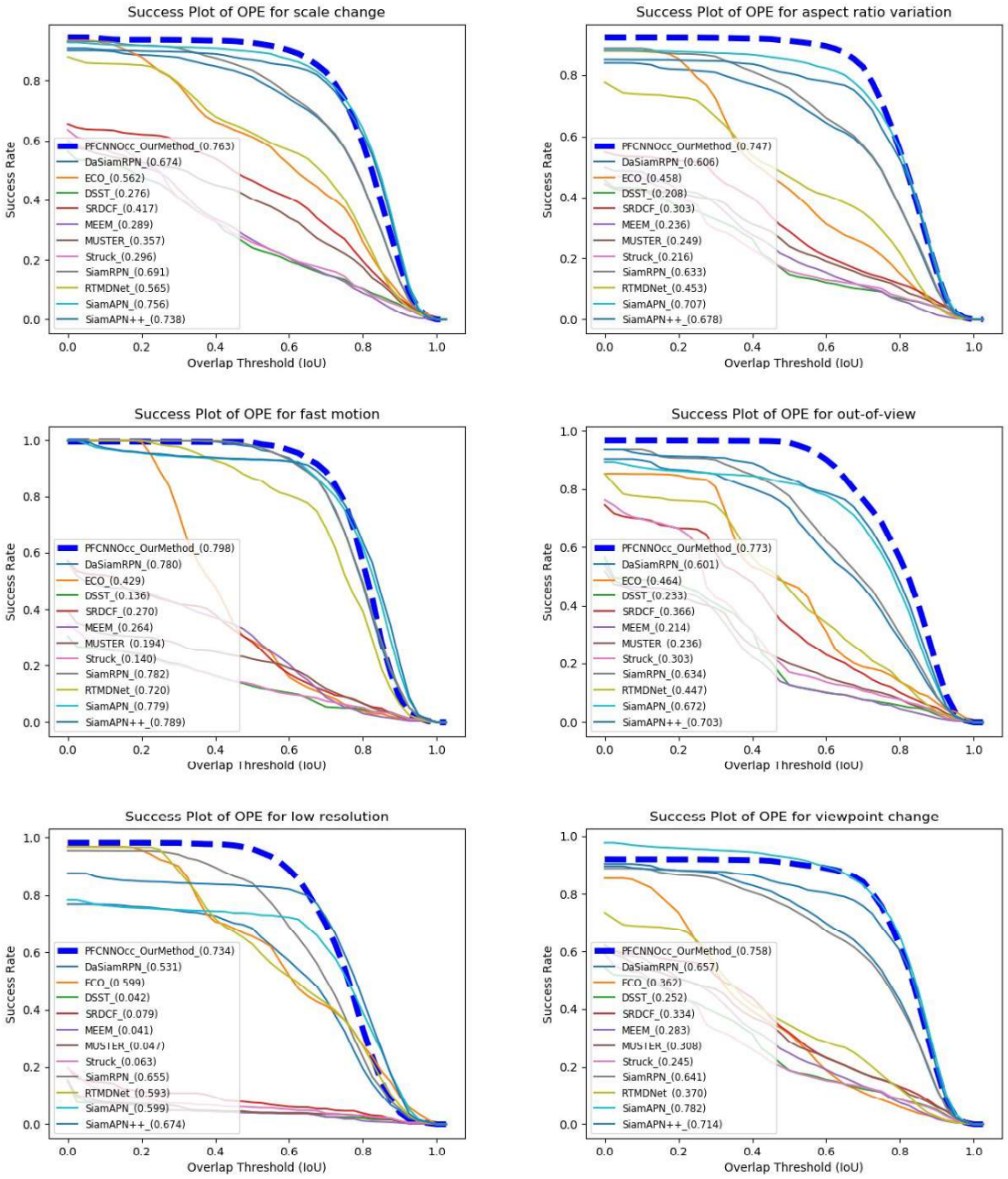


Figure 6.35. Attribute-based success plots of the SOTA trackers and the final version of PFCNNOcc.

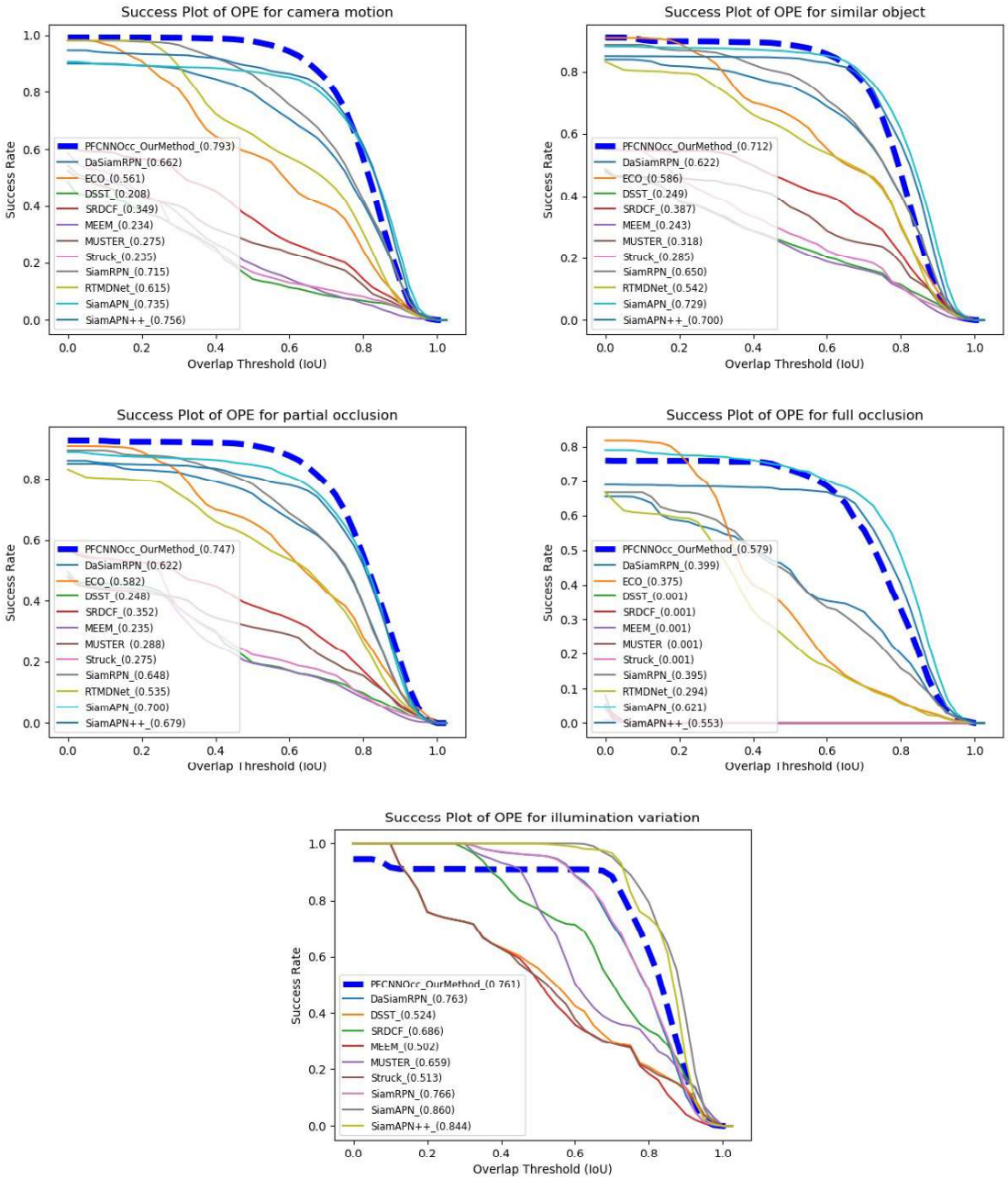


Figure 6.36. Attribute-based success plots of the state-of-the-art trackers and the final version of PFCNNOcc (cont.).

Table 6.3. Attribute-based success results of the SOTA trackers and the final version of PFCNNOcc. The best three performances are, respectively highlighted with red, green, and blue colors.

Attribute-Based Success Scores											
Trackers	PO	FO	SC	ARV	LR	FM	OOV	IV	VC	CM	SO
DaSiamRPN	0.622	0.399	0.674	0.606	0.531	0.780	0.601	0.763	0.657	0.662	0.622
ECO	0.582	0.375	0.562	0.458	0.599	0.429	0.464	-	0.362	0.561	0.586
DSST	0.248	0.001	0.276	0.208	0.042	0.136	0.233	0.524	0.252	0.208	0.249
SRDCF	0.352	0.001	0.417	0.303	0.079	0.270	0.366	0.686	0.334	0.349	0.387
MEEM	0.235	0.001	0.289	0.236	0.041	0.264	0.214	0.502	0.283	0.234	0.243
MUSTER	0.288	0.001	0.357	0.249	0.047	0.194	0.236	0.659	0.308	0.275	0.318
Struck	0.275	0.001	0.296	0.216	0.063	0.140	0.303	0.513	0.245	0.235	0.285
SiamRPN	0.648	0.395	0.691	0.633	0.655	0.782	0.634	0.766	0.641	0.715	0.650
RTMDNet	0.535	0.294	0.565	0.453	0.593	0.720	0.447	-	0.370	0.542	0.867
SiamAPN	0.700	0.621	0.756	0.707	0.599	0.779	0.672	0.860	0.782	0.735	0.729
SiamAPN++	0.679	0.553	0.738	0.678	0.674	0.789	0.703	0.844	0.714	0.756	0.700
PFCNNOcc	0.747	0.579	0.763	0.747	0.734	0.798	0.773	0.761	0.758	0.793	0.712

6.3.2.3. Scenario-Based Comparison. The final PFCNNOcc and the SOTA trackers are compared depending on their precision and success scores for each video or image sequence, listed in Table 5.1. As observed from the precision and success results, displayed in Figures 6.37-6.44, the final PFCNNOcc outperforms the SOTA trackers by far. The details are elaborated in the following paragraphs.

The final PFCNNOcc achieves the top precision scores in the scenarios, named as car1_1, car1_2, car1_3, car2, car3, car5, car6_1, car6_2, car6_3, car6_4, car6_5, car8_1, car9, car10, car16_1, car16_2, car17, car18, compared to the SOTA trackers. It means that the final PFCNNOcc outperforms or shares the same top precision scores with the SOTA trackers. Depending on the Siamese-based network used, the proposed tracker performs equally well or superior to SiamAPN [35] and SiamAPN++ [39], for 85%(18/21) of all scenarios, listed in Table 5.1. In terms of the success scores, the proposed method outperforms the SOTA in car1_3, car6_1, car6_2, car6_3, car6_5, car8_1, car9 and car16_1 that is 38%(8/21) of the videos, and, in consequence, it achieves top-3 performing scores at the 72%(15/21) of the scenarios. The very distinguishable success of the PFCNNOcc against the SOTA trackers in car6_1, car6_2, and car6_5 that include high magnitude scale change, aspect ratio variation and viewpoint change, stems from the novel "Re-Identification Module Final PFCNNOcc". While predicted flow vectors provide the correct displacement of the targets, they also contribute to the highly precise estimation of the location of the target. A precise segmentation map, predicted on a pixel basis, improves the correctness of the predicted location and, especially, the size of the target on the scene, even under severe scale and aspect ratio variations.

In car4 and car7, PFCNNOcc achieves the top-4 performing success scores. Especially, the poorest precision and success scores of the PFCNNOcc are observed only in the car7 because of an identical car that is present on the scene after a short-term full occlusion state. As soon as the tracked car is visualized on the scene again, the PFCNNOcc re-detects the car with the identical visual model of the target within a very close proximity. Therefore, while the possibility of detecting the target rather than its identical one is almost the same, the PFCNNOcc re-detects and tracks the

wrong car with identical visual features, contributing to the lowest precision and success score. When all the results of the SOTA trackers are analyzed, the PFCNNOcc outperforms most of the SOTA trackers, especially SiamAPN++ [35], SiamRPN [37], RTMDNet [25], and SRDCF [21].

In car1_2, car1_3, car2 that include partial occlusion and full occlusion, coupled with other visual problems such as scale change, aspect ratio variation, out-of-view, viewpoint change, camera motion and similar object, the novel functions of the final PFCNNOcc, "Re-Identification Module Final PFCNNOcc"and "Occlusion Detector and Calculator Final PFCNNOcc", provide precise target tracking where the target is visible on the scene. "Full Occlusion Dense Search Module Final PFCNNOcc"re-detects the target in a short-term or a long-term full occlusion state with high precision. These occlusion states result from the road signs, statues, trees, other vehicles on the scene, and the disappearance condition from the scene. This is observed especially for car1_3, one of the hardest scenarios in the UAV123 10 fps dataset, because of the long-term partial and full occlusion. In the meantime, the car undergoes serious scale and aspect ratio variations and drastic changes in its visual model. The final PFCNNOcc outperforms the state-of-the-art with a full precision score (1.000) and a success score of 0.695, which is the highest one by far. This score is lower than the success scores of other scenarios however, it is still the highest among other trackers since this scenario is a tough one for all trackers.

In car3, car4, car6_5, car8_1, car9, car10 and car16_1, the partial occlusion is coupled with the other visual problems except the full occlusion problem. "Re-Identification Module Final PFCNNOcc"and "Occlusion Detector and Calculator Final PFCNNOcc"processes successfully in segmenting and matching the cars on the scene and detecting the state and the degree of partial occlusion. From this point of view, the final PFCNNOcc performs the top precision scores in these scenarios with the partial occlusion, and is also able to achieve the highest success scores for 57%(4/7) of them.

In car16_1, car16_2, car17, and car18, the car moves non-linearly, which initiates a very fast motion, coupled with a very drastic visual model variation, viewpoint change, as well as serious scale and aspect ratio variations. This is one of the hardest scenarios. The final PFCNNOcc surpasses the SOTA trackers in these scenarios by utilizing the "Re-Identification Module Final PFCNNOcc"and the "Particle Filter"module. "Re-Identification Module Final PFCNNOcc"utilizes the predicted optical flow maps and segmentation masks, extracted from ROI sampling areas in consecutive images, with a robust matching algorithm to localize the tracked target with its size. Then the "Particle Filter Module"fine-tunes the location and size of the target at the end of the iteration. These features of "Re-Identification Module Final PFCNNOcc"and the "PF Module"are the contributions to the literature, and pave the way for the final PFCNNOcc as the most robust method in terms of the precision and success scores in these scenarios.

"Merged Target Detection "condition evolves the target tracking result in car1_2 and car10, where the target is partially occluded or comes into very close proximity with the other vehicles or similar objects, and helps the "Re-Identification Module Final PFCNNOcc"and "Occlusion Detector and Calculator Final PFCNNOcc"function to lead the final PFCNNOcc to success. In car8_2, the PFCNNOcc continues high-precision tracking under severe scale change, coupled with aspect ratio variation and rotation at the beginning of the scenario in terms of the precision and success scores. When the tracked car is merged with a bigger car, following on the back and close to the camera, and the location and size of the target, estimated, are deteriorate. This merged target condition contributes to the degradation in the precision and success scores, even though the "Merged-Target Detection "re-detects the target successfully after this condition vanishes, and the PFCNNOcc continues the tracking.

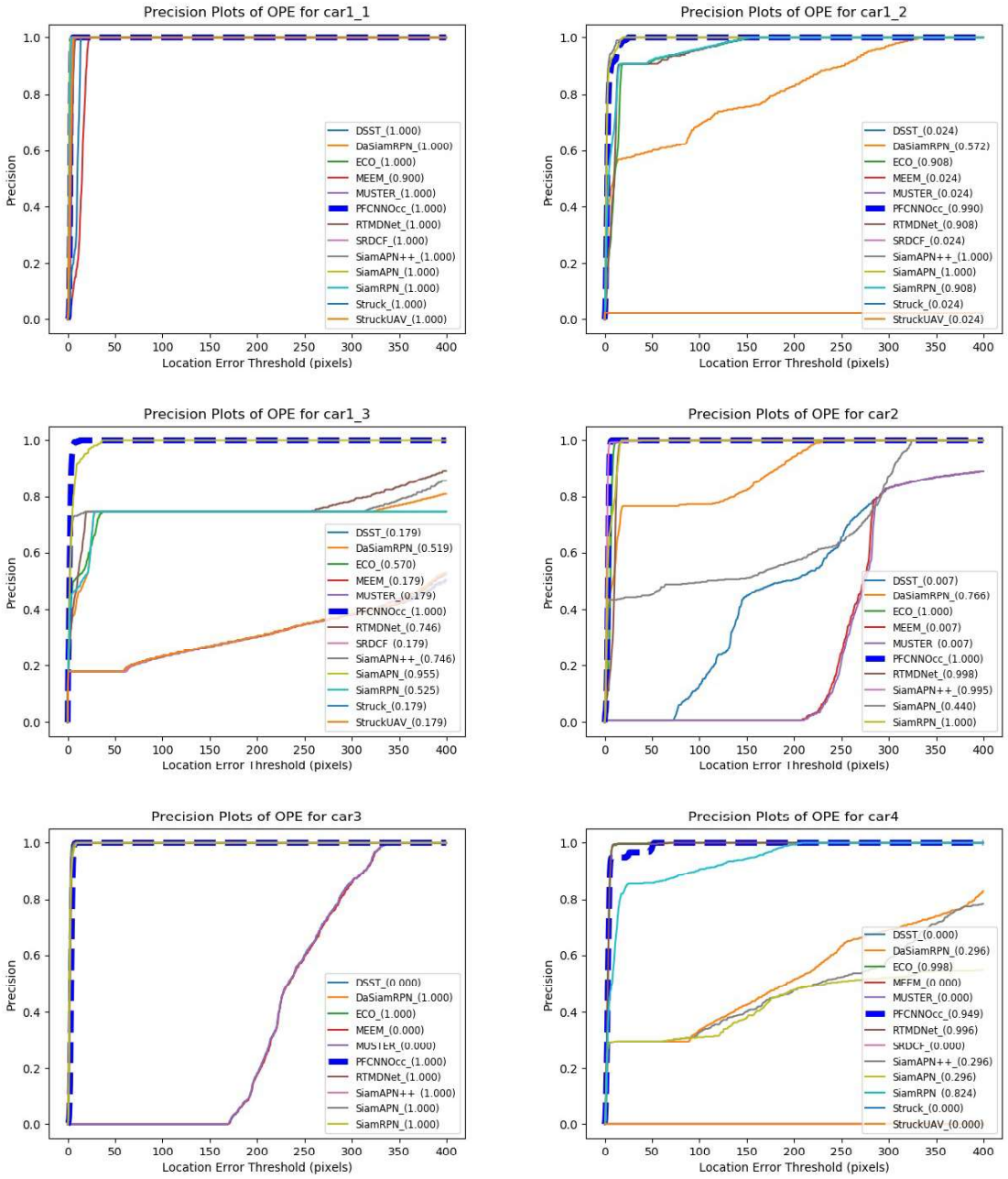


Figure 6.37. Scenario-based precision plots of the SOTA trackers and the final version of PFCNNOcc.

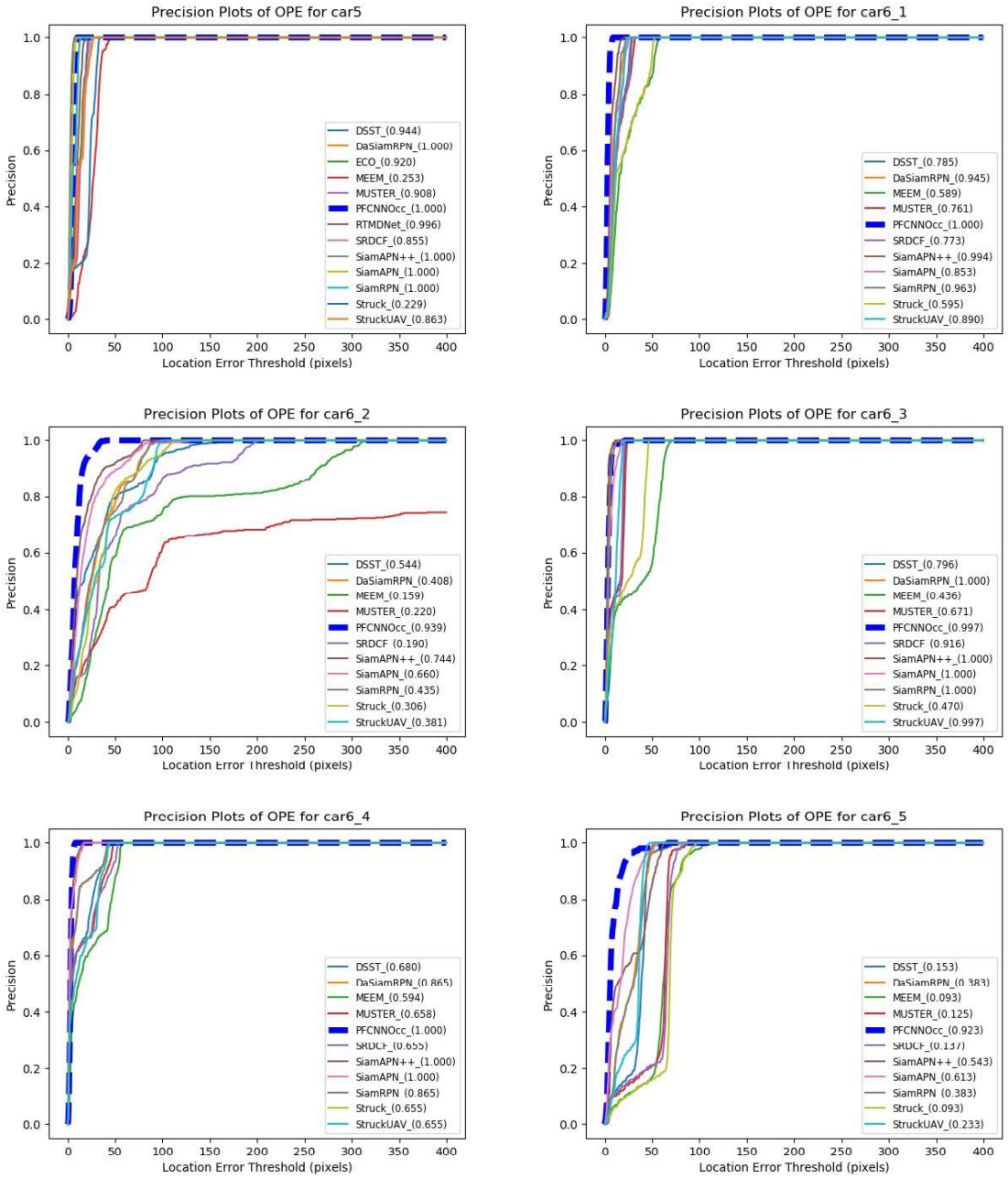


Figure 6.38. Scenario-based precision plots of the SOTA trackers and the final version of PFCNNOcc (cont.).

