Self-correcting Bayesian target tracking

by

Tewodros Atanaw Biresaw

BSc in Electrical Engineering 2005

MSc in Advanced Robotics 2010

A dissertation submitted to

University of Genova

and

Queen Mary University of London

in partial fulfilment of the requirements for the Degree of

Doctor of Philosophy

in the subject of

Interactive and Cognitive Environments

University of Genova - Italy

Queen Mary University of London - UK

December 2014

Acknowledgements

This PhD Thesis has been developed in the framework of, and according to, the rules of the Erasmus Mundus Joint Doctorate on Interactive and Cognitive Environments EMJD ICE [FPA n° 2010-0012] with the cooperation of the following Universities:

Alpen-Adria-Universität Klagenfurt – AAU

Queen Mary, University of London – QMUL

Technische Universiteit Eindhoven – TU/e

Università degli Studi di Genova – UNIGE

Universitat Politècnica Catalunya – UPC

According to ICE regulations, the Italian PhD title has also been awarded by the Università degli Studi di Genova.

Acknowledgements

I would like to thank first my supervisors Professor Carlo S. Regazzoni and Professor Andrea Cavallaro for their continuous support, advice and valuable suggestions that helped me to have this research outcome. A big thank to all present and past people in the ISIP40 group at University of Genova and the AC group at Queen Mary University of London, for the discussions and support that became fundamental parts of my Ph.D.

I would also express my gratitude to my family and friends that always believed, supported and motivated me, even from far away. Special thanks to my parents, Atanaw and Ethenesh, and siblings who have cared for me, my whole life. Dad, you have been an outstanding inspiration to my carrier.

I, Tewodros Atanaw Biresaw, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature:

Date:

UNIGE supervisor | QMUL supervisor | Author

Professor Carlo S Regazzoni | **Professor Andrea Cavallaro** | **Tewodros Atanaw Biresaw**

Self-correcting Bayesian target tracking

Abstract

Visual tracking, a building block for many applications, has challenges such as occlusions, illumination changes, background clutter and variable motion dynamics that may degrade the tracking performance and are likely to cause failures. In this thesis, we propose Track-Evaluate-Correct framework (self-correlation) for existing trackers in order to achieve a robust tracking. For a tracker in the framework, we embed an evaluation block to check the status of tracking quality and a correction block to avoid upcoming failures or to recover from failures. We present a generic representation and formulation of the self-correcting tracking for Bayesian trackers using a Dynamic Bayesian Network (DBN). The self-correcting tracking is done similarly to a self-aware system where parameters are tuned in the model or different models are fused or selected in a piece-wise way in order to deal with tracking challenges and failures. In the DBN model representation, the parameter tuning, fusion and model selection are done based on evaluation and correction variables that correspond to the evaluation and correction, respectively. The inferences of variables in the DBN model are used to explain the operation of self-correcting tracking. The specific contributions under the generic self-correcting framework are correlation-based self-correcting tracking for an extended object with model points and tracker-level fusion as described below.

For improving the probabilistic tracking of extended object with a set of model points, we use Track-Evaluate-Correct framework in order to achieve self-correcting tracking. The framework combines the tracker with an on-line performance measure and a correction technique. We correlate model point trajectories to improve on-line the accuracy of a failed or an uncertain tracker. A model point tracker gets assistance from neighbouring trackers whenever degradation in its performance is detected using the on-line performance measure. The correction of the model point state is based on the correlation information from the states of other trackers. Partial Least Square regression is used to model the correlation of point tracker states from short windowed trajectories adaptively. Experimental results on data obtained from optical motion capture sys-

tems show the improvement in tracking performance of the proposed framework compared to the baseline tracker and other state-of-the-art trackers. The proposed framework allows appropriate re-initialisation of local trackers to recover from failures that are caused by clutter and missed detections in the motion capture data.

Finally, we propose a tracker-level fusion framework to obtain self-correcting tracking. The fusion framework combines trackers addressing different tracking challenges to improve the overall performance. As a novelty of the proposed framework, we include an online performance measure to identify the track quality level of each tracker to guide the fusion. The trackers in the framework assist each other based on appropriate mixing of the prior states. Moreover, the track quality level is used to update the target appearance model. We demonstrate the framework with two Bayesian trackers on video sequences with various challenges and show its robustness compared to the independent use of the trackers used in the framework, and also compared to other state-of-the-art trackers. The appropriate online performance measure based appearance model update and prior mixing on trackers allows the proposed framework to deal with tracking challenges.