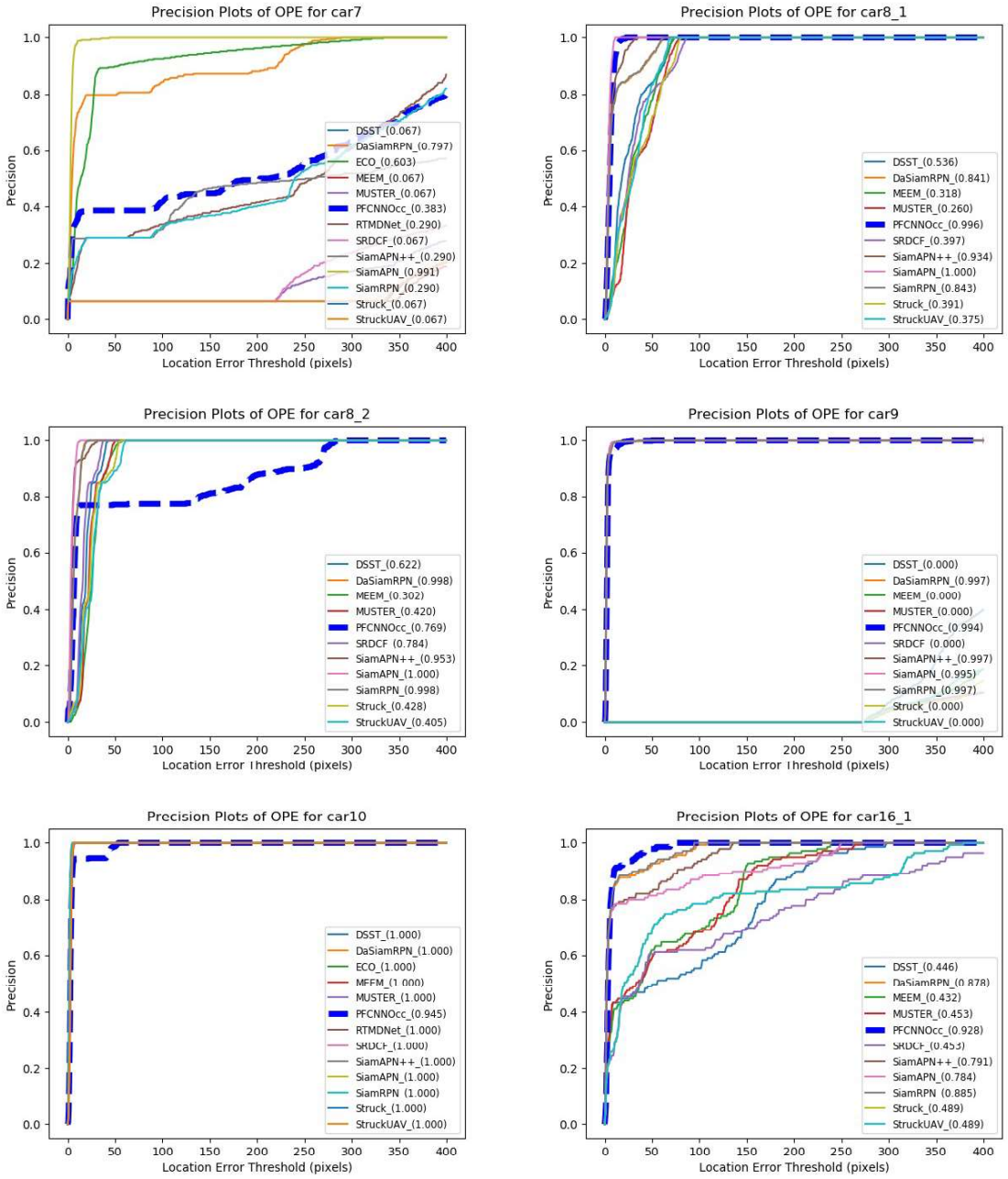


Figure 6.39. Scenario-based precision plots of the SOTA trackers and the final version of PFCNNOcc (cont.).

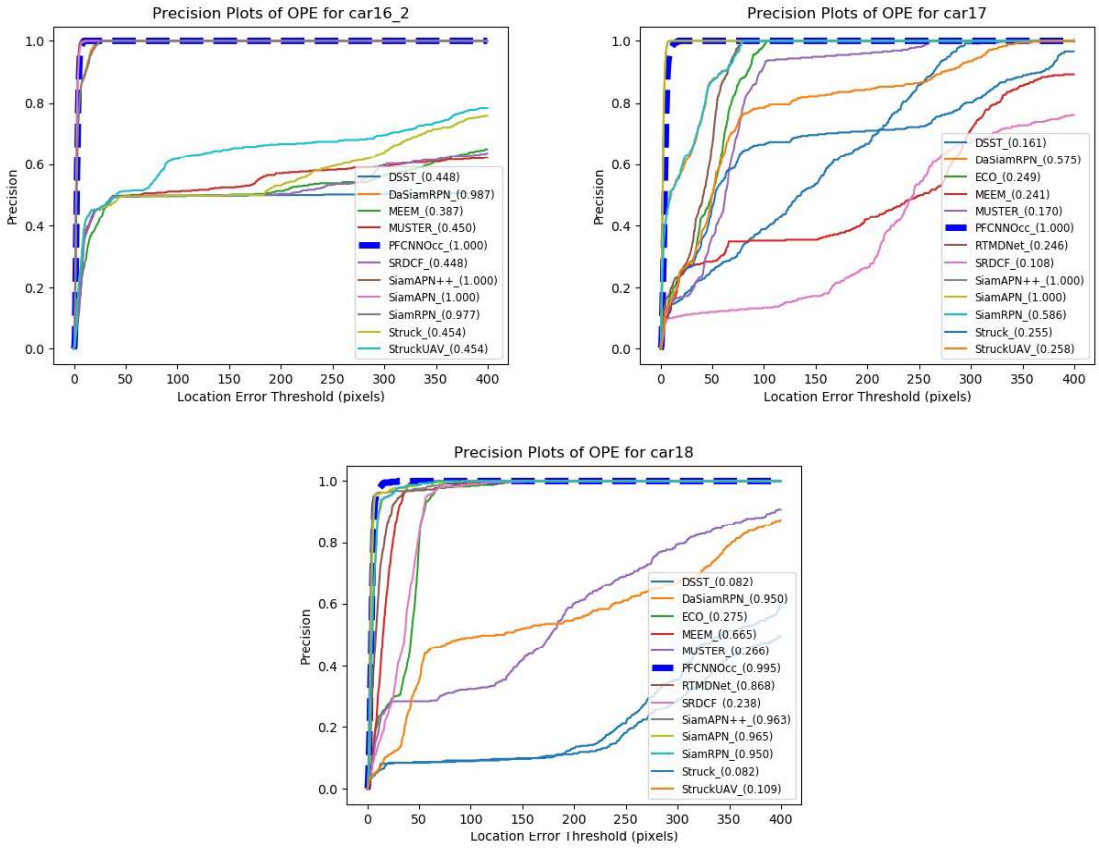


Figure 6.40. Scenario-based precision plots of the SOTA trackers and the final version of PFCNNOcc (cont.).

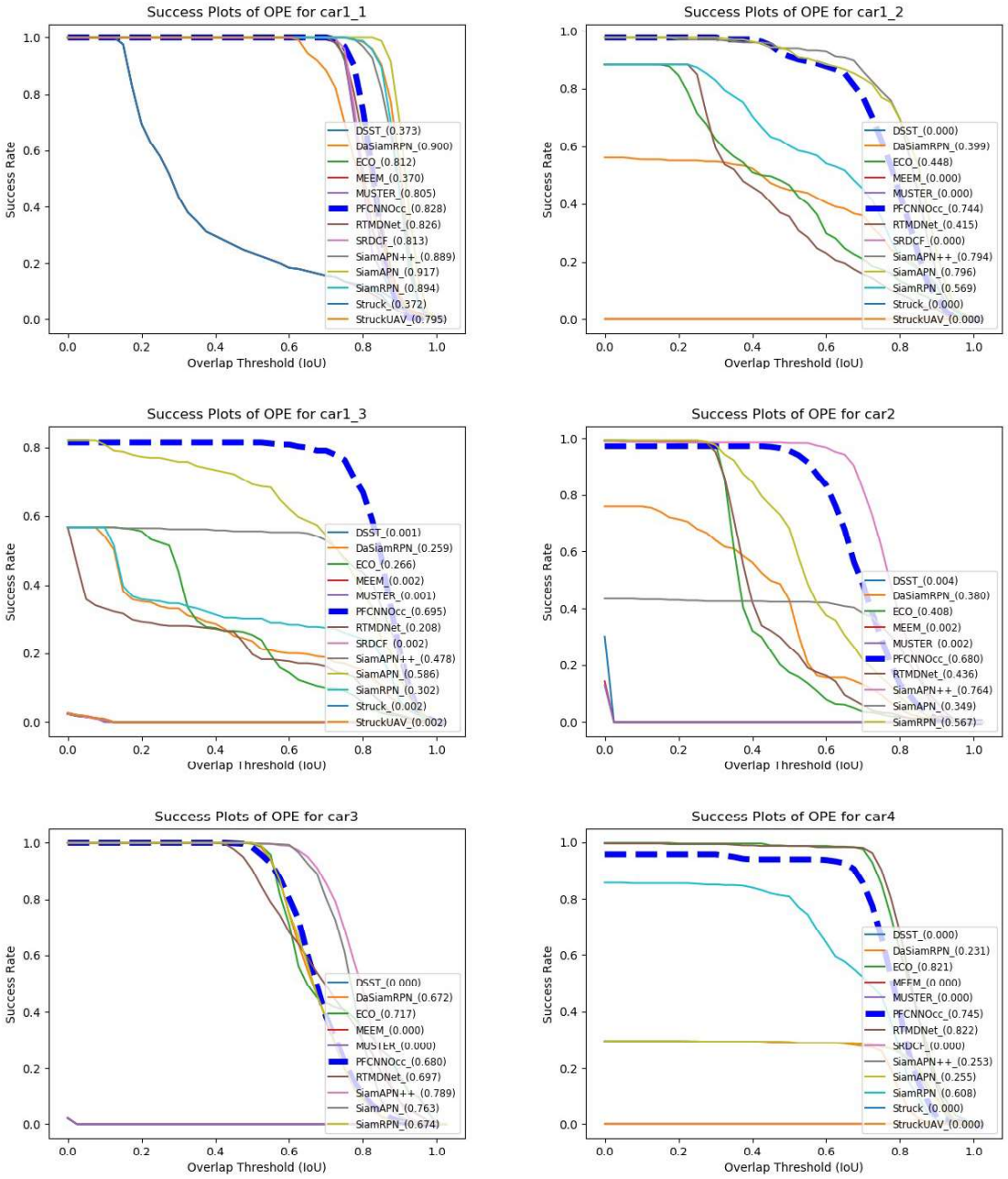


Figure 6.41. Scenario-based success plots of the SOTA trackers and the final version of PFCNNOcc.

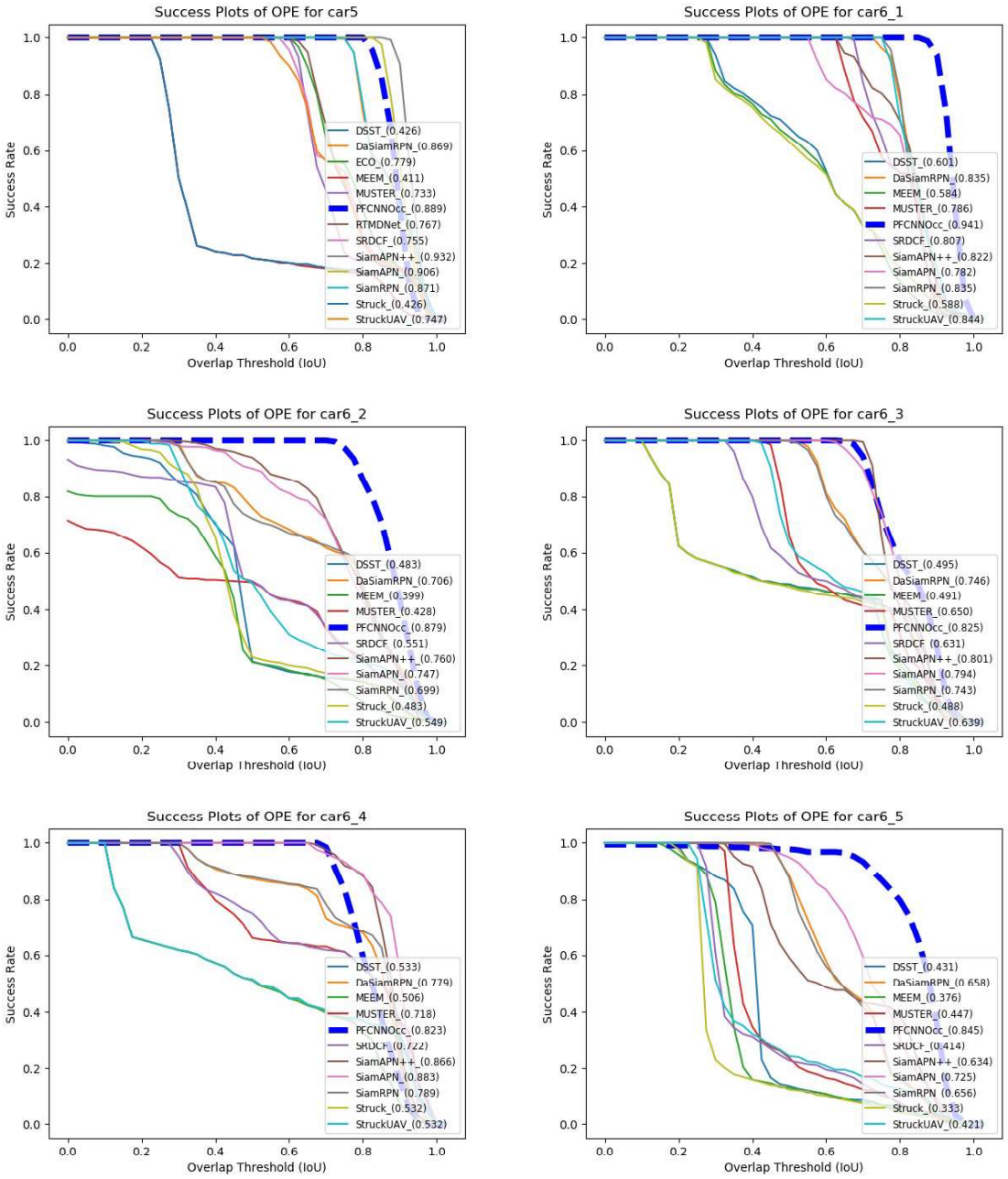


Figure 6.42. Scenario-based success plots of the state-of-the-art trackers and the final version of PFCNNOcc (cont.).

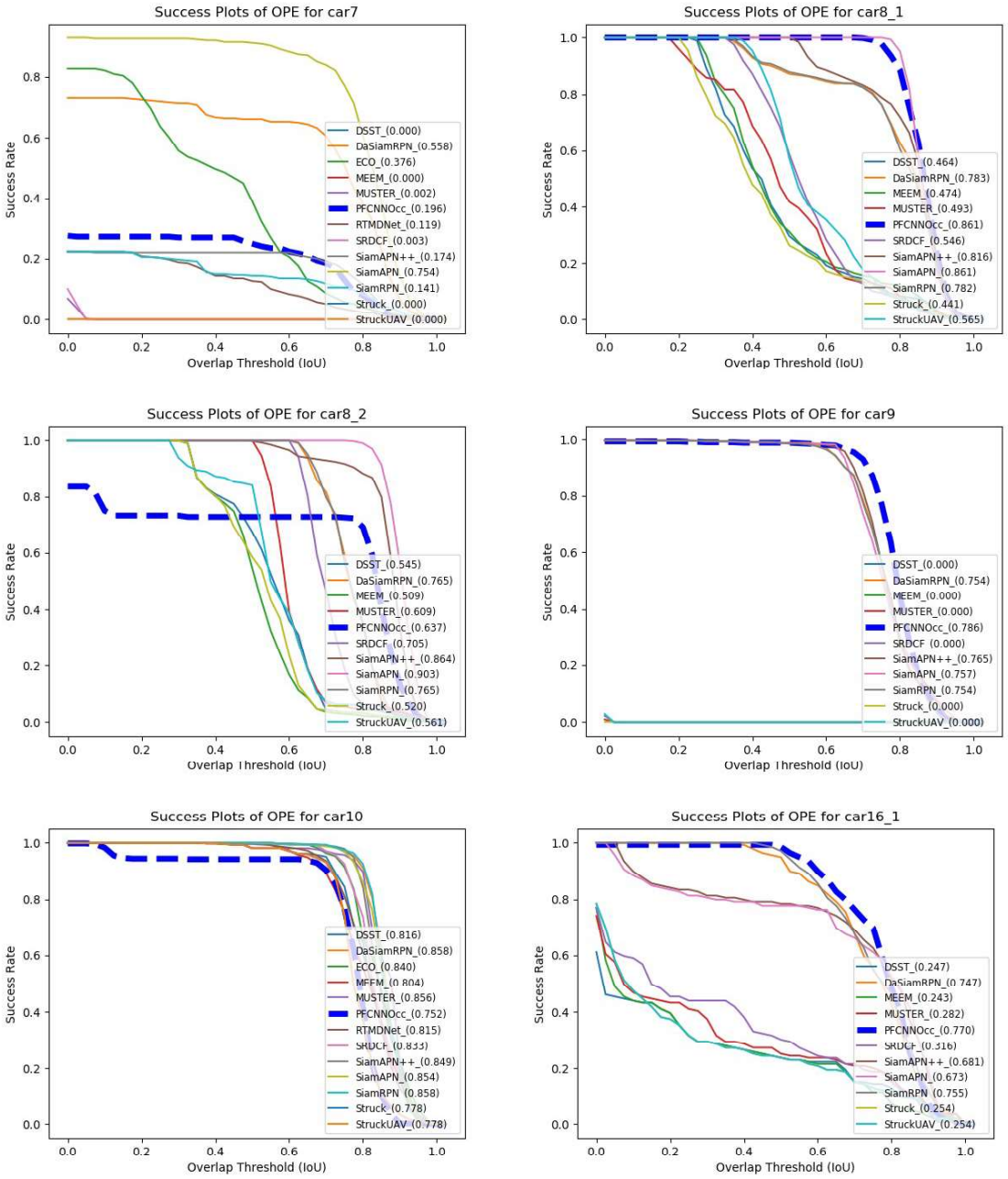


Figure 6.43. Scenario-based success plots of the SOTA trackers and the final version of PFCNNOcc (cont.).