Contents

Acknowledgements **3**

Abstract **5**

Publications **10**

Glossary of abbreviations **11**

Glossary of symbols **13**

1 Introduction **18**

1.1 Motivation 18

1.2 Contributions 21

1.3 Organisation of the thesis 22

2 Related work **24**

2.1 Introduction 24

2.2 Appearance models 26

2.3 Motion models 27

2.4 Searching and matching strategies 28

2.5 Performance measures 28

2.5.1 Trajectory and feature properties 29

2.5.2 Model validation 30

2.5.3 Multiple states 31

2.6 Correction 32

2.6.1 Adaptive modelling 32

2.6.2 Re-initialisation 32

2.6.3 Multiple states 34

2.7 Fusion in tracking 35

2.7.1 Selection of units to fuse 35

2.7.2 Fusion methods . . . 38
2.7.3 Component-level fusion . . . 38
2.7.4 Tracker-level fusion . . . 39
2.8 Discussion . . . 41

3 Dynamic Bayesian Network framework for self-correcting tracking 43
3.1 Introduction . . . 43
3.2 Bayesian tracking . . . 44
3.3 Dynamic Bayesian Network model of self-correcting tracking . . . 45
3.3.1 DBN model . . . 45
3.3.2 Inference: filtering equations . . . 46
3.4 Approximate inference using the TEC framework . . . 48
3.4.1 Estimation of the evaluation variable . . . 48
3.4.2 Estimation of the correction variable . . . 49
3.4.3 Approximate filtering equations . . . 49
3.5 Tracking correction . . . 49
3.6 Coupled DBN model of interacting units . . . 51
3.7 Summary . . . 53

4 Correlation-based self-correcting tracking 55
4.1 Introduction . . . 55
4.2 Extended object tracker . . . 56
4.3 Quality measure . . . 58
4.3.1 Addressing performance degradation . . . 59
4.3.2 Strong and weak trackers . . . 62
4.4 Correction . . . 62
4.4.1 Partial Least Square regression . . . 63
4.4.2 Correlated trackers selection . . . 65
4.4.3 Correction . . . 68
4.5 Discrete variables of the DBN model . . . 68
4.6 Experimental results and analysis . . . 69
4.6.1 Experimental setup . . . 71

4.6.2 Evaluation measures . . . 73
4.6.3 Discussion . . . 75
4.7 Summary . . . 81

5 Tracker-level fusion for robust tracking **84**
5.1 Introduction . . . 84
5.2 Performance measure . . . 85
5.3 Fusion . . . 88
5.3.1 Prior state correction . . . 88
5.3.2 Estimated-state fusion . . . 91
5.3.3 Computational complexity . . . 91
5.4 Visual trackers . . . 92
5.4.1 Colour-Histogram-based Particle Filter (CHPF) . . . 92
5.4.2 Least Soft-thresold Squares (LSS) tracker . . . 93
5.5 Appearance model update . . . 94
5.6 Discrete variables of the DBN model . . . 95
5.7 Experimental results and analysis . . . 97
5.7.1 Experimental setup . . . 97
5.7.2 Parameters . . . 99
5.7.3 Evaluation measures . . . 100
5.7.4 Discussion . . . 101
5.8 Summary . . . 108

6 Conclusions **110**
6.1 Summary of achievements . . . 110
6.2 Future research directions . . . 112

Bibliography **114**

Publications

Journal papers

[J1] T. A. Biresaw, A. Cavallaro and C. S Regazzoni. Correlation-based self-correcting tracking. *Neurocomputing*, DOI: http://dx.doi.org/10.1016/j.neucom.2014.10.057, available online 4 November 2014.

[J2] T. A. Biresaw, A. Cavallaro and C. S Regazzoni. Tracker-level fusion for robust Bayesian visual tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, DOI: http://dx.doi.org/10.1109/TCSVT.2014.2360027, available online 24 September 2014.

Conference papers

[C1] T. A. Biresaw, M. S. Alvarez and C. S Regazzoni. Online failure detection and correction for Bayesian sparse feature-based object tracking. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance*, Klagenfurt, August 30 - September 2, 2011.