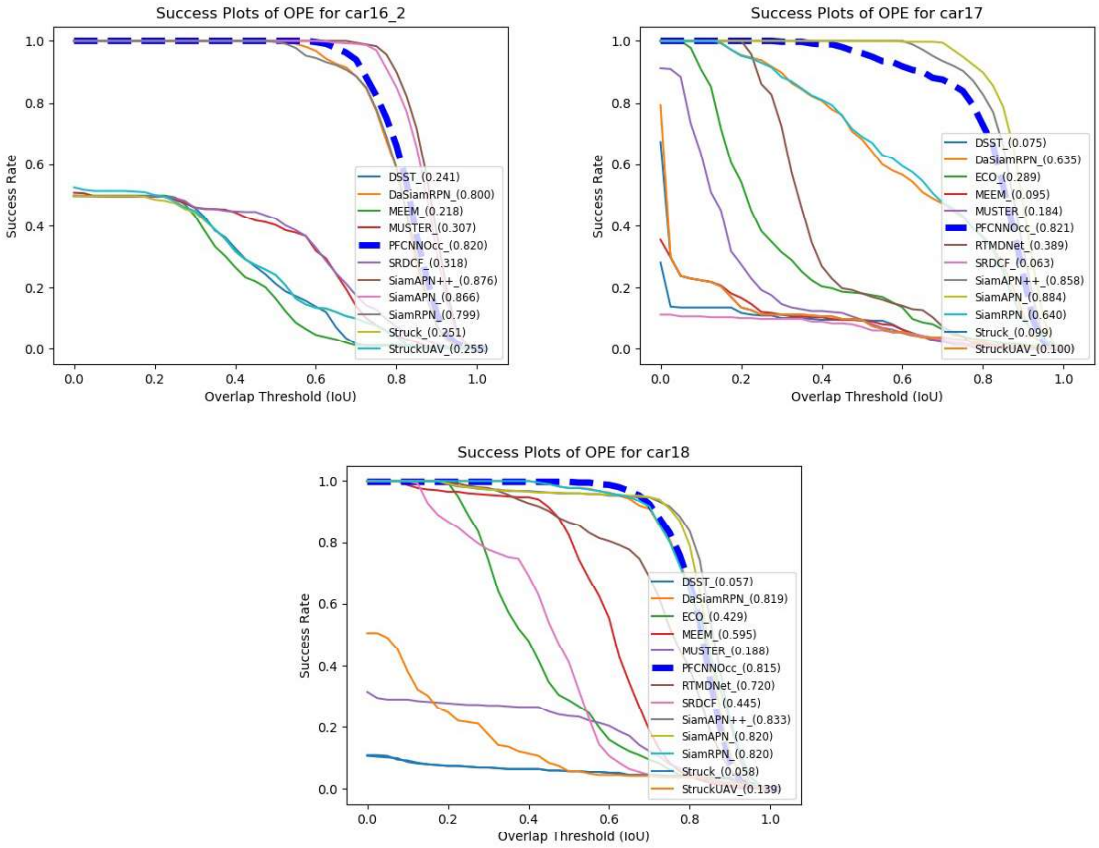


Figure 6.44. Scenario-based success plots of the SOTA trackers and the final version of PFCNNOcc (cont.).

6.3.3. Simulation Results for the Ablation Study with the Final Version of the Tracking Model on the UAV123 10fps Dataset

Simulation results of the final version of the PFCNNOcc, its ablation versions and the SOTA trackers on the UAV123 10 fps dataset [2] are extracted and evaluated in three different comparison approaches in terms of precision and success scores; 1. Overall comparison, 2. Attribute-based comparison and 3. Scenario-based comparison. Even though the precision and success results of the SOTA trackers are depicted in the plots, the main aim is to analyze and put forward the evolution of the final version of the PFCNNOcc depending on its ablation versions. In the meantime, the details about the simulation results in Section 6.3.2 are valid too for the ablation analysis. Additionally, the ablation versions of the PFCNNOcc, namely PFCNNOccWOM,

PFCNNOccWOOccCalc, and UDSCNNOcc, are annotated with the addition of “ABL ”keyword on the precision and success plots in Figures 6.45-6.55. Because of the similar scores of the methods, such as PFCNNOcc and UDSCNNOcc, the lines in plots may coincide with each other, and may become invisible. “ABL ”stands for the “ablation ”in this context. And the videos, showing the simulation results during tracking for each scenario, are presented in [100].

6.3.3.1. Overall Comparison. Overall comparison between the final version of the PFCNNOcc and its ablation versions on the UAV123 10fps dataset [2] is shown in Figure 6.45 and Table 6.4. As understood from the precision and success results, “Occlusion Calculator” and “Merged Target Detection” functions evolve the tracking performance of the PFCNNOcc step by step. PFCNNOccWOOccCalc shows the lowest tracking performance because of the lack of the “Occlusion Calculator” function, which is responsible for calculating the degree of overall occlusion during tracking. Without updating the degree of overall occlusion, the PFCNNOccWOOccCalc is unaware of the occlusion transitions. When the target is under full occlusion, the PFCNNOcc-WOOccCalc is not able to switch to the “Full Occlusion Dense Search Module Final PFCNNOcc” function to search for the occluded target. Another weakness of this ablation version is updating the reference feature of the target under the partial occlusion, which disturbs the reference feature tailored during the re-identification and tracking processes. The integration of the “Occlusion Calculator” function boosts the precision and success scores of the PFCNNOcc significantly.

The importance of the “Merged Target Detection” function is easily discerned from the performance differences between the PFCNNOcc and the PFCNNOccWOM. “Merged Target Detection” provides the PFCNNOcc with situational awareness against another vehicle, and that can merge with a tracked target and a capability to discriminate the target from a newly emerged vehicle after the merged condition vanishes. The PFCNNOcc gains a very vital performance evolution that drives it to the top of the SOTA with the invention of the “Merged Target Detection” function.

While the PFCNNOcc outperforms the UDSCNNOcc and the SOTA trackers by the integration of the "Particle Filter" module to fine-tune the rough estimate of the target state variables in terms of the overall precision score, the uniform dense search function of the UDSCNNOcc provides a slightly higher success score compared to that of the PFCNNOcc. These results prove that while the PFCNNOcc provides more precise location estimates of the bounding boxes for the target, the UDSCNNOcc has better estimates for the size of the target, depending on the success results. It is thought that the performance difference between the UDSCNNOcc and the PFCNNOcc in terms of the success scores occurs because of the insufficient number of particles, tailored by the PFCNNOcc to sample the region around the rough estimate location of the target. The performance of the PFCNNOcc may increase if the number of particles is increased. Modeling the process equation with a non-linear model instead of a random walk model can boost the performance of the PFCNNOcc advantageously.

Table 6.4. Overall performance of the final version of PFCNNOcc and the ablation versions. The best three performances are highlighted with red, green, and blue colors.

Ablation Versions	Precision Score	Success Score
PFCNNOccWOOccCalc	0.723	0.588
PFCNNOccWOM	0.903	0.730
UDSCNNOcc	0.942	0.782
PFCNNOcc	0.943	0.763

6.3.3.2. Attribute-Based Comparison. Attribute-based precision and success scores of the PFCNNOcc and its ablation versions against visual problems, such as, SC, ARV, FM, OOV, LR, VC, CM, SO, PO, FO and IV are presented in Figures 6.46-6.49 and Tables 6.5 and 6.6. The observations and analysis about the ablation versions of the PFCNNOcc in Section 6.3.3.1 are also valid for the attribute-based comparison. The integration of the "Occlusion Calculator" "Merged Target Detection" and the "Particle Filter" module into the PFCNNOcc increases the performance considerably in terms of

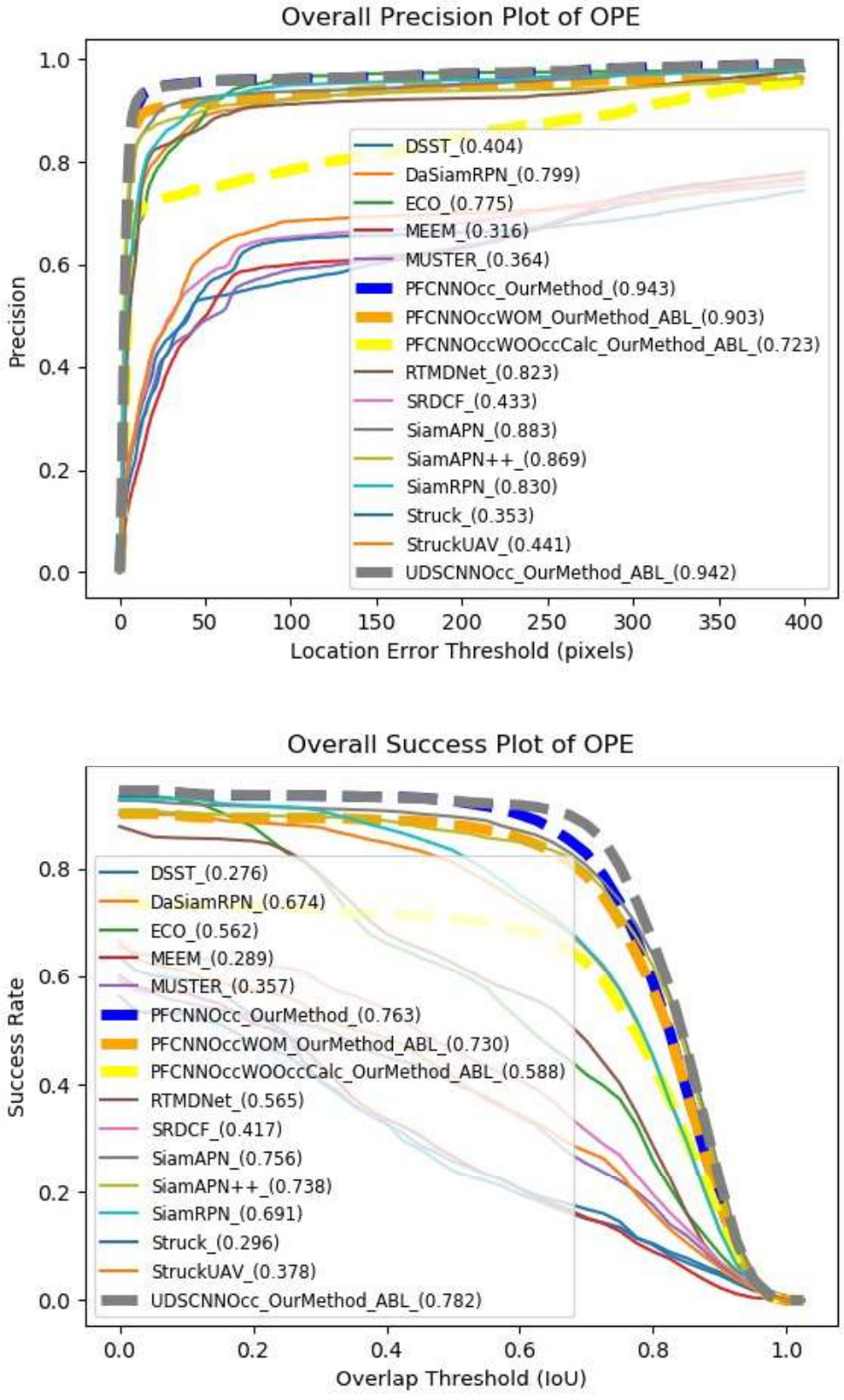


Figure 6.45. Overall precision and success plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers.

the precision and success scores. Especially for the full occlusion problem, the "PFCNNOccWOOccCalc" shows a moderate precision and success scores because of the lack of "Occlusion Calculator" providing the updated calculation of the degree of overall occlusion. Being unaware of the actual value of the degree of the overall occlusion leads to the incorrect timing for the update of the reference feature during all iterations of the tracking. The misleading reference feature creates poor performance in terms of precision and success scores. The "PFCNNOccWOM" without "Merged Target Detection" function faces performance degradation in terms of precision and success scores against all visual problems. "Merged Target Detection" creates a great deal of contribution to the tracking performance of the PFCNNOcc, especially for videos with similar objects, partial occlusion, and full occlusion. While the PFCNNOcc outperforms the UDSCNNOcc and the SOTA trackers by the integration of the "Particle Filter" module in terms of the precision score, the uniform dense search function of the UDSCNNOcc provides a slightly higher success score compared to that of the PFCNNOcc. It is thought that the performance difference between the UDSCNNOcc and the PFCNNOcc in terms of the success scores occurs because of the insufficient number of particles, tailored by the PFCNNOcc to sample the region around the rough estimate location of the target. The performance of the PFCNNOcc may increase if the number of particles is increased.

6.3.3.3. Scenario-Based Comparison. Success-based precision and success scores of the PFCNNOcc and its ablation versions are presented in Figures 6.37-6.40. As clearly seen from the precision and success results, "Occlusion Calculator" and "Merged Target Detection" functions evolve the tracking performance of the PFCNNOcc step by step for almost all scenarios. In the scenarios, such as car1_2, car1_3, car2, car4, car7, car9, in which the full occlusion exists, PFCNNOccWOOccCalc shows the lowest tracking performance because of the lack of the "Occlusion Calculator" function. When the target is under full occlusion, the PFCNNOccWOOccCalc is not able to switch to the "Full Occlusion Dense Search Module Final PFCNNOcc" function to search for the occluded target. Another weakness of this ablation version is updating the reference feature of the target under the partial occlusion, which disturbs the reference feature.

In car1_2, car8_2, and car10, the tracked target is merged by other vehicles partially or fully, and the absence of the "Merged Target Detection" in the PFCNNOcc-WOM leads to a performance degradation in terms of the precision and success scores. The PFCNNOcc gains a very vital performance evolution that drives it to the top of the SOTA with the invention of the "Merged Target Detection" function. While the PFCNNOcc outperforms the UDSCNNOcc and the SOTA trackers by the integration of the "Particle Filter" module to fine-tune the rough estimate of the target state variables in terms of the precision scores, the uniform dense search function of the UDSCNNOcc provides a slightly higher success score compared to that of the PFCNNOcc. These results prove that while the PFCNNOcc provides more precise location estimates of the bounding boxes for the target, the UDSCNNOcc has better estimates for the size of the target, depending on the success results.

Table 6.5. Attribute-based precision results of the final version of PFCNNOcc and ablation versions. The best three performances are highlighted in red, green, and blue colors.

Attribute-Based Precision Scores											
Ablation Versions	PO	FO	SC	ARV	LR	FM	OOV	IV	VC	CM	SO
PFCNNOccWOOccCalc	0.692	0.514	0.723	0.650	0.608	0.670	0.703	0.951	0.666	0.664	0.754
PFCNNOccWOM	0.872	0.740	0.903	0.898	0.933	0.952	0.943	0.921	0.898	0.964	0.857
UDSCNNOcc	0.930	0.834	0.942	0.933	0.977	0.974	0.970	0.927	0.925	0.979	0.924
PFCNNOcc	0.932	0.843	0.943	0.936	0.977	0.979	0.969	0.922	0.929	0.979	0.925

Table 6.6. Attribute-based success results of the final version of PFCNNOcc and ablation versions. The best three performances are highlighted in red, green, and blue colors.

Attribute-Based Success Scores											
Ablation Versions	PO	FO	SC	ARV	LR	FM	OOV	IV	VC	CM	SO
PFCNNOccWOOccCalc	0.554	0.342	0.588	0.519	0.455	0.559	0.556	0.808	0.539	0.537	0.583
PFCNNOccWOM	0.698	0.488	0.730	0.715	0.704	0.772	0.746	0.761	0.729	0.781	0.657
UDSCNNOcc	0.770	0.602	0.782	0.759	0.782	0.800	0.814	0.779	0.760	0.818	0.739
PFCNNOcc	0.747	0.579	0.763	0.747	0.734	0.798	0.773	0.761	0.758	0.793	0.712

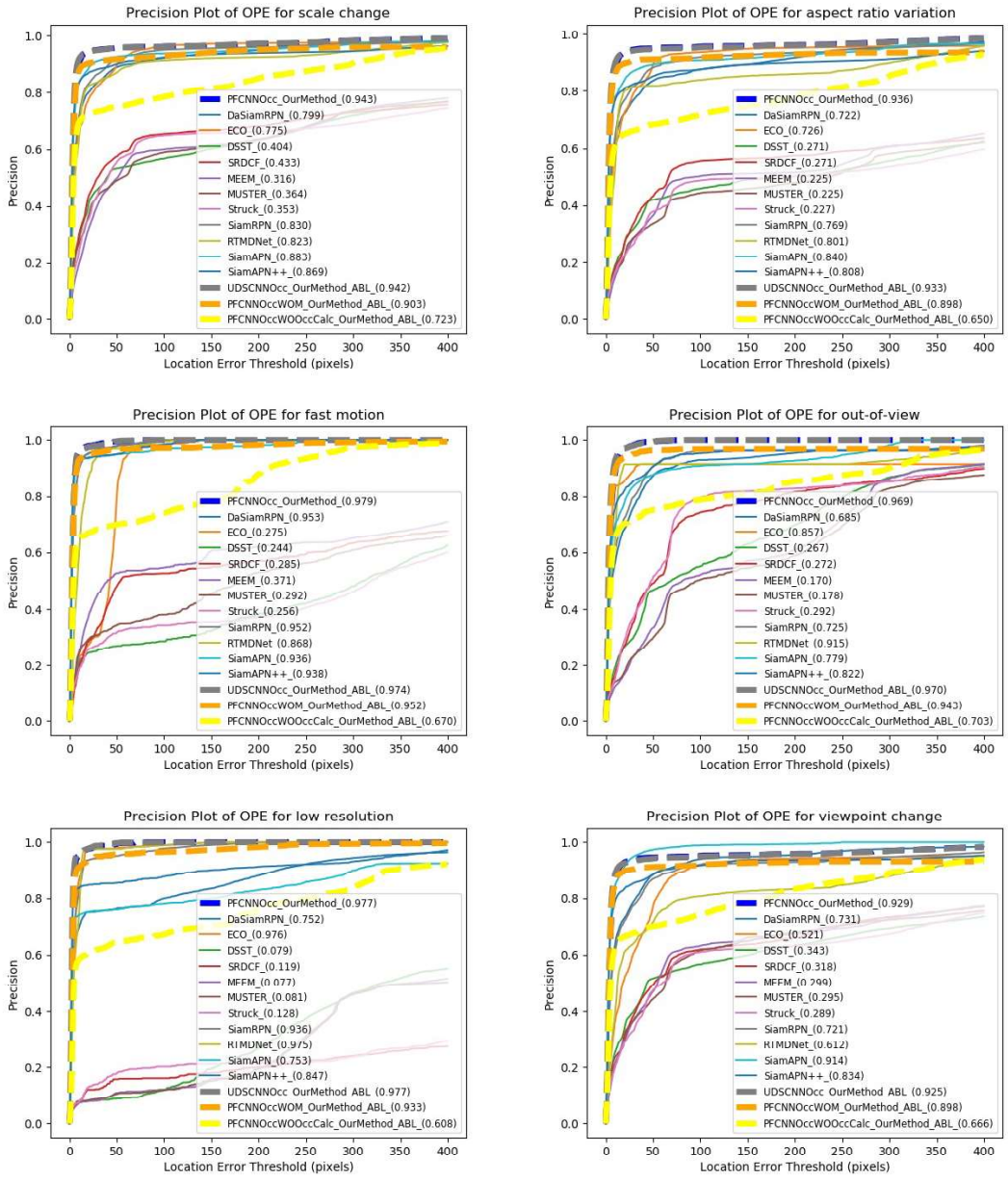


Figure 6.46. Attribute-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers.

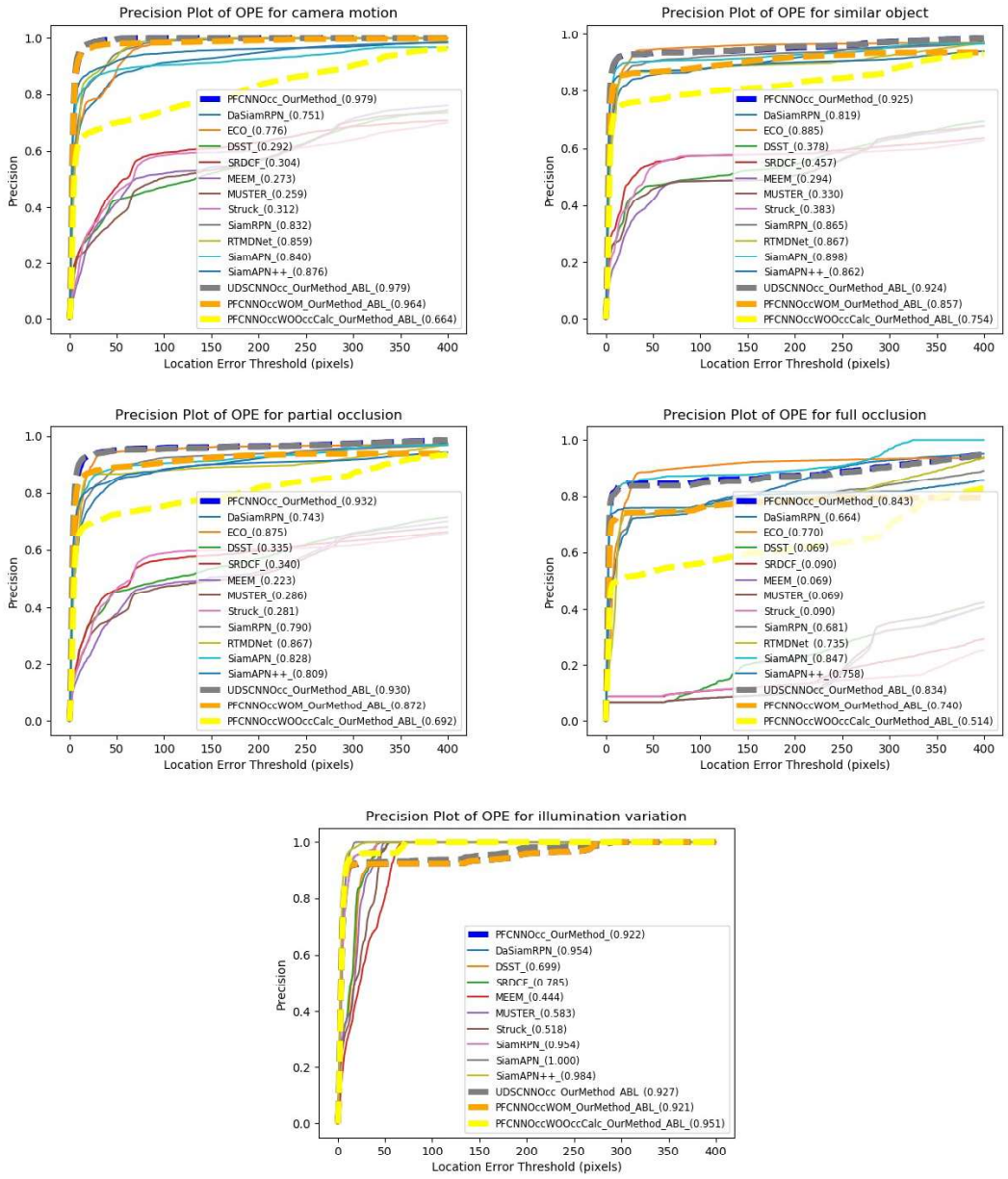


Figure 6.47. Attribute-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.).