[C2] T. A. Biresaw and C. S Regazzoni. A Bayesian network for online evaluation of sparse features based multitarget tracking. In *Proc. of IEEE Int. Conf. on Image Processing*, Orlando, September 30 - October 3, 2012.

Glossary of abbreviations

AFT	Adaptive Fragment based Tracker	97
CDBN	Coupled Dynamic Bayesian Network	51
CHPF	Colour-Histogram based Particle Filter	92
CLF	Component-Level Fusion	35
CMU	Carnegie Mellon University dataset	71
CPM	Change Point Models	45
CT	Compressive Tracking	97
DBN	Dynamic Bayesian Network	43
FCT	Fast Compressive Tracking	97
HDAF	Hungarian Data Association Filter	70
HSV	Hue Saturation Value colour space	26
IMM	Interactive Multiple Model	38
JPDAF	Joint Probabilistic Data Association Filter	58
KLT	Kanade-Lucas-Tomasi tracker	19
LSS	Least Soft-thresold Squares tracker	93
L1T	ℓ_1 minimisation based Tracker	97
MAP	Maximum A Posteriori estimation	45
MMS	Minimum Mean Square estimation	45
MMSE	Minimum Mean Square Error estimation	58
MTT	Multi-Task sparse learning Tracker	97
NNDAF	Nearest Neighbour Data Association Filter	70
PDAF	Probabilistic Data Association Filter	58
PGM	Probabilistic Graphical Models	44
PLS	Partial Least Square	63
PT	Particle filter Tracker	71
OSPA	Optimal Sub-Pattern Assignment	73

RGB	Red Green Blue colour space	26
SA	Similar Appearance tracker	71
SLDS	Switching Linear Dynamical System	45
SSD	Sum-of-Squared-Differences	31
TEC	Track-Evaluate-Correct	19
TLD	Tracking-Learning-Detection	33
TLF	Tracker-Level Fusion	35
VTS	Visual Tracking by Sampling	97
3DDCT	3D Discrete Cosine Transform tracker	97
2D-c	2D motion capture dataset with Clutter	72
2D-m	2D motion capture dataset with Misdetections	72
3D-c	3D motion capture dataset with Clutter	72
3D-m	3D motion capture dataset with Misdetections	72

Glossary of symbols

I_k	Image at frame k	44
K	Duration of frames in image sequence	44
$\mathbf{I}$	Image sequence of K frame numbers	44
$\mathbf{x}_k$	Estimated state of a target at frame k	44
$\mathbf{z}_k$	Measurement taken at frame k	44
$\mathbf{Z}_k$	Set of measurements upto frame k	44
$p(x)$	Probability of a random variable x	44
$p(x\|z)$	Probability of a random variable x given random variable z	44
$\hat{\mathbf{x}}_k$	Best estimated state from the distribution $p(\mathbf{x}_k\|\mathbf{Z}_k)$	45
$E(\cdot)$	Expectation operator	45
$\mathbf{p}_k$	Performance evaluation variable at frame k	45
$\mathbf{c}_k$	Correction variable at frame k	46
p	Value of performance evaluation variable $\mathbf{p}_k$	46
c	Value of correction variable $\mathbf{c}_k$	46
$\mathbb{N}^0$	Whole numbers	46
$\hat{c}$	Value of variable $\mathbf{c}_{k-1}$, at frame $k-1$	47
$\hat{\mathrm{p}}$	Value of variable $\mathbf{p}_{k-1}$, at frame $k-1$	47
f_{p}	Probability density function of evaluation variable	48
I_p	Information used by the performance measure function	48
f_c	Probability density function of correction variable	49
T	Tracker	50
$\Theta(\cdot)$	Correction function for a tracker and estimated state	50
I_c	Information used in the correction function $\Theta(\cdot)$	50
$\overline{T}(\cdot)$	Corrected tracker	50
$\overline{\mathbf{x}}_k$	Corrected state of a target at frame k	50
f_{Θ_m}	Corrected probability density estimator for motion model	50