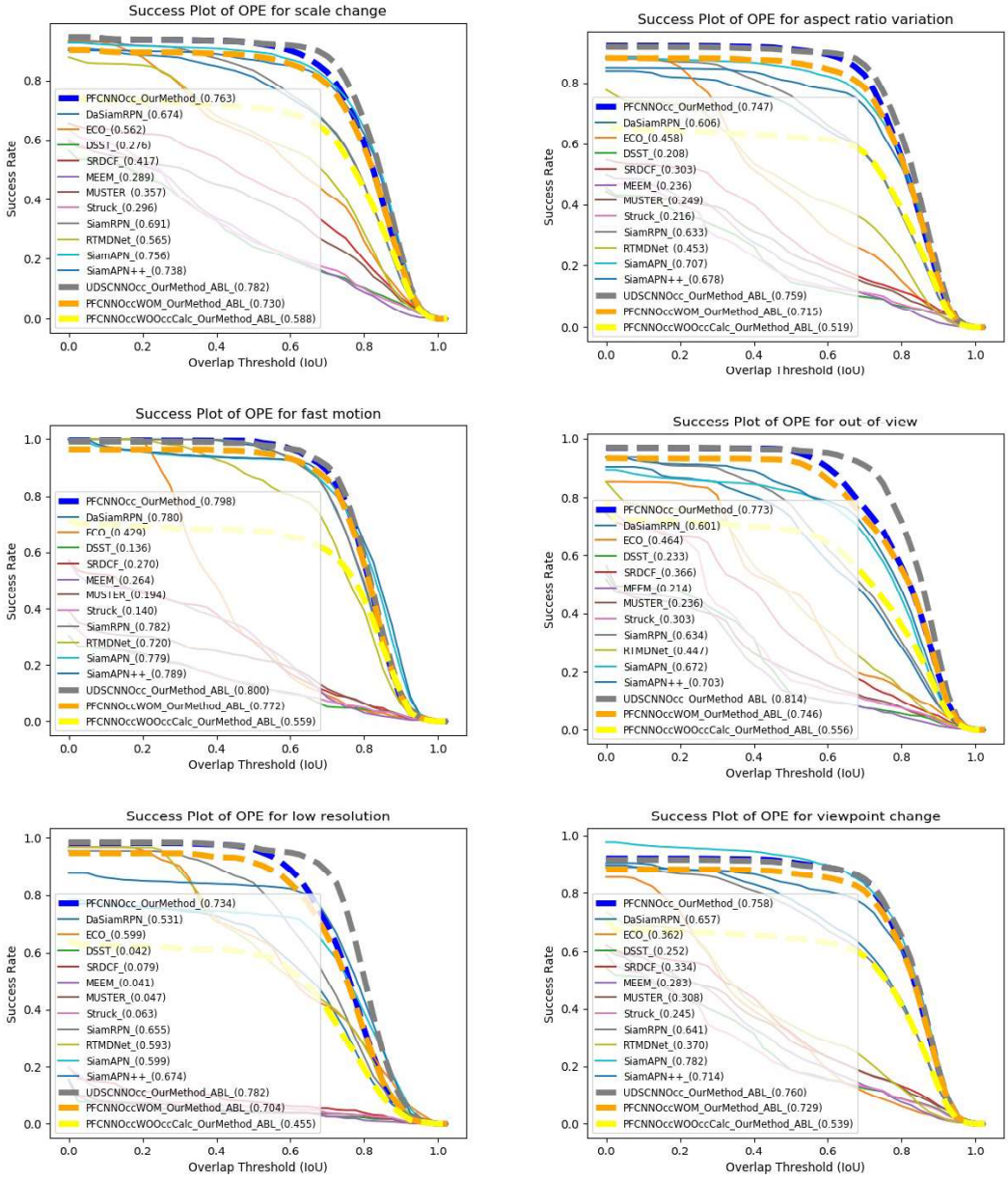


Figure 6.48. Attribute-based success plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers.

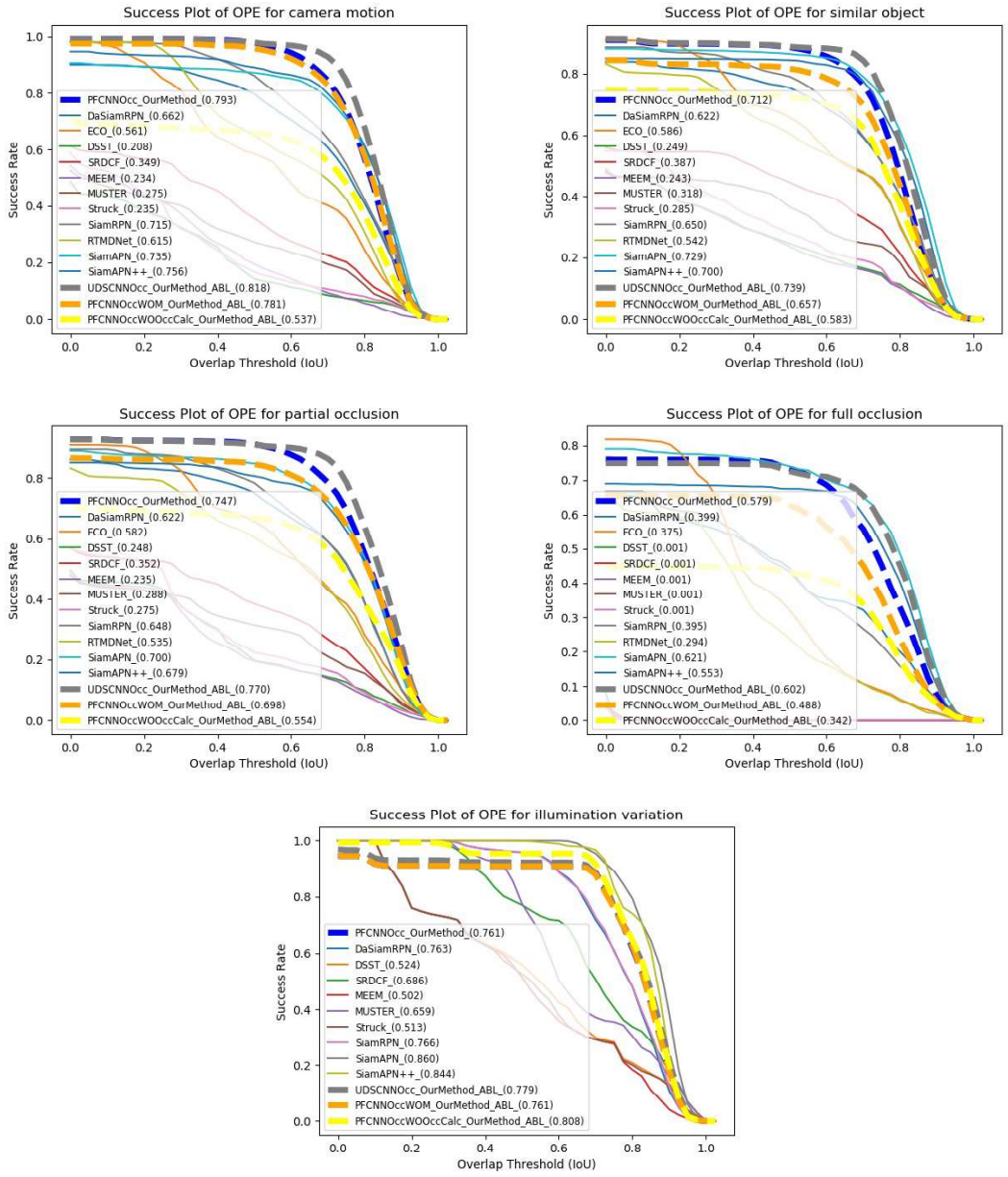


Figure 6.49. Attribute-based success plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.).

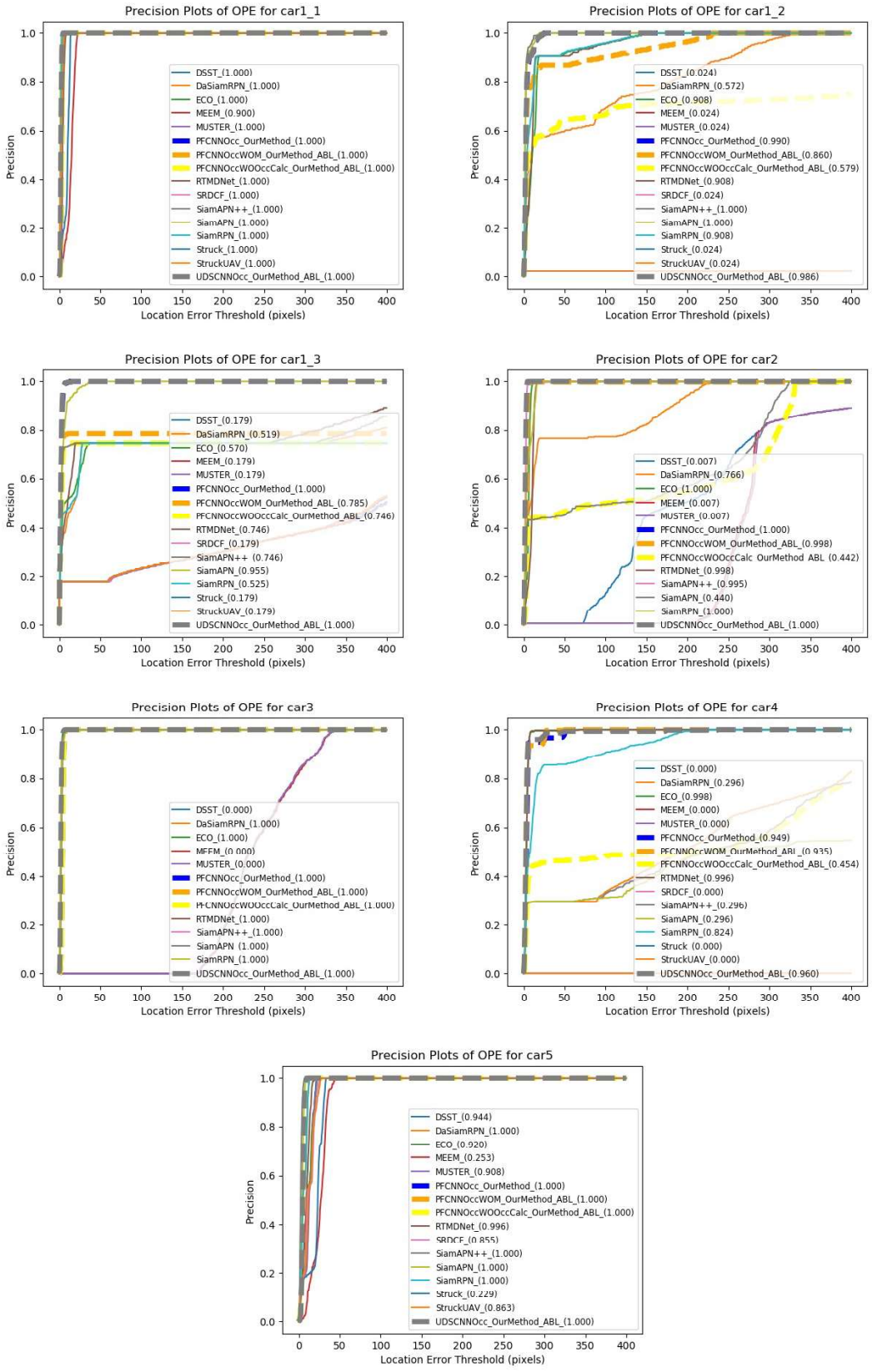


Figure 6.50. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers.

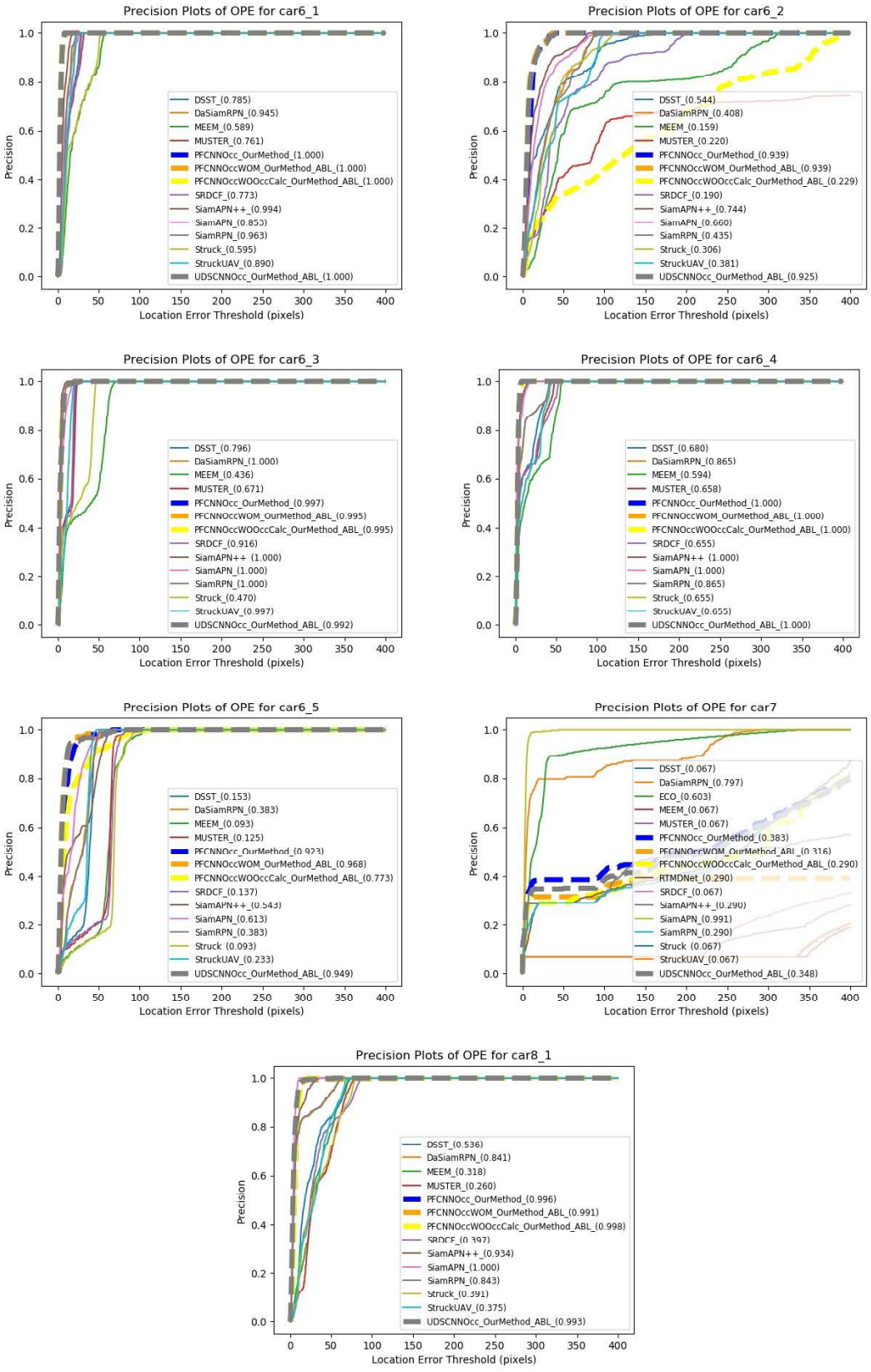


Figure 6.51. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.).

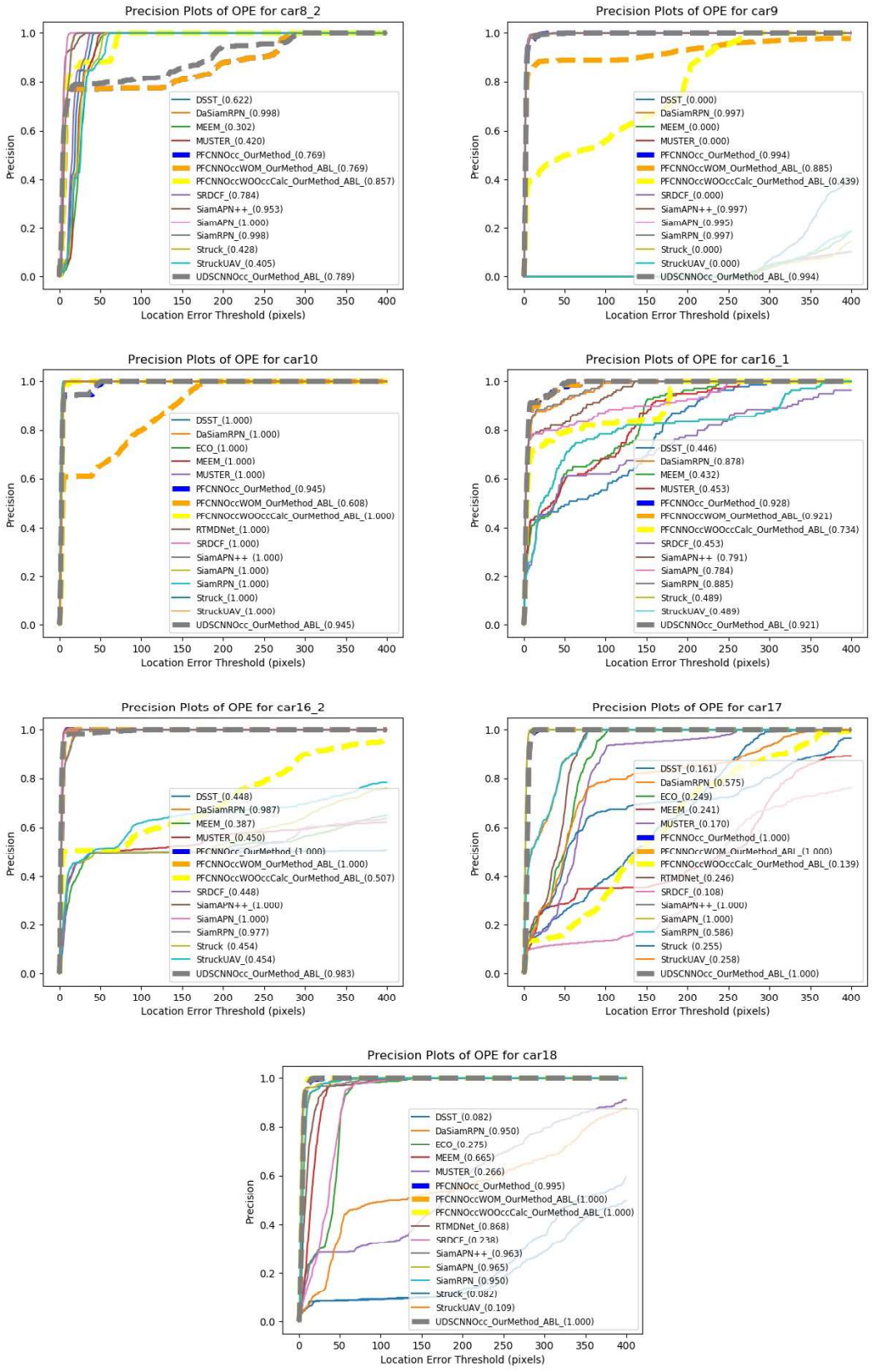


Figure 6.52. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.).