f_{Θ_o}	Corrected probability density estimator for appearance model	50
f_{Θ_p}	Corrected probability density estimator for prior density	51
f_c^i	Probability density function of correction variable for unit i	52
$\mathbf{x}_k^i$	Estimated state of local tracker i at frame k	56
$\mathbb{N}^+$	Positive natural numbers	56
N_k	Number of model points	56
$\widehat{\mathbf{z}_k^i}$	The $\widehat{i^{th}}$ measurement at frame k	56
M_k	Number of model point measurements	56
T^i	Tracker i	56
$x_{j,k}^i$	Position component of j^{th} coordinate in the state $\mathbf{x}_k^i$	57
$\dot{x}_{j,k}^i$	Velocity component of j^{th} coordinate in the state $\mathbf{x}_k^i$	57
$\mathcal{N}(A,B)$	Gaussian probability density with mean A and covariance B	57
F_k	State transition matrix in the prediction model of Kalman filter	57
Q_k	Covariance matrix of the prediction model in the Kalman filter	57
H	Observation matrix in the observation model of Kalman filter	57
R_k	Covariance matrix of the observation model in the Kalman filter	57
I_D	Identity matrix of dimension D	57
$\otimes$	Tensor product operator for matrices	57
P_g	Gate probability of a Gaussian distribution	57
$\mathrm{p}_{c,k}$	Track quality measure for covariance error at frame k	59
$\mathrm{p}_{o,k}$	Track quality measure for absence of measurements at frame k	61
$\mathrm{p}_{s,k}$	Track quality measure for coalescence with other trackers at frame k	62
$\mathbf{w}_k$	List of indices for weak trackers at frame k	63
$\mathbf{s}_k$	List of indices for strong trackers at frame k	63
w_i	Index of the i^{th} weak tracker	64
s_j	Index of the j^{th} strong tracker	64
$N_{\mathbf{w}_k}$	Number of weak trackers at frame k	64
$N_{\mathbf{s}_k}$	Number of strong trackers at frame k	64
$\mathbf{\Gamma}_k^{w_i}$	Trajectory of a weak trackers w_i at frame k	64
$\mathbf{\Gamma}_k^{s_j}$	Trajectory of a strong trackers s_j at frame k	64

m	Trajectory length	64
$\boldsymbol{\beta}_k^{w_i s_j}$	PLS correlation model for a weak tracker w_i with a strong tracker s_j at frame k	64
U	Component matrices of predictor variable	64
V	Component matrices of response variable	64
$\mathbf{b}_k^{w_i s_j}$	Translation vector component of $\boldsymbol{\beta}_k^{w_i s_j}$	64
$\mathbf{A}_k^{w_i s_j}$	Components of $\boldsymbol{\beta}_k^{w_i s_j}$ without $\mathbf{b}_k^{w_i s_j}$	64
$\mathbf{E}_k^{w_i s_j}$	Training fitting error of the correlation model $\boldsymbol{\beta}_k^{w_i s_j}$	65
$\mathbf{B}_k^{w_i s_j}$	Matrix obtained by concatenating m vectors of $\mathbf{b}_k^{w_i s_j}$	65
$\mathbf{C}_k^{w_i s_j}$	Covariance matrix of the PLS correlation model at frame k	65
$\overline{\mathbf{x}}_k^{w_i}$	State of a weak tracker w_i estimated from prediction of strong trackers at frame k (Corrected state)	65
$\mathbf{g}_k^{w_i}$	List of strong trackers ordered based on correlation level to a weak tracker w_i at frame k	66
$g_n^{w_i}$	The n^{th} strong tracker from the list $\mathbf{g}_k^{w_i}$	66
γ	Number of strong trackers selected for prediction of a weak tracker state	67
C_T	Threshold covariance of correlation model	67
$\alpha_k^{w_i g_p^{w_i}}$	Weight assigned for the prediction of a weak tracker w_i from the p^{th} strong tracker g^{w_i} at frame k	67
$\widetilde{\alpha}_k^{w_i g_p^{w_i}}$	Normalised weight $\alpha_k^{w_i g_p^{w_i}}$	68
L_m	Misdetection duration	73
N_c	Amount of clutter	73
φ	Uniformly-distributed pseudo-random number generator	73
$\mathbf{o}_k^{\bar{i}}$	Ground-truth state of target $\bar{i}$ at frame k	73
$\mathbf{D}_k$	OSPA score at frame k	74
$d_{\grave{c}}(\cdot)$	Cut-off distance in OSPA	74
$\grave{c}$	Cut-off parameter in OSPA	74
a	Order of OSPA	74
FP	Number of false positive tracks	74
d_{Th}	Threshold distance parameter for the estimation of FP	74