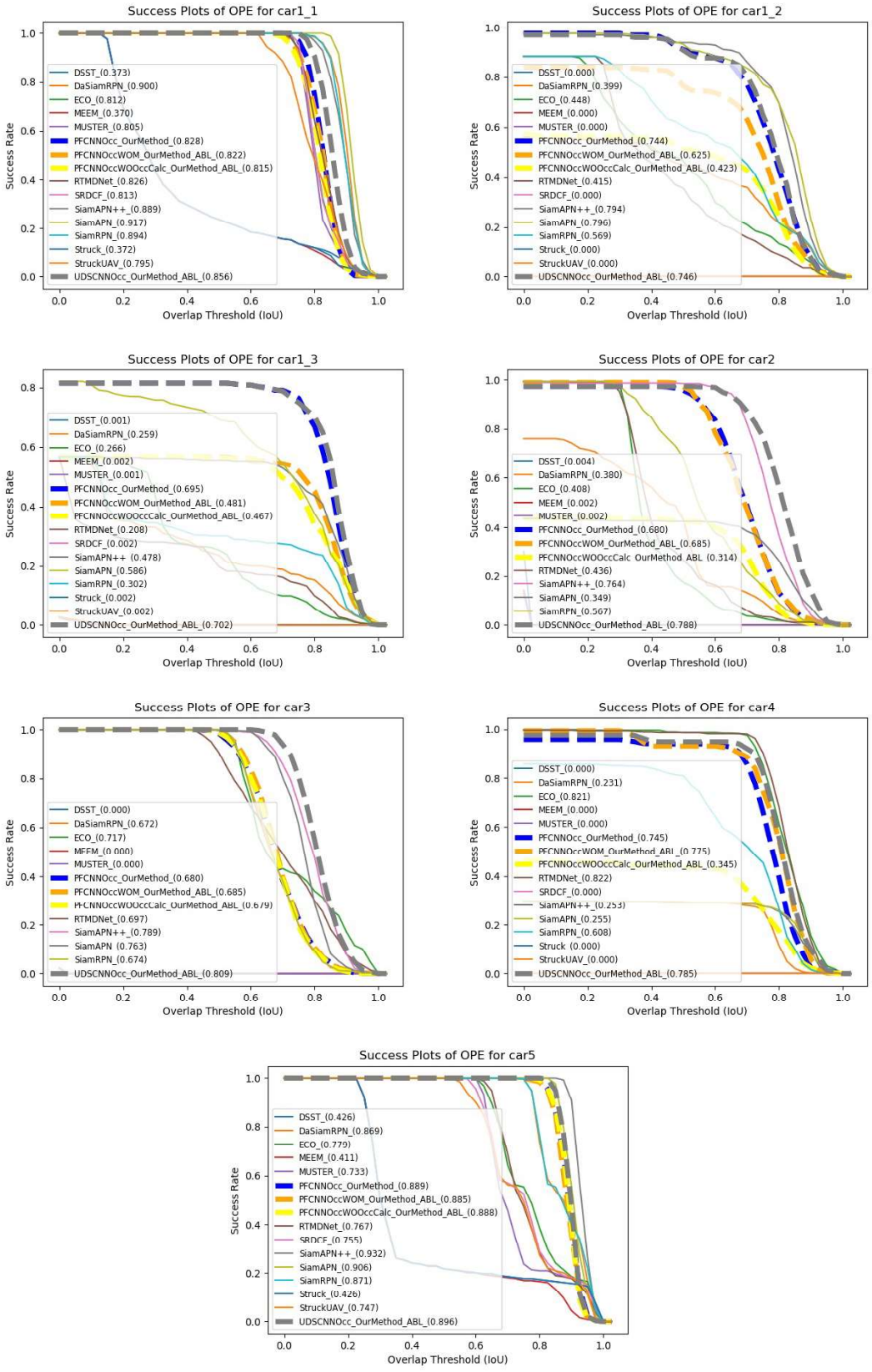


Figure 6.53. Scenario-based success plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers.

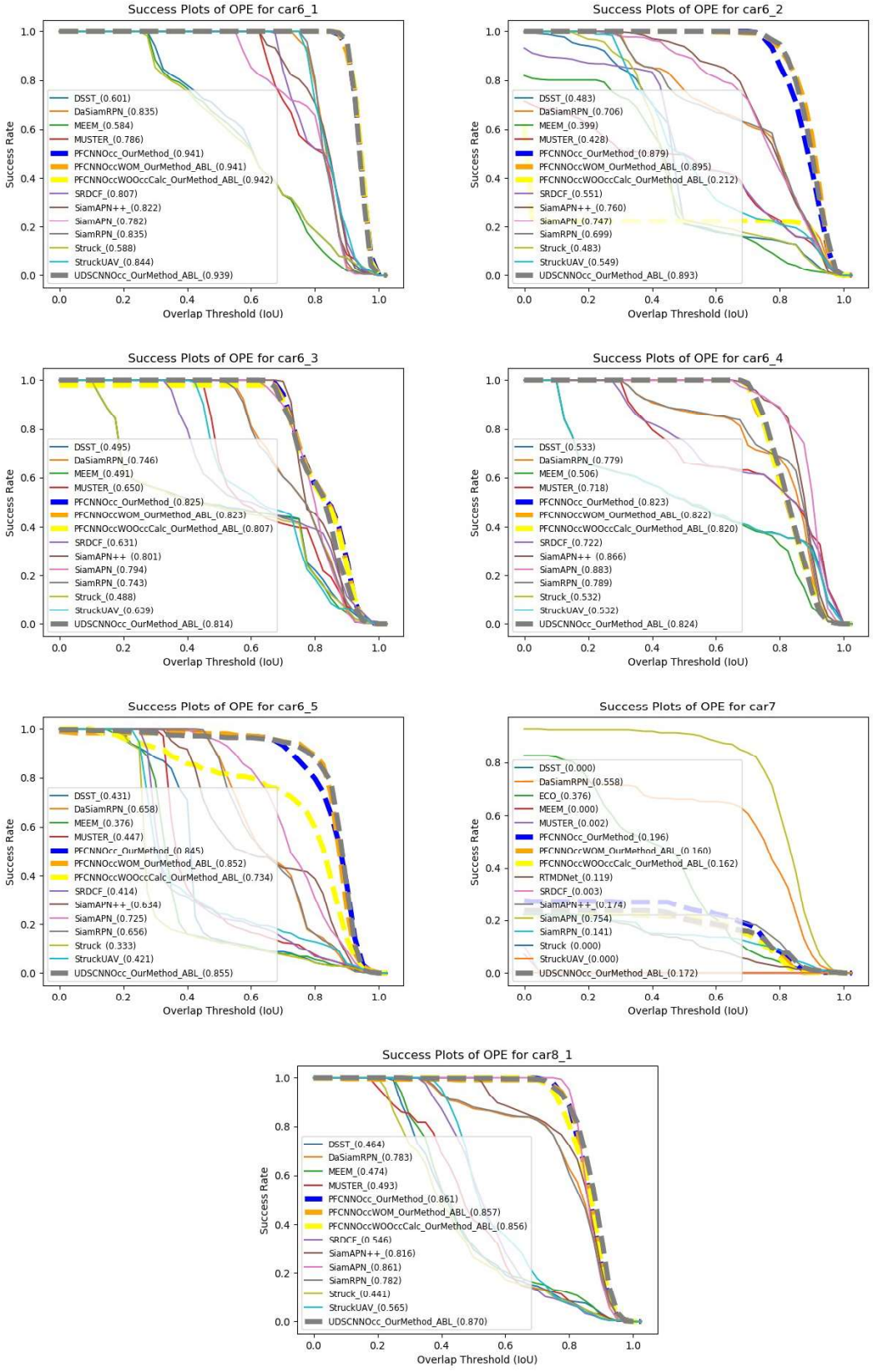


Figure 6.54. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.).

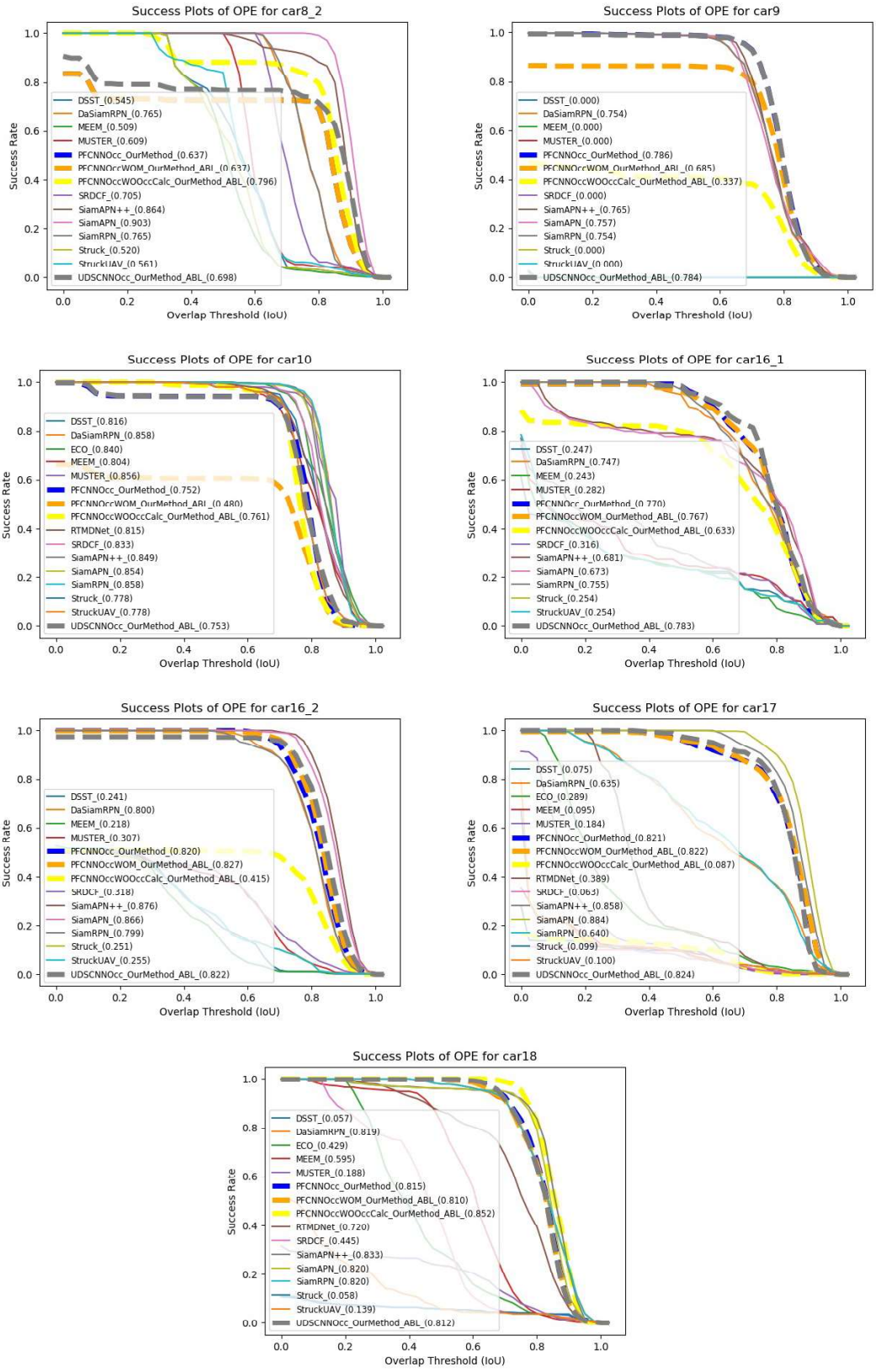


Figure 6.55. Scenario-based precision plots of the final version of PFCNNOcc, the ablation versions, and the SOTA trackers (cont.).

7. CONCLUSIONS AND FUTURE WORK

In this thesis, the goal is to design a method that can track a vehicle against all visual problems, especially occlusion, in videos from UAVs flying at low altitudes. The complexity of the method differs based on the number of visual problems present in a video. A road map is drawn with two steps to create a robust method. The first step is to design a method that can track vehicles against visual problems, except occlusion. This method is called PFCNN. The next and final step is to design PFCNNOcc, which is capable of tracking a vehicle under all visual problems, especially partial and full occlusion, which is not addressed by the PFCNN.

The PFCNN focuses on resolving abrupt changes in scale, aspect ratio, visual model, target motion, and rotation for vehicle tracking in low-altitude UAV videos. The PFCNN, which combines a CNN and a particle filter, is robust to rotation, scale variation, changes in aspect ratio, fast motion, camera motion, viewpoint changes, and the existence of similar objects in the image. The PFCNN makes the following three contributions:

- A mechanism for rescaling particles against changes in scale and aspect ratio is developed. Tracking under scale and aspect ratio variations is enhanced by implementing a particle rescaling technique in which each particle is resized with a different aspect ratio.
- A new structure that mimics the APF [7–9] is obtained by completely integrating the CNN model with the particle filter. State variable estimation is unquestionably improved by supporting the weight update process of the particle filter with their class probability scores, computed by a CNN.
- The unified multi-part vehicle tracking framework is created, where several tracking agents can function independently and produce estimates of bounding boxes during the particle sampling phase. During the estimation stage of particle filtering, the individual estimates for the parts are aggregated to get a final bounding

box estimate for the whole vehicle. The multi-part tracking framework is individually applied to different parts of the vehicle to be able to keep track of the diverse movement of the vehicle and to improve tracking performance against aspect ratio variation, abrupt rotational changes, and fast motion issues. Because of this, it is simpler to spot sudden variations in scale, aspect ratio, and rotation for parts than to spot them for the entire vehicle.

For the time being, PFCNN outperforms all benchmark trackers in [4], [12, 13], and shares the top-2 ranks with the DL based methods such as DaSiamRPN [38].

While PFCNN and APF mimic the sampling scheme of the particle filter that uses the optimal importance density, PFCNN outperforms APF when used for target tracking under challenging scenarios, and when target and camera motion are nonlinear with high dynamics. Hence, the contribution of the PFCNN is provided by utilizing the class probability term in the weight update process. The use of the class probability, calculated by CNN, is the novelty of the structure, and the key point that paves the way to its success.

On the other hand, the PFCNNOcc method is designed to predict optical flow and occlusion simultaneously. In order to utilize and train the CNN model in PFCNNOcc, large-scale UAV datasets that include a rich diversity of objects and encompass various weather conditions and landscapes are required. Currently, neither a synthetic nor a real-life dataset exists in the literature that includes the ground truth for the optical flow, occlusion, and segmentation information obtained from UAVs. Hence, a synthetic dataset, named as S-UAV dataset, containing optical flow, occlusion, and segmentation reference information of vehicles from the perspective of a UAV has been created for the first time. The dataset is generated using the Unity game engine, thereby filling the existing gap in the literature. The S-UAV dataset is very vital in the training of the PFCNNOcc, and plays a very important role in the success of the PFCNNOcc during target tracking. It is an important contribution and useful to the literature. In the meantime, the segmentation of the vehicles, especially cars, in the UAV123 dataset [2],

is achieved to create the ground truth data. This segmentation dataset is used to fine-tune the segmentation model in the final version of the PFCNNOcc and boosts its tracking performance.

The PFCNNOcc is designed to achieve a robust target tracking in videos from UAVs flying at low altitudes against all visual problems, especially against short or long-term partial and full occlusion coupled with aspect ratio change, background clutter, camera motion, fast motion, illumination variation, low resolution, out-view, similar object, scale variation, viewpoint change. In a rarely occurring scenario where the target is fully occluded, and a very similar or the same object is in the vicinity of the last known location of the target, the PFCNNOcc may sometimes detect the similar object instead of the full-occluded target. The PFCNNOcc combines CNN models, trained with the novel synthetic S-UAV dataset [10, 11] and the UAV123 dataset [2] to predict the optical flow and occlusion maps, segmentation masks, and classification scores. The success of PFCNNOcc is based on the following novel functions:

- Re-Identification Module Final PFCNNOcc: It utilizes the forward and backward optical flow predictions, and the segmentation masks of the consecutive images with a novel matching algorithm to re-detect the target at the beginning of each image pair and to determine the location and the size of it at the pixel level with high precision under short-term or long-term partial or full occlusion states with varying frequencies. In the meantime, the average motion vector for the target is also calculated. The location, size, and motion vector, calculated with high accuracy, pave the way to boost the tracking performance of the final version of the PFCNNOcc.
- Occlusion Detector and Calculator Module Final PFCNNOcc: It is vital to decide whether a pixel is non-occluded, will-be-occluded, and already-occluded, and to calculate the degree of occlusion by considering the possible scale and aspect ratio variations. This novel function increases the robustness of the method against partial and full occlusion, and hence, it makes a difference compared to the SOTA trackers.

- Full Occlusion Dense Search Module Final PFCNNOcc: It is a combination of the "Re-Identification Module Final PFCNNOcc"and "Occlusion Detector and Calculator Final PFCNNOcc"with a dense-search algorithm to re-detect the target after a short-term or long-term full occlusion. The overall performance of most of the SOTA trackers decreases because of the full occlusion condition, creating a huge uncertainty in terms of the location and the size after the target vanishes. The amount of this uncertainty is cumbersome for the trackers to handle after the target appears on the scene again. Hence, "Full Occlusion Dense Search Module Final PFCNNOcc"has the key contribution to why the PFCNNOcc performs the best.
- Merged-Target Detection: At the end of the iteration, it detects whether or not a target is occluded by another vehicle during tracking. After the occlusion or merged-target condition ends, the discrimination capability of the function, separating the target from other vehicles on the run, helps the PFCNNOcc to continue tracking precisely.

All these novel functions lead the final PFCNNOcc to the top of the SOTA in terms of precision and success scores against all visual problems, and even the Siamese network-based CNN methods are not able to achieve such high tracking scores. Moreover, the results of the ablation study prove that "Occlusion Calculator" and "Merged Target Detection" functions evolve the tracking performance of the PFCNNOcc step by step to the top of the SOTA trackers. The addition of the background class to the training dataset positively affects the correctness of class scores that are utilized to update the weight of particles in the particle filter, as well as the reliability estimation of the state variables of the tracked target.

Hence, these functions are the contributions of the PFCNNOcc. To the best of the author's knowledge, the PFCNNOcc is the first method in the literature, predicting and utilizing the optical flow and segmentation to estimate the state variables of the target as well as the occlusion state and degree of it during tracking in UAV videos.

Based on the contributions and the results obtained in this thesis, new ideas for future work are determined and itemized as,

- The Evolvement of S-UAV Dataset: The diversity of vehicle models will be increased, and new urban and rural environments and scenarios will be prepared. And these scenarios will be increased via the application of some transformations to the vehicles, such as scaling, illumination variation, and rotation. Segmentation masks are currently created based on the vehicle and background. It is intended to be evolved by considering the type of vehicles, too. Additionally, the dataset will also include maritime vehicles.
- The Evolvement of the Ground Truth Data of UAV123 Dataset: Segmentation masks are currently created based on the vehicle and the background. They will be evolved by considering the type of vehicles too.
- Development of MultiTrack-PFCNNOcc: PFCNNOcc will be redesigned to be able to detect and track multiple vehicles, simultaneously. Tracking all vehicles on the scene at the same time will provide the relative position and dynamic motion of the targets to each other, and this information will provide a chance to keep track of the occluder and occluded information, as well as the degree of occlusion during tracking.

REFERENCES

1. Maraş, B., N. Arıca and A. Baytan Ertüzün, "Görsel Hedef Takibi Yöntemlerine Genel Bakış", *EMO Bilimsel Dergi*, Vol. 7, pp. 5 – 16, 2017.

2. University, K. A., "UAV123 Dataset", `https://ivul.kaust.edu.sa/benchmark-and-simulator-uav-tracking-dataset`, 2016, accessed on June 10, 2025.

3. University, P. S., "VIVID DARPA UAV Dataset", `https://vision.cse.psu.edu/data/vividEval/datasets/datasets.html`, 2005, accessed on June, 10 2025.

4. Taufique, A. M. N., B. Minnehan and A. Savakis, "Benchmarking Deep Trackers on Aerial Videos", *Sensors*, Vol. 20, No. 2, p. 547, 2020.

5. Wang, Y., Y. Yang, Z. Yang, L. Zhao, P. Wang and W. Xu, "Occlusion Aware Unsupervised Learning of Optical Flow", *Computer Vision and Pattern Recognition (CVPR)*, pp. 4884–4893, Salt Lake City, USA, 2018.

6. Maraş, B., N. Arica and A. Ertüzün, "A Robust Vehicle Tracking in Low-altitude UAV Videos", *Machine Vision and Applications*, Vol. 34, No. 5, pp. 77–100, 2023.

7. Johansen, A. M. and A. Doucet, "A Note on Auxiliary Particle Filters", *Elsevier*, Vol. 78, No. 12, pp. 1498–1504, 2008.

8. McNames, J., "Auxiliary Particle Filter, ECE 510 State Space Tracking Lecture", `https://www.youtube.com/watch?v=IGI24Z4KysM`, 2016, accessed on June 10, 2025.

9. Blevins, J., "Notes on the Auxiliary Particle Filter Based on Pitt and Shephard", `https://jblevins.org/notes/auxiliary-particle-filter`, 2009, ac-

cessed on June 10, 2025.

10. Yılmaz, C., B. Maraş, N. Arıca and A. B. Ertüzün, "Creation of Annotated Synthetic UAV Video Dataset for Object Detection and Tracking", *31st Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, İstanbul, Türkiye, 2023.

11. Yılmaz, C., B. Maraş, G. Yılmaz, G. Ceylan, Hamamcıoğlu, N. Arıca and A. B. Ertüzün, "Enhancing Object Detection Algorithms by Synthetic Aerial Images", *31st Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, İstanbul, Türkiye, 2023.

12. Mueller, M., N. Smith and B. Ghanem, "A Benchmark and Simulator for UAV Tracking", *European Conference on Computer Vision (ECCV)*, pp. 445–461, Amsterdam, Netherlands, 2016.

13. Wu, Y., J. Lim and M.-H. Yang, "Online Object Tracking: A Benchmark", *Computer Vision and Pattern Recognition (CVPR)*, pp. 2411–2418, Portland/USA, 2013.

14. Maraş, B., N. Arıca and A. B. Ertüzün, "Object Tracking by Combining Tracking-by-Detection and Marginal Particle Filter", *24th Signal Processing and Communication Application Conference (SIU)*, pp. 1029–1032, Zonguldak, Türkiye, 2016.

15. Smeulders, A. W. M., D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan and M. Shah, "Visual Tracking: An Experimental Survey", *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 36, No. 7, pp. 1442–1468, 2014.

16. Matas, J., K. Mikolajczyk and Z. Kalal, "P-N learning: Bootstrapping Binary Classifiers by Structural Constraints", *Computer Vision and Pattern Recognition (CVPR)*, pp. 49–56, Los Alamitos, CA, USA, 2010.

17. Baker, S. and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework",

International Journal of Computer Vision, Vol. 56, No. 3, pp. 221 – 255, 2004.

18. Henriques, J. F., R. Caseiro, P. Martins and J. Batista, "Exploiting the Circulant Structure of Tracking-by-Detection with Kernels", *European Conference on Computer Vision (ECCV)*, pp. 702–715, Berlin, Germany, 2012.

19. Henriques, J. F., R. Caseiro, P. Martins and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters", *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 37, No. 3, pp. 583–596, 2015.

20. Danelljan, M., G. Häger, F. S. Khan and M. Felsberg, "Discriminative Scale Space Tracking", *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 39, No. 8, pp. 1561–1575, 2017.

21. Danelljan, M., G. Häger, F. S. Khan and M. Felsberg, "Learning Spatially Regularized Correlation Filters for Visual Tracking", *International Conference on Computer Vision (ICCV)*, pp. 4310–4318, Santiago, Chile, 2015.

22. Hare, S., A. Saffari and P. H. S. Torr, "Struck: Structured Output Tracking with Kernels", *International Conference on Computer Vision (ICCV)*, pp. 263–270, Barcelona, Spain, 2011.

23. Hare, S., S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks and P. H. S. Torr, "Struck: Structured Output Tracking with Kernels", *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 38, No. 10, pp. 2096–2109, 2016.

24. Zhang, J., S. Ma and S. Sclaroff, "MEEM: Robust Tracking via Multiple Experts Using Entropy Minimization", *European Conference on Computer Vision (ECCV)*, pp. 188–203, Zurich, Switzerland, 2014.

25. Nam, H. and B. Han, "Learning Multi-domain Convolutional Neural Networks for Visual Tracking", *Computer Vision and Pattern Recognition (CVPR)*, pp.

4293–4302, Los Alamitos, CA, USA, 2016.

26. Pu, S., Y. Song, C. Ma, H. Zhang and M. Yang, "Deep Attentive Tracking via Reciprocative Learning", *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, pp. 1935–1945, Montreal, Canada, 2018.

27. Danelljan, M., A. Robinson, F. Shahbaz Khan and M. Felsberg, "Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking", *European Conference on Computer Vision (ECCV)*, pp. 472–488, Amsterdam, Netherlands, 2016.

28. Danelljan, M., G. Bhat, F. Khan and M. Felsberg, "ECO: Efficient Convolution Operators for Tracking", *Computer Vision and Pattern Recognition (CVPR)*, pp. 6931–6939, Honolulu, HI, USA, 2017.

29. Li, F., C. Tian, W. Zuo, L. Zhang and M.-H. Yang, "Learning Spatial-Temporal Regularized Correlation Filters for Visual Tracking", *Computer Vision and Pattern Recognition (CVPR)*, pp. 4904–4913, Salt Lake City, UT, USA, 2018.

30. Jalil Mozhdehi, R. and H. Medeiros, "Deep Convolutional Correlation Iterative Particle Filter for Visual Tracking", *Computer Vision and Image Understanding*, Vol. 222, No. C, p. 103479, 2022.

31. Mozhdehi, R. J. and H. Medeiros, "Deep Convolutional Particle Filter for Visual Tracking", *International Conference on Image Processing (ICIP)*, pp. 3650–3654, Beijing, China, 2017.

32. Mozhdehi, R. J., Y. Reznichenko, A. Siddique and H. Medeiros, "Deep Convolutional Particle Filter with Adaptive Correlation Maps for Visual Tracking", *International Conference on Image Processing (ICIP)*, pp. 798–802, Athens, Greece, 2018.

33. Mozhdehi, R. J., Y. Reznichenko, A. Siddique and H. Medeiros, "Convolutional Adaptive Particle Filter with Multiple Models for Visual Tracking", *Advances in Visual Computing*, pp. 474–486, Las Vegas, NV, USA, 2018.

34. Mozhdehi, R. J. and H. Medeiros, "Deep Convolutional Likelihood Particle Filter for Visual Tracking", *Advances in Computer Vision and Computational Biology*, pp. 27–38, Cham, 2021.

35. Fu, C., Z. Cao, Y. Li, J. Ye and C. Feng, "Onboard Real-time Aerial Tracking with Efficient Siamese Anchor Proposal Network", *Transactions on Geoscience and Remote Sensing*, Vol. 60, pp. 1–13, 2021.

36. Bertinetto, L., J. Valmadre, J. F. Henriques, A. Vedaldi and P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking", *European Conference on Computer Vision (ECCV)*, pp. 850–865, Amsterdam, Netherlands, 2016.

37. Li, B., J. Yan, W. Wu, Z. Zhu and X. Hu, "High Performance Visual Tracking with Siamese Region Proposal Network", *Computer Vision and Pattern Recognition (CVPR)*, pp. 8971–8980, Salt Lake City, UT, USA, 2018.

38. Zhu, Z., Q. Wang, B. Li, W. Wu, J. Yan and W. Hu, "Distractor-Aware Siamese Networks for Visual Object Tracking", *European Conference on Computer Vision (ECCV)*, p. 103–119, Berlin, Heidelberg, Germany, 2018.

39. Cao, Z., C. Fu, J. Ye, B. Li and Y. Li, "SiamAPN++: Siamese Attentional Aggregation Network for Real-Time UAV Tracking", *International Conference on Intelligent Robots and Systems (IROS)*, pp. 3086–3092, Prague, Czech Republic, 2021.

40. Chen, B., D. Wang, P. Li, S. Wang and H. Lu, "Real-Time 'Actor-Critic' Tracking", *European Conference on Computer Vision (ECCV)*, pp. 328–345, Berlin, Heidelberg, Germany, 2018.

41. Horn, B. K. P. and B. G. Schunck, "Determining Optical Flow", *Artificial Intelligence*, Vol. 17, No. 1-3, pp. 185–203, 1981.

42. Savian, S., M. Elahi and T. Tillo, *Optical Flow Estimation with Deep Learning, a Survey on Recent Advances*, pp. 257–28, Bolzano, Italy, 2020.

43. Wang, Y., Y. Yang, Z. Yang, L. Zhao and W. Xu, "Occlusion Aware Unsupervised Learning of Optical Flow", *Computer Vision and Pattern Recognition (CVPR)*, pp. 4884–4893, Salt Lake City, UT, USA., 2018.

44. Brox, T., C. Bregler and J. Malik, "Large Displacement Optical Flow", *Computer Vision and Pattern Recognition (CVPR)*, pp. 41–48, Miami, Florida, USA, 2009.

45. Lucas, B. D. and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, p. 674–679, San Francisco, CA, USA, 1981.

46. Güney, F. and A. Geiger, "Deep Discrete Flow", *Asian Conference on Computer Vision (ACCV)*, pp. 207–224, Taipei, Taiwan, 2016.

47. Dosovitskiy, A., P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks", *International Conference on Computer Vision (ICCV)*, pp. 2758–2766, Santiago, Chile, 2015.

48. Mayer, N., E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation", *Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, Las Vegas, NV, USA, 2016.

49. Butler, D. J., J. Wulff, G. B. Stanley and M. J. Black, "A Naturalistic Open Source Movie for Optical Flow Evaluation", *European Conference on Computer*

Vision (ECCV), pp. 611–625, Berlin, Germany, 2012.

50. Menze, M. and A. Geiger, "Object Scene Flow for Autonomous Vehicles", *Computer Vision and Pattern Recognition (CVPR)*, pp. 3061–3070, Boston, MA, USA, 2015.

51. Kiefer, B., D. Ott and A. Zell, "Leveraging Synthetic Data in Object Detection on Unmanned Aerial Vehicles", *International Conference on Pattern Recognition (ICPR)*, pp. 3564–3571, Montreal, QC, Canada, 2022.

52. Lazaridis, L., A. Dimou and P. Daras, "Abnormal Behavior Detection in Crowded Scenes Using Density Heatmaps and Optical Flow", *European Signal Processing Conference (EUSIPCO)*, pp. 2060–2064, Rome, Italy, 2018.

53. Richter, S. R., Z. Hayder and V. Koltun, "Playing for Benchmarks", *International Conference on Computer Vision (ICCV)*, pp. 2213–2222, Venice, Italy, 2017.

54. Hu, Y.-T., H.-S. Chen, K. Hui, J.-B. Huang and A. G. Schwing, "SAIL-VOS: Semantic Amodal Instance Level Video Object Segmentation – A Synthetic Dataset and Baselines", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3100–3110, Long Beach, CA, USA, 2019.

55. Schröder, G., T. Senst, E. Bochinski and T. Sikora, "Optical Flow Dataset and Benchmark for Visual Crowd Analysis", *International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Auckland, New Zealand, 2018.

56. Gaidon, A., Q. Wang, Y. Cabon and E. Vig, "VirtualWorlds as Proxy for Multi-object Tracking Analysis", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4340–4349, Las Vegas, NV, USA, 2016.

57. Bai, M., W. Luo, K. Kundu and R. Urtasun, "Exploiting Semantic Information and Deep Matching for Optical Flow", *European Conference on Computer Vision*

(ECCV), pp. 154–170, Amsterdam, Netherlands, 2016.

58. Bailer, C., K. Varanasi and D. Stricker, "CNN-Based Patch Matching for Optical Flow with Thresholded Hinge Embedding Loss", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2710–2719, Los Alamitos, CA, USA, 2017.

59. Xu, J., R. Ranftl and V. Koltun, "Accurate Optical Flow via Direct Cost Volume Processing", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5807–5815, Los Alamitos, CA, USA, 2017.

60. Ronneberger, O., P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, Munich, Germany, 2015.

61. Ilg, E., N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1647–1655, Los Alamitos, CA, USA, 2017.

62. Ranjan, A. and M. Black, "Optical Flow Estimation Using a Spatial Pyramid Network", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2720–2729, Los Alamitos, CA, USA, 2017.

63. Hui, T.-W., X. Tang and C. C. Loy, "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8981–8989, Salt Lake City, UT, USA, 2018.

64. Cheng, J., Y.-H. Tsai, S. Wang and M.-H. Yang, "SegFlow: Joint Learning for Video Object Segmentation and Optical Flow", *International Conference on Computer Vision (ICCV)*, pp. 686–695, Venice, Italy, 2017.

65. Sun, D., X. Yang, M.-Y. Liu and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume", *Conference on Computer Vision*

and Pattern Recognition (CVPR), pp. 8934–8943, Los Alamitos, CA, USA, 2018.

66. Sun, D., X. Yang, M.-Y. Liu and J. Kautz, "Models Matter, So Does Training: An Empirical Study of CNNs for Optical Flow Estimation", *IEEE Transactions on Pattern Analysis & Machine Intelligence (TPAMI)*, Vol. 42, No. 06, pp. 1408–1423, 2020.

67. Ren, Z., J. Yan, B. Ni, B. Liu, X. Yang and H. Zha, "Unsupervised Deep Learning for Optical Flow Estimation", *Conference on Artificial Intelligence (AAAI)*, p. 1495–1501, San Francisco, California, USA, 2017.

68. Jaderberg, M., K. Simonyan, A. Zisserman and K. Kavukcuoglu, "Spatial Transformer Networks", *International Conference on Neural Information Processing Systems - Volume 2*, p. 2017–2025, Montreal, Canada, 2015.

69. Yu, J. J., A. W. Harley and K. G. Derpanis, "Back to Basics: Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness", *European Conference on Computer Vision (ECCV)*, pp. 3–10, Amsterdam, Netherlands, 2016.

70. Meister, S., J. Hur and S. Roth, "UnFlow: unsupervised learning of optical flow with a bidirectional census loss", *AAAI Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence Conference and AAAI Symposium on Educational Advances in Artificial Intelligence*, pp. 7251 – 7259, New Orleans, Louisiana, USA, 2018.

71. Liu, L., J. Zhang, R. He, Y. Liu, Y. Wang, Y. Tai, D. Luo, C. Wang, J. Li and F. Huang, "Learning by Analogy: Reliable Supervision from Transformations for Unsupervised Optical Flow Estimation", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6488–6497, Los Alamitos, CA, USA, 2020.

72. Ren, Z., J. Yan, X. Yang, A. Yuille and H. Zha, "Unsupervised Learning of Optical

Flow With Patch Consistency and Occlusion Estimation", *Pattern Recognition*, Vol. 103, p. 107191, 2020.

73. Ilg, E., T. Saikia, M. Keuper and T. Brox, "Occlusions, Motion and Depth Boundaries with a Generic Network for Disparity, Optical Flow or Scene Flow Estimation", *European Conference on Computer Vision (ECCV)*, pp. 626–643, Munich, Germany, 2018.

74. Tian, L., Z. Tu, D. Zhang, J. Liu, B. Li and J. Yuan, "Unsupervised Learning of Optical Flow with CNN-based Non-Local Filtering", *Transactions on Image Processing*, Vol. 29, pp. 8429–8442, 2020.

75. Hur, J. and S. Roth, "MirrorFlow: Exploiting Symmetries in Joint Optical Flow and Occlusion Estimation", *International Conference on Computer Vision (ICCV)*, pp. 312–321, Los Alamitos, CA, USA, 2017.

76. Zhang, C., C. Feng, Z. Chen, W. Hu and M. Li, "Parallel Multiscale Context-based Edge-preserving Optical Flow Estimation with Occlusion detection", *Signal Processing: Image Communication*, Vol. 101, p. 116560, 2022.

77. Hur, J. and S. Roth, "Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5747–5756, Long Beach, CA, USA, 2019.

78. Saxena, R., R. Schuster, O. Wasenmüller and D. Stricker, "PWOC-3D: Deep Occlusion-Aware End-to-End Scene Flow Estimation", *Intelligent Vehicles Symposium (IV)*, pp. 324–331, Paris, France, 2019.

79. Janai, J., F. Güney, A. Ranjan, M. Black and A. Geiger, "Unsupervised Learning of Multi-Frame Optical Flow with Occlusions", *European Conference on Computer Vision (ECCV)*, pp. 713–731, Munich, Germany, 2018.

80. Zhang, C., Z. Zhou, Z. Chen, W. Hu, M. Li and S. Jiang, "Self-attention-based

Multiscale Feature Learning Optical Flow with Occlusion Feature Map Prediction", *Transactions on Multimedia*, Vol. 24, pp. 3340 – 3354, 2022.

81. Elnashar, G., A. Ralescu, A. Youssif, O. Elmowafy and W. Asker, "Occlusion Detection and Processing Using Optical Flow and Particle Filter", *International Journal of Advanced Intelligence Paradigms*, Vol. 15, No. 1, pp. 63–76, 2020.

82. Du, D., Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang and Q. Tian, "The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking", *European Conference on Computer Vision (ECCV)*, pp. 375–391, Online, 2018.

83. Wu, Y., J. Lim and M.-H. Yang, "Object Tracking Benchmark", *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Vol. 37, No. 9, pp. 1–1, 2015.

84. Hong, Z., Z. Chen, C. Wang, X. Mei, D. Prokhorov and D. Tao, "Multi-Store Tracker (MUSTER): A Cognitive Psychology Inspired Approach to Object Tracking", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 749–758, Boston, MA, USA, 2015.

85. Candy, J. V., *Bayesian Signal Processing: Classical, Modern and Particle Filtering Methods*, Wiley-Interscience, New York, 2009.

86. Arulampalam, M. S., S. Maskell, N. Gordon and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking", *Signal Processing, IEEE Transactions*, Vol. 50, No. 2, pp. 174–188, 2002.

87. Simonyan, K. and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *International Conference on Learning Representations (ICLR)*, pp. 1–14, San Diego, CA, USA, 2015.

88. Stanford Vision Lab, P. U., Stanford University, "Image Net", `http://www.image-net.org/`, 2021, accessed on June 10, 2025.

89. Colab, G., "A Keras API", https://colab.research.google.com/github/Hvass-Labs/TensorFlow-Tutorials/blob/master/03C_Keras_API.ipynb, 2025, accessed on June 10, 2025.

90. TensorFlow, "Adam", https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam, 2025, accessed on June 10, 2025.

91. Technologies, U., "Unity Engine", https://unity.com/products/unity-engine, 2025, accessed on June 10, 2025.

92. Community, B. O., "Blender", http://www.blender.org, 2018, accessed on June 10, 2025.

93. Welch, T. A., "A Technique for High-performance Data Compression", *Computer*, Vol. 17, No. 06, pp. 8–19, 1984.

94. Corporation, C., "CVAT.ai", https://www.cvat.ai/, 2025, accessed on June 10, 2025.

95. Yilmaz, C., "S-UAV Dataset", https://sites.google.com/view/modatuv/synthethic-dataset, 2023, accessed on June 10, 2025.

96. Foundation, T. L., "RMSprop", https://pytorch.org/docs/stable/generated/torch.optim.RMSprop.html, 2025, accessed on June 10, 2025.

97. Maraş, B., "PFCNN_UAV123_30fps_Visual_Results", https://github.com/bahri9746/PFCNN_UAV123_30fps_Visual_Results, 2025, accessed on June 10, 2025.

98. Maraş, B., "FirstVer_PFCNNOcc_S-UAV_Visual_Results", https://github.com/bahri9746/FirstVer_PFCNNOcc_S-UAV_Visual_Results, 2025, accessed on June 10, 2025.

99. Maraş, B., "PFCNNOcc_UAV123_10fps_Visual_Results", https://github.com/bahri9746/PFCNNOcc_UAV123_10fps_Visual_Results, 2025, accessed on June 10, 2025.

100. Maraş, B., "Ablation-Versions-of-PFCNNOcc_UAV123-10-fps_Visual_Results", https://github.com/bahri9746/Ablation-Versions-of-PFCNNOcc_UAV123-10-fps_Visual_Results, 2025, accessed on June 10, 2025.

101. Luo, Y., X. Wang, Y. Liao, Q. Fu, C. Shu, Y. Wu and Y. He, "A Review of Homography Estimation: Advances and Challenges", *Electronics*, Vol. 12, No. 24, p. 4977, 2023.

APPENDIX A: THE ALGORITHM TO CREATE THE OPTICAL FLOW DATA

First, for each triangle that makes up the vehicle, 2-D position information of six pixel locations is recorded: three corner points from the video images at time instants T_n and T_{n+1}, along with three additional points determined between the corners. Then, the operations described in Figure A.1 are applied. Using the pixel locations belonging to the same triangle in consecutive images, the homography matrix is calculated. The structure of the homography matrix, the method of projecting the pixels within the triangles between the images, and the calculation of optical flow are outlined in the following steps [101]:

- At least four matching points between two images are required to find the homography matrix. Hence, the corner points of the triangle, the points between these corner points and the matrices (3x6), having pixel locations belonging these six points, V_n and V_{n+1}, from two consecutive video images, T_n and T_{n+1}, have to be determined to find the homography, H, matrix as

$$H = \begin{bmatrix} h00 & h01 & h02 \\ h10 & h11 & h12 \\ h20 & h21 & h22 \end{bmatrix}, V_n = \begin{bmatrix} x_n^1 & x_n^2 & . & x_n^6 \\ y_n^1 & y_n^2 & . & x_n^6 \\ 1 & 1 & . & 1 \end{bmatrix}, H = V_{n+1} * V_n^{-1} \tag{A.1}$$

  where h_{ij}'s represent the elements of the matrix at ith row and jth column. While T and T_{n+1} refer to the consecutive synthetic vehicle images, generated with Unity, n is the number of images inside the video sequence, $n = 0, 1, .., N$.
- The position vectors of all pixels, $[x_n^i, y_n^i, 1]$ in 3-D coordinates, within the triangle in the T_n video image, are multiplied and projected by the homography matrix to calculate the position vectors of the pixels in the T_{n+1} image coordinates,

$[x^i_{n+1}, y^i_{n+1}, 1]$ as

$$\begin{bmatrix} x^i_{n+1} \\ y^i_{n+1} \\ 1 \end{bmatrix} = H \begin{bmatrix} x^i_n \\ y^i_n \\ 1 \end{bmatrix} \tag{A.2}$$

where x^i_n and y^i_n are the x and y position of all points, $i, i = 1, 2, ..., M$, inside the triangle at time n. And x^i_{n+1}, y^i_{n+1} are the x and y position of all points, projected to T_{n+1} image coordinates at time $n + 1$, and these points generate a triangle in the T_{n+1} image coordinates.

- Subsequently, the position vectors of the pixels transferred to the T_{n+1} image coordinates are subtracted from the position vectors of the pixels inside the same triangle in the T_n image, $[x^i_n, y^i_n]$, on a pixel basis to find the optical flow vectors as

$$\begin{bmatrix} HorizontalOpticalFlow^i_n \\ VerticalOpticalFlow^i_n \end{bmatrix} = \begin{bmatrix} x^i_{n+1} \\ y^i_{n+1} \end{bmatrix} - \begin{bmatrix} x^i_n \\ y^i_n \end{bmatrix} \tag{A.3}$$

where $HorizontalOpticalFlow^i_n$ and $VerticalOpticalFlow^i_n$ refers to the optical flow vector along x and y image coordinates of pixel, i, in the triangle. Thus, while transitioning from the T_n image to the T_{n+1} image, the amount of horizontal (x-axis) and vertical (y-axis) movement of all pixels belonging to a triangle that is part of the vehicle is obtained as the optical flow vector of forward motion, as shown in Figure A.1.

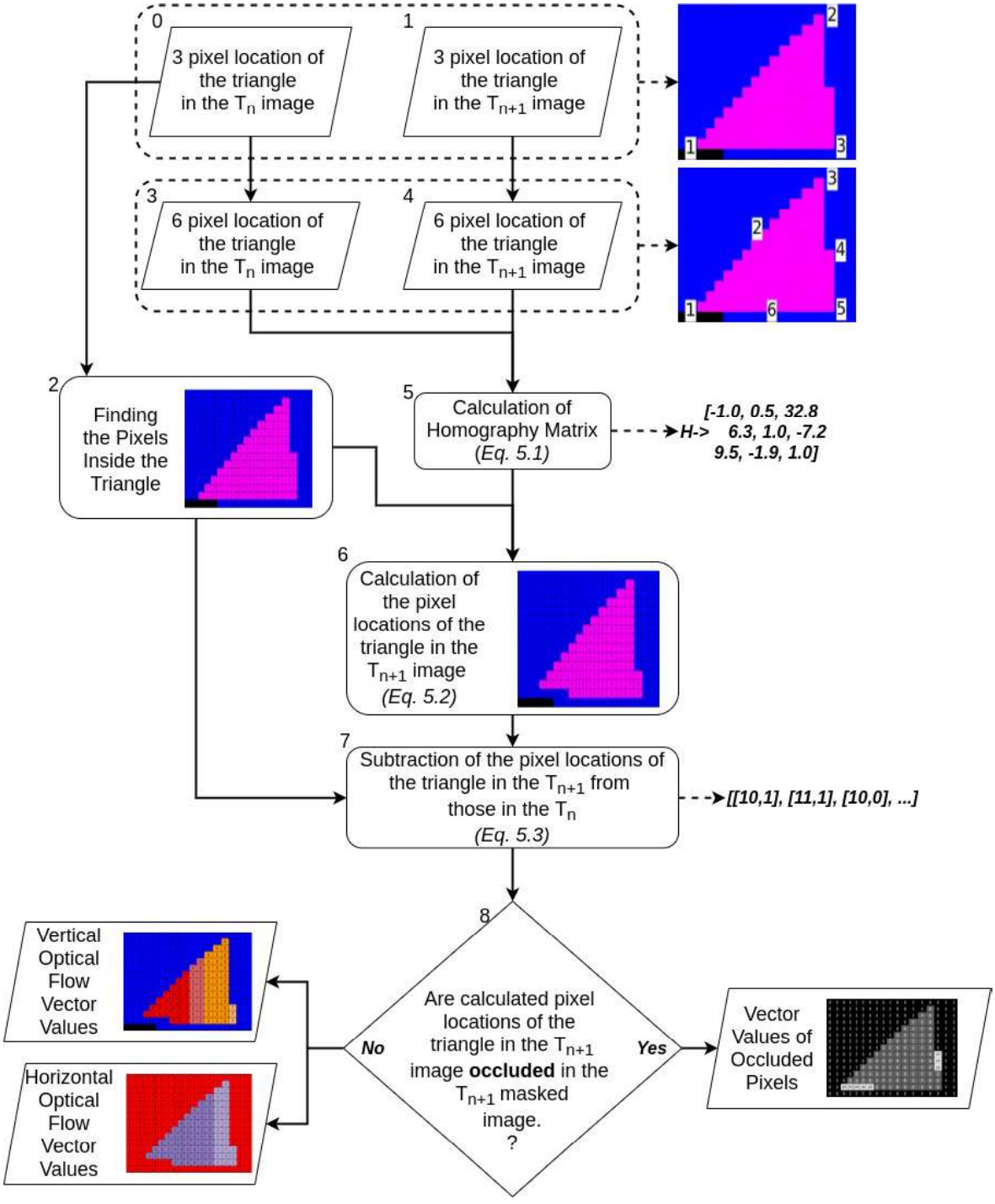


Figure A.1. Calculation of optical flow vectors.

APPENDIX B: TRAINING OF CNN MODELS

B.1. Training of the CNN Model for Optical Flow and Occlusion

During training, in advance of the iterations over "Flow Model" and "Occlusion Model", the different numbers of features from images are extracted using "Pyramidal Feature Extractor" with six layers as shown in Figure B.1. The original image tensor of size 1x3x250x250 (batch size x number of channels x width of image x height of image), and each layer of the pyramid is constructed by the output of "Double Convolution" module whose input is the output of previous "Double Convolution" module. Fifth layer of the pyramid starts with 1x16x125x125, and the dimensions of the features at the upper layers decreases while the number of channels increases. Hence, "Pyramidal Feature Extractor" can be defined as an "Encoder".

When Figure 5.2 is examined, it is evident that the "Flow Model" and "Occlusion Model" share the same sub-modules, apart from their input sections. Unlike the "Occlusion Model", the input of the "Flow Model" includes similarity scores, which are the correlations of the features, extracted through the "Pyramidal Feature Extractor", shown in Figure B.1 and warped using estimated forward and backward flow tensors by the "Cost Volume Computation Model". After features are extracted at the beginning of each batch iteration, "Flow and Occlusion Models" are iteratively trained in a loop. Dimensions of features at the input and the output of the sub-modules are displayed in a format of batch size x number of channels x D_l x D_l, where D_l represents dimension of features along x and y axes in image coordinates for the pyramidal level l.

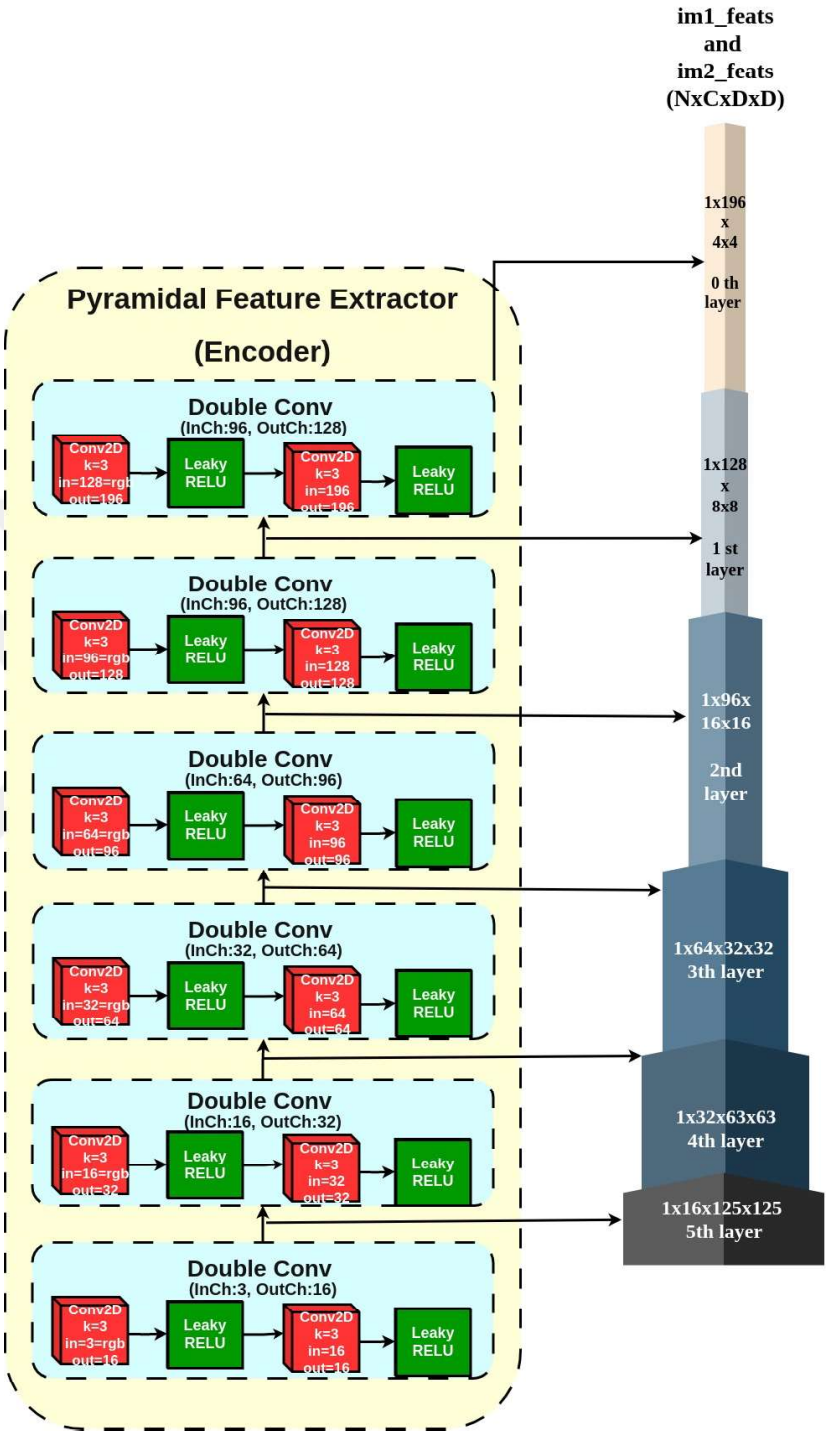


Figure B.1. Pyramidal feature extractor.

For clarity, one iteration of the training will be considered here. Subindex l is taken as 0, hence D_l is 4 for this layer. After pyramidal features with six layers are constructed, the feature with 1x196x4x4 tensor in the zeroth layer is fed to the

"Flow Model" as depicted in Figure B.2. For the zeroth layer, the feature are sent to "Compute Cost Volume" sub-module. Here, the original features, extracted by "Pyramidal Feature Extractor", and the ones, derived by warping the original features with predicted forward flow ($predicted_flow_f$) and backward flow ($predicted_flow_b$) are correlated to find $corr_f$ and $corr_b$. As shown in Figure B.2, for the zeroth layer, flows are initiated with zero tensors whose dimensions are the same as the features of the zeroth layer. For the other iterations or layers, flows predicted at the end of the iteration of the previous layer are used after being upsampled with "Upsample2d_as" sub-module and warped with "Warping Layer". Before $predicted_flow_f$ and $predicted_flow_b$ are fed to the "Warping Layer", their flow vectors along with x and y axes are rescaled using u_{scale} and v_{scale} parameters depending on the pyramid level of extracted features. While rescaling predicted flows ($predicted_flow_f$ and $predicted_flow_b$) from actual flow size (level=6 (actual, top layer)) into the dimension of predicted flows at lower pyramidal level, flow vectors are divided with u_{scale} and v_{scale} parameters to align the magnitude of vectors appropriately depending on the level in pyramid.

Then, 1x115x4x4 tensor is constructed by concatenation of correlated features of image features, $corr_f$ and $corr_b$, as outputs of "Compute Cost Volume" sub-module, rescaled flow tensors, $flow_f$ and $flow_b$, and image features, $im1_1x1_feat$ and $im2_1x1_feat$, and sent to "Flow Estimator" module, shown in Figure B.3 to find the residual flows, $flow_res_f$ and $flow_res_b$, with dimension of 1x2x4x4, and intermediate outputs, $output_intm_f$ and $output_intm_b$, with dimension of 1x563x4x4.

As realized from the outputs of "Flow Estimator" sub-module, it acts as an encoder because it outputs high level intermediate features. Then residual flow outputs are added to $flow_f$ and $flow_b$ to create the intermediate flows with the same name, $flow_f$ and $flow_b$. All intermediate outputs and flows are then concatenated, and sent to the "Flow Context Network" module as a tensor with the dimension of 1x565x4x4, as shown in Figure B.4 , to be decoded into $flow_fine_f$ and $flow_fine_b$.

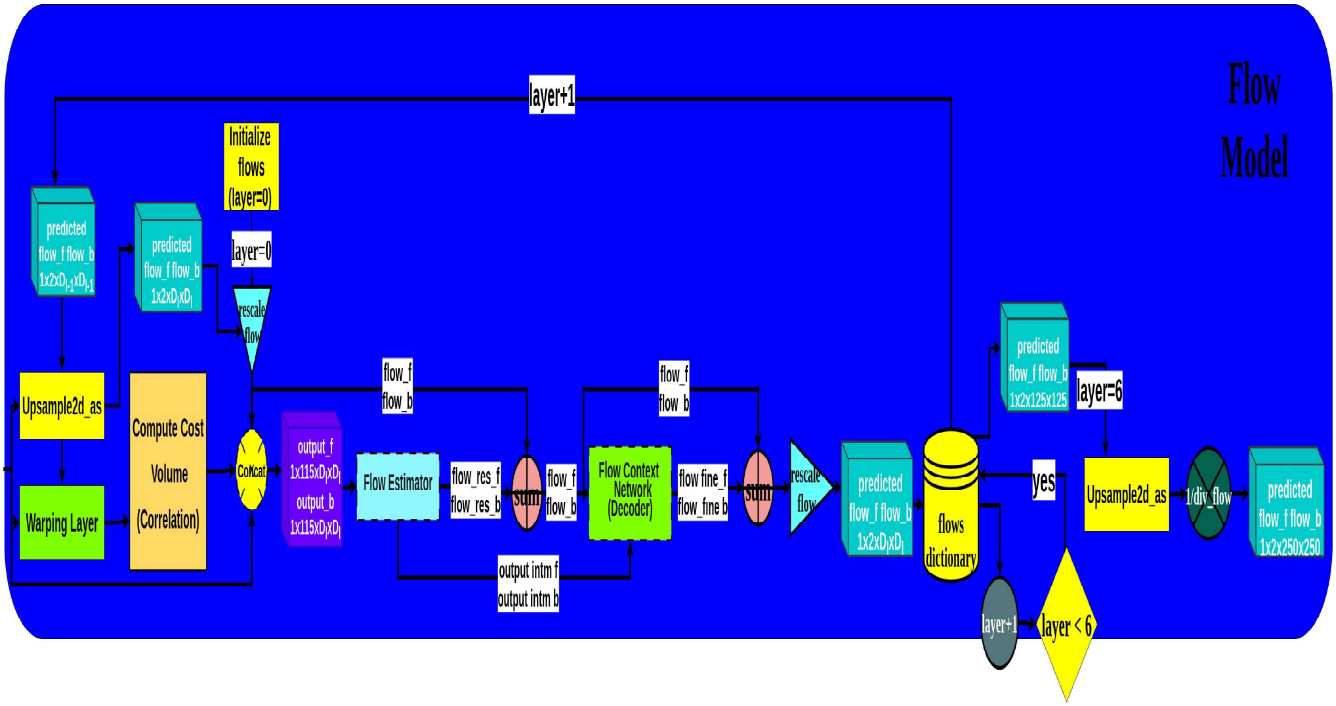


Figure B.2. Flow model.

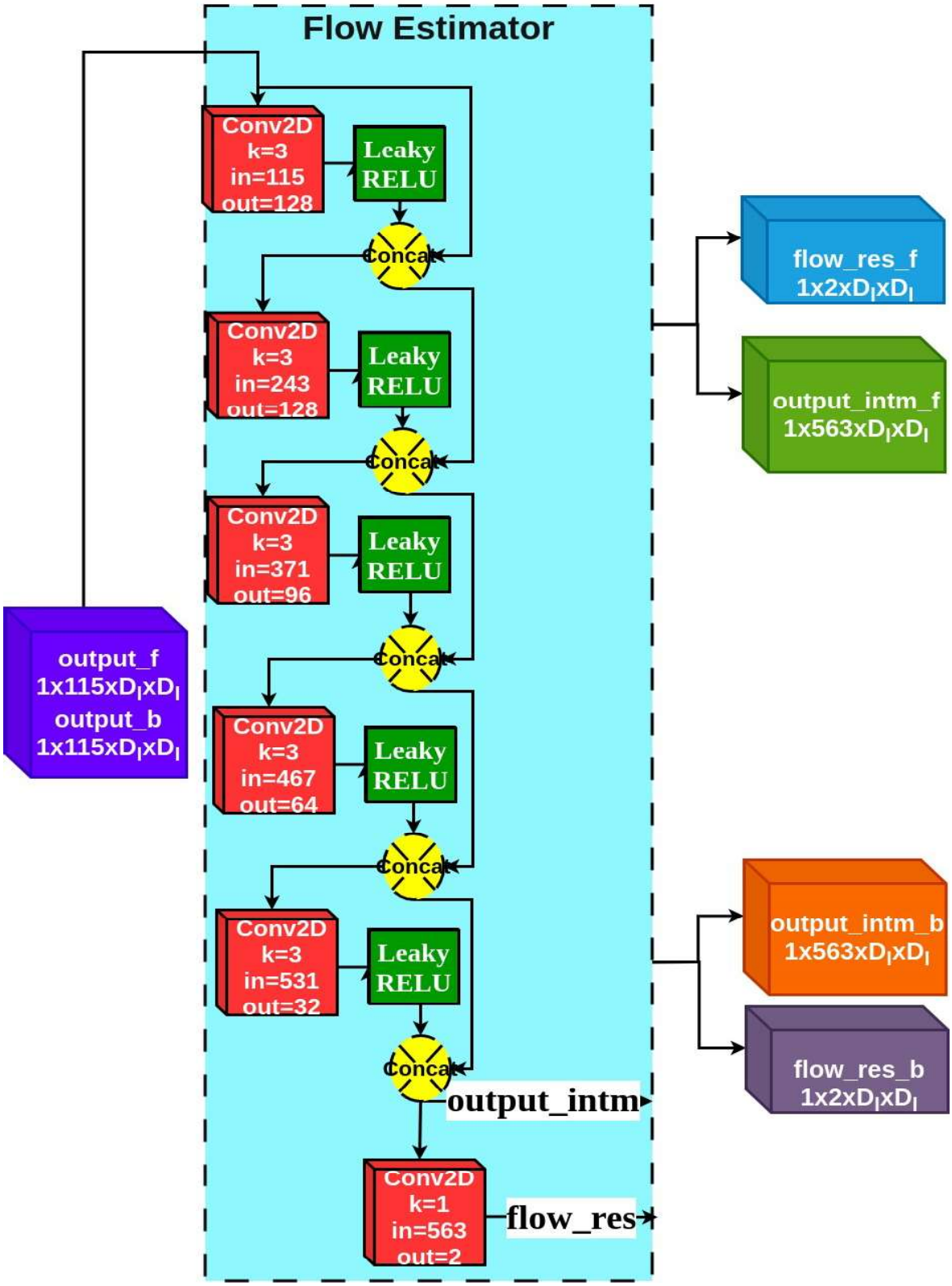


Figure B.3. Flow estimator.

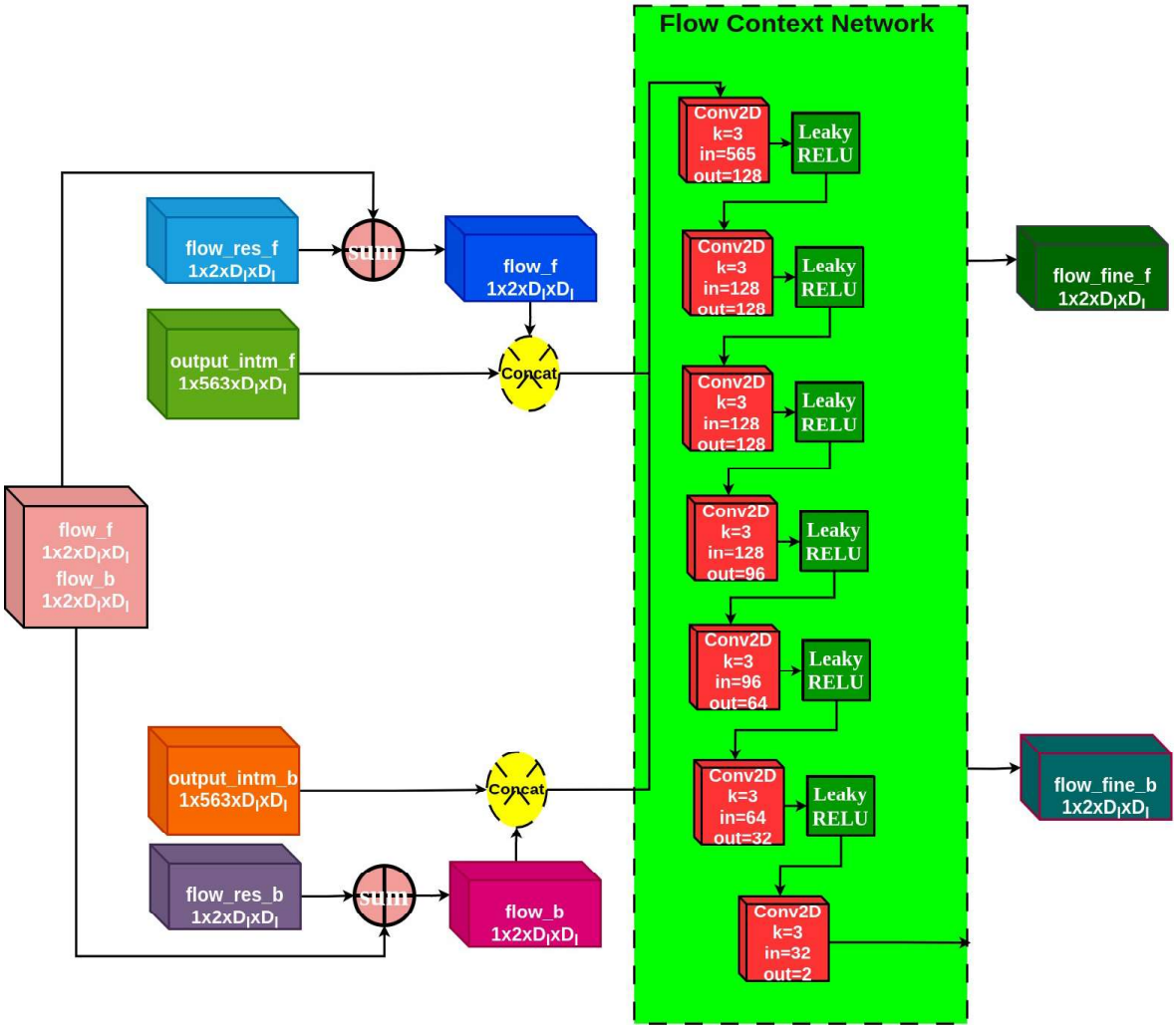


Figure B.4. Flow context network.

"Flow Context Network" acts as a decoder by creating low-level fine optical flow tensors, $flow_fine_f$ and $flow_fine_b$, from high-level features. Fine flow tensors are added to $flow_f$ and $flow_b$ and rescaled to find $predicted_flow_f$ and $predicted_flow_b$ for this layer. As displayed in Figure B.2, rescaling is applied to increase to magnitude of the flow vector depending on the level of processed features in the pyramid. With rescaling, the flow vectors from low levels are aligned with the flow vectors in the actual pyramid level (level=6). As a final step of the layer, the predicted flows are appended into "Flows Dictionary" sub-module for storage.

After forward propagation of all features for each pyramid layer through “Flow Model”, the final prediction of flows are stored in “Flows Dictionary” until the full resolution predicted flows of dimension of 1x2x250x250 are found with the application of “Upsample 2d_as” sub-module and “1/div_flow” sub-module, as a last flow alignment, at level 6.

In the meantime, to calculate the total flow loss, “Flows Dictionary” is sent to “Flow Loss” module. For each layer, ground truth flows with full resolution (1x2x250x250), $target_flow_f$ and $target_flow_b$ are downsampled to the resolution of $predicted_flow_f$ and $predicted_flow_b$, (1x2xD_lxD_l), and then L2 norm or EPE is used to calculate the loss for that specific layer as

$$EPE = \sqrt{(target_flow - predicted_flow)^2}. \tag{B.1}$$

Total flow loss is derived via the summation of flow loss for each layer by considering the losses for each resolution of the forward and backward predicted flow vectors.

”Occlusion Model”, shown in Figure B.5, is almost same with “Flow Model” except that the number of channels in occlusion tensors, occ_f and occ_b (1x1xD_lxD_l) are the one, and rescaling sub-modules are not used. That is why no further explanation is included to explain the forward propagation process of the occlusion detection process of “Occlusion Model”. Forward propagation of “Flow Model” is also valid for “Occlusion Model”.

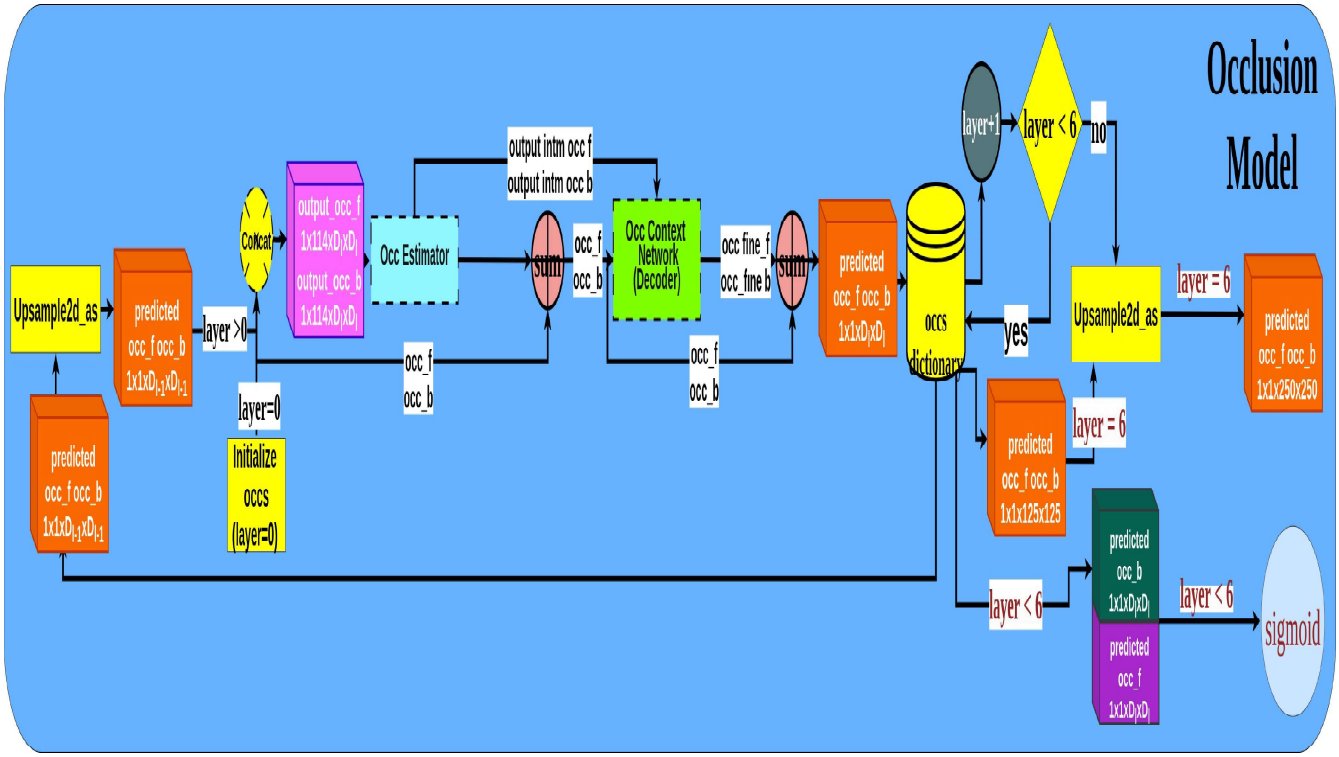


Figure B.5. Occlusion model.

After forward propagation of all features for each pyramid layer through "Occ Model", the final prediction of occs, *predicted_occ_f* and *predicted_occ_b* of the dimension of 1x1xD_lxD_l are stored in "Occs Dictionary" until the full resolution predicted occs of the dimension of 1x1x250x250 are found with the application of "Upsample 2d_as" sub-module. To calculate the total occ loss, "Occs Dictionary" is sent to "Occ Loss" module. For each layer, ground truth occs with full resolution (1x1x250x250), *target_occ_f* and *target_occ_b* are downsampled to the resolution of *predicted_occ_f* and *predicted_occ_b*, (1x1xD_lxD_l), and then average F1 score ($F1_{avg}$) is used to calculate the loss for that specific layer as

$$\begin{aligned} true_{pos} &= -\sum target_occ * log(pred_occ) \\ false_{neg} &= -\sum (1 - target_occ) * log(1 - pred_occ) \\ denom_{true_{pos}} &= \sum target_occ + \sum pred_occ \\ denom_{false_{neg}} &= \sum (1 - target_occ) + \sum (1 - pred_occ) \\ F1_{avg} &= (\sum (\frac{true_{pos}}{denom_{true_{pos}}}) + \sum (\frac{false_{neg}}{denom_{false_{neg}}})) * (250 * 250 * 0.5) \end{aligned} \tag{B.2}$$

where $true_{pos}$ and $false_{neg}$ represent the true positive and false negative values, calculated using the ground truth and predicted occlusion maps. Total occ loss is derived via the summation of occ loss for each layer by considering the losses for each resolution of the forward and backward predicted occlusion maps.

Then, as a final step of forward propagation, total flow loss and total occlusion loss (occ loss) are summed to find the total loss corresponding to all layers for each batch to be able to update the parameters in the "Flow Model" and "Occlusion Model" simultaneously in terms of both losses. Hence, each model is not trained only by its own error model, and the total error, a combination of both errors, provides the chance to update both models in correlation. This process is performed for all images in the dataset based on the chosen epoch and the number of batches.

"Flow Model" and "Occlusion Model", trained/fine-tuned using S-UAV dataset and UAV123 [2], are utilized to extract forward and backward optical flow and occlusion maps in the PFCNNOcc tracking method, as will be explained in Section 5.3. The same

models are tailored during tracking in both S-UAV [10] and UAV-123 [2] datasets.

B.2. Training of the CNN Model for Segmentation

"Segmentation Model" is designed similarly to the U-Net model described in [60], as shown in Figure B.6. The first part of the model acts as an encoder, extracting features by reducing the original image dimensions and increasing the number of channels until it reaches "Output_5" (2x1024x15x15). Then, in the second part of the model, the processing of the reduced features concatenated with residual tensors increases the tensor dimensions to "Output_up_4" (2x64x250x250). Therefore, the second part of this model acts as a decoder. The final output of the decoder is fed into the "Conv2D" module to find the "Predicted Mask", providing predicted segmentation tensors of size 2x2x250x250 with the required number of channels or classes for two-input images. The number of classes is taken as 2; vehicle and background. The first channel of the tensor represents the background, and the second channel represents the segmented pixels of the vehicle. After "Predicted Mask" is separated into "im1 Predicted Mask" and "im2 Predicted Mask" tensors of size 1x2x250x250, the softmax function is applied to the segmented tensors to create "im1 Softmax Predicted Mask" and "im2 Softmax Predicted Mask" tensors. The total segmentation loss is calculated by summing cross-entropy loss and dice loss as

$$\begin{aligned} seg_loss &= CrossEntropyLoss(predicted_mask, target_mask) \\ dice_loss &= 1 - \frac{2 * \sum(softmax_predicted_mask * one_hot_target_mask)}{\sum softmax_predicted_mask + \sum one_hot_target_mask} \\ total_seg_loss &= seg_loss + dice_loss \end{aligned} \tag{B.3}$$

where cross-entropy loss is found using predicted masks and target masks of input images in a batch, while dice loss is calculated by tailoring the softmax of predicted masks and one-hot function of target masks. At the end of forward propagation, to train or update the parameters in "Segmentation Model", the *total_seg_loss* is backpropagated using the chosen optimizer.

”Segmentation Model” in Figure B.6, trained using the S-UAV dataset, is utilized to find masks of segmented vehicles of input images for the PFCNNOcc tracking method in the S-UAV [10] dataset. Because of the additional features, arose from the types and shapes of vehicles in UAV-123 [2] real-life dataset, “Segmentation Model” in Figure B.6 is re-trained to update all learnable parameters using the ground truth dataset of segmented vehicles of UAV-123, as explained in Section 4.3. The size of the targets in the UAV 123 dataset [2] varies depending on the sudden scale changes and the altitude of the UAV, recording the scene during the dataset creation. Even in some scenarios, the width or the height of the targets is bigger than the size of the ROI, which is 250x250. In some of the scenarios, the target is getting very small compared to the targets in the S-UAV dataset [10]. Hence, to train the “Segmentation Model” using the UAV123 dataset effectively, an intense and adaptive data augmentation is applied to the UAV123 dataset. After reading the size of a target from the ground truth, if its width and height are smaller than 75 pixels, its image is upscaled by 2. Otherwise, if the width or the height of the target is bigger than 200 pixels, its image is downscaled by 2 to be able to fit within the ROI during training and tracking. Moreover, an additional affine transform, constructed by the combination of rotation, translation, and scaling, is applied to all images in the dataset. This adaptation of the UAV123 dataset is one of the most important features that increases the success of training, and adapting the dataset into the “Segmentation Model”.

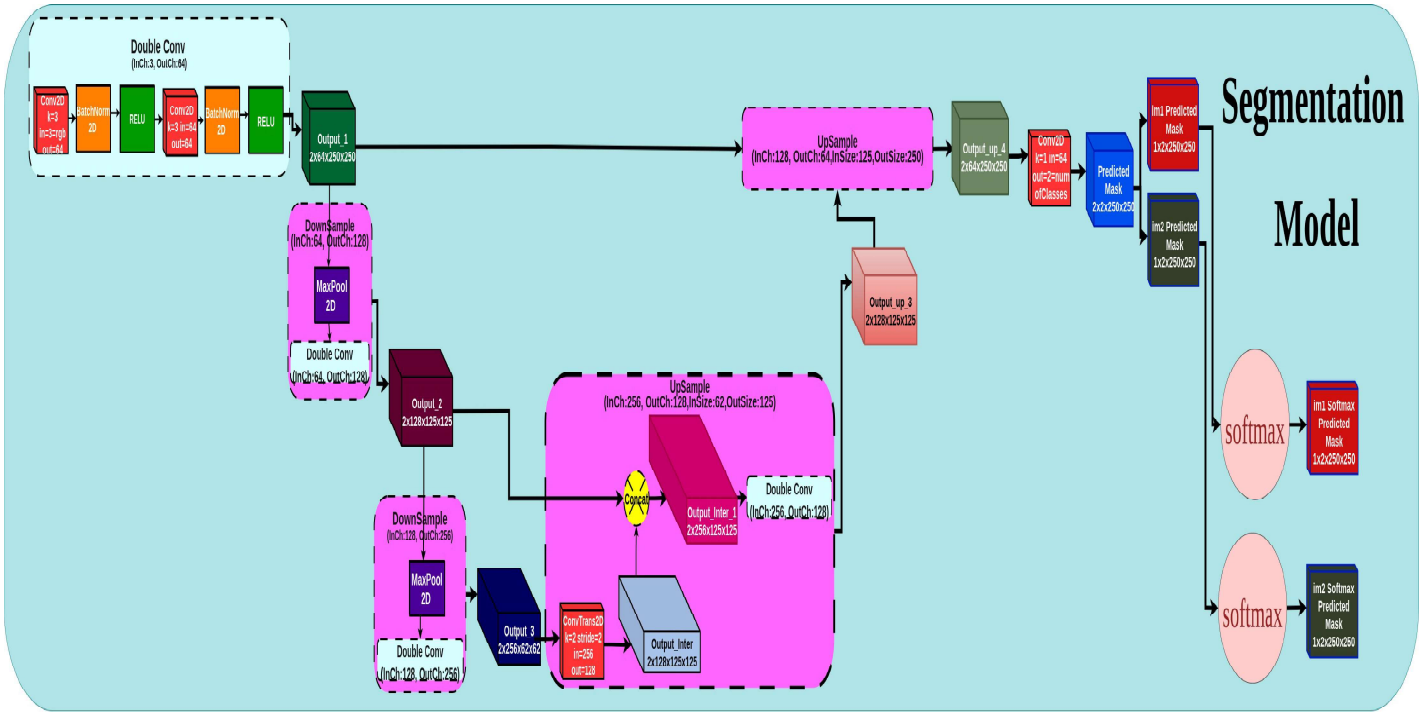


Figure B.6. Segmentation model.

B.3. Training of the CNN Model for Classification

The "Classifier Model" is inspired by the VGG16 model described in [87], and is shown in Figure B.7. In the beginning of the training of "Classifier Model", for each iteration, two images as anno1 (1x3x224x224) and anno2 (1x3x224x224) are formed by consecutively cropping and resizing, im1 (1x3x250x250) and im2 (1x3x250x250) using bounding box ground truth data from the S-UAV dataset [10]. And then anno1 and anno2 are concatenated to form 2x3x224x224 tensor to be processed by the model. As understood from Figure B.7, "Classifier Model" acts as an "encoder" until Output_15 (2x512x14x14) is extracted while building high-level features via increasing the number of channels of features at each level, and decreasing tensor resolution along the width and height of tensors. Then, Output_19 (2x512x14x14) is max-pooled and linearized to create Output_20 (2x4096), which has a 1x4096 vector feature for each input image. At the end of the forward propagation, prediction scores (logits) for five classes (bus, car, lorry, truck, van) are generated for each bounding box as shown in Figure B.7. During the training of the model with the UAV dataset [2], the predictions are made for two classes: 1. car, 2. background. To be able to update all learnable parameters of the model, the class loss is calculated using cross-entropy between the computed scores and the actual reference class data (target class). To train or update the parameters in "Classifier Model", the class loss is backpropagated using the chosen optimizer at the end of forward propagation.

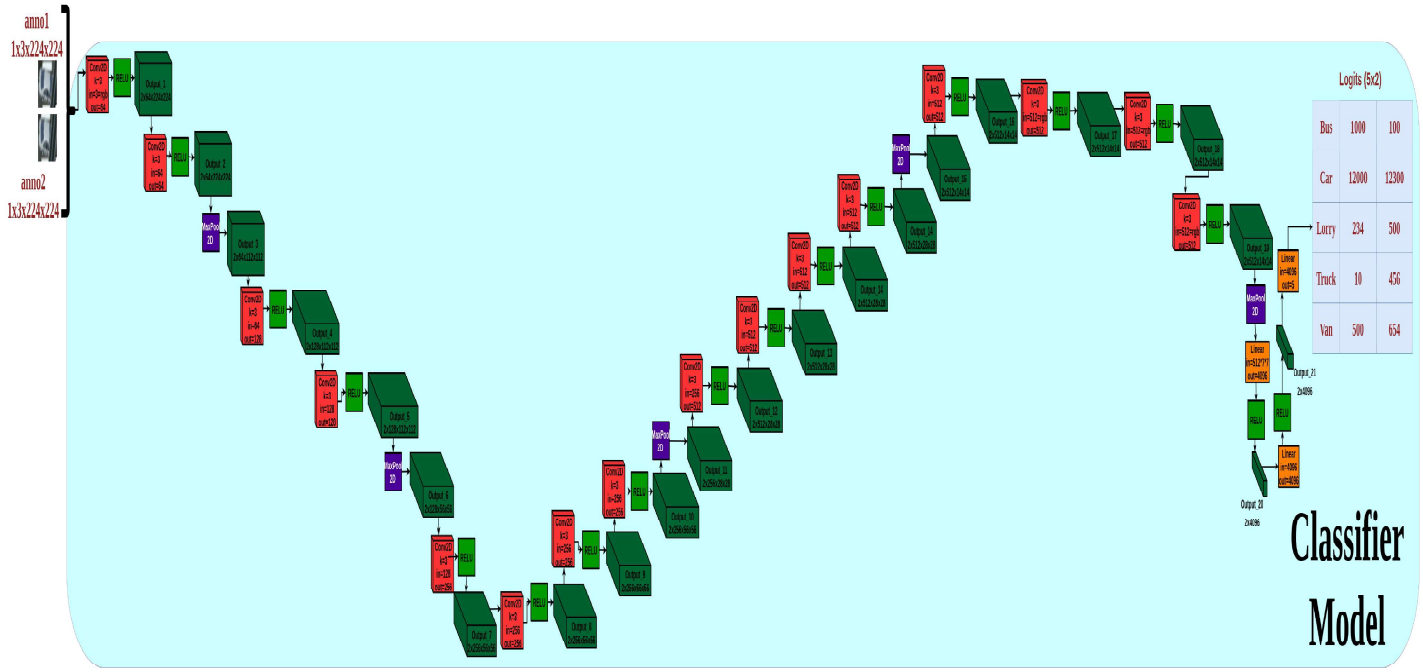


Figure B.7. Classifier model.