ν	Frame length parameter for the estimation of FP	74
$\mathbf{x}_k^{i,r}$	State of the r^{th} particle from tracker i at frame k	85
$\pi_k^{i,r}$	Weight of the r^{th} particle from tracker i at frame k	85
N_r	Number of particles in a tracker	85
$\delta(\cdot)$	Dirac delta function	86
C_k^i	Covariance matrix of tracker i at frame k	86
$\check{\mathbf{x}}_k^i$	Position mean state vector of particles at frame k	86
(x_k^i, y_k^i)	Centre position of the bounding box in state $\mathbf{x}_k^i$	86
h_k^i	Height of bounding box in the state $\mathbf{x}_k^i$	86
w_k^i	Width of bounding box in the state $\mathbf{x}_k^i$	86
$\widehat{C}_k^i$	Covariance matrix of the centre state (x_k^i, y_k^i) for tracker i at frame k	87
$\widetilde{u}_k^i$	Spatial uncertainty of the covariance matrix	87
u_k^i	Smoothed spatial uncertainty	87
ρ	Smoothing parameter for the spatial uncertainty	87
p_1	Good quality track label	87
p_2	Medium quality track label	87
p_3	Low-quality track label	87
u_{Th_1}	Threshold of spatial uncertainty for good quality track performance	87
u_{Th_2}	Threshold spatial uncertainty for low-quality track performance	87
M	Number of trackers in a fusion framework	89
$\Lambda(\cdot,\cdot)$	Sampling function for particles in a tracker	89
η_{ij}	Weight to determine amount of particles exchange between trackers i and j	89
$\widetilde{\eta}_{ij}$	Normalised weight η_{ij}	89
η^1, η^2, η^3	Weight parameters to determine η_{ij}	90
$\widehat{\mathbf{x}}_k^i$	Best state estimate of tracker i from the distribution $p(\mathbf{x}_k^i \vert \mathbf{Z}_k)$	91
$\widehat{\mathbf{x}}_k^f$	Estimated state of the target by the fusion framework at frame k	91
$\mathrm{O}(\cdot)$	Computational complexity	91
d_B	Bhattacharyya distance	92

p_u	Colour histogram measured from candidate sample	92
q_u	Colour histogram of the reference target model	92
$\mathbf{\grave{D}}$	Dictionary or basic matrix of the target	93
d_{LSS}	Least Soft-thresold Square distance	94
ω_G	Gaussian noise vectors	94
ω_L	Laplacian noise vectors	94
λ	Regularisation constant for estimating d_{LSS}	94
α	Appearance model update parameter	95
O_A	Area overlap score	100
A_o	Bounding box of tracker estimate	100
A_g	Bounding box of ground truth	100
$\lvert \cdot \rvert$	Area estimation operator	100
$\kappa(\tau)$	Lost track ratio with a threshold τ	100
τ	Threshold value for lost track ratio	100
AUC_κ	Area under lost track ratio curve	100
μ	Average value	107
σ	Standard deviation value	107

Chapter 1

Introduction

1.1 Motivation

Tracking, in the computer vision community, refers to estimate state(s) of a target or multiple targets. The state estimation is done based on measurements such as feature points, contours and regions of interest taken from a single camera or multiple cameras. Tracking plays important roles to link low-level image processing and high-level video content analysis in various applications [123]. The estimated states help to locate, identify and determine the dynamic properties of one or many targets from the measurements in time. In surveillance systems, tracking provides vital information in the form of people counting, security and traffic flow analysis [37, 13, 94]. In human-computer interaction, tracking allows machines or entities in the environment to understand commands from humans [135]. Tracking is used in robotics in order to identify the target of interest for interaction and for coordination among multiple robotic agents [82, 95]. In addition to this, tracking is main building block in medical image processing [83], wearable technologies [79] and autonomous navigation [35].

Targets in the scene can undergo challenges such as occlusions, illumination changes, pose changes, photometric changes, variable motion dynamics and be similar to background clutter (Fig. 1.1). These challenges may lead to a performance degradation or tracking failure. Different challenges require different modelling approaches. For example, a target undergoing illumination changes needs a tracker with an illumination invariant or an adaptive appearance model [134], while a target with variable motion dynamics needs a tracker with multiple motion models or that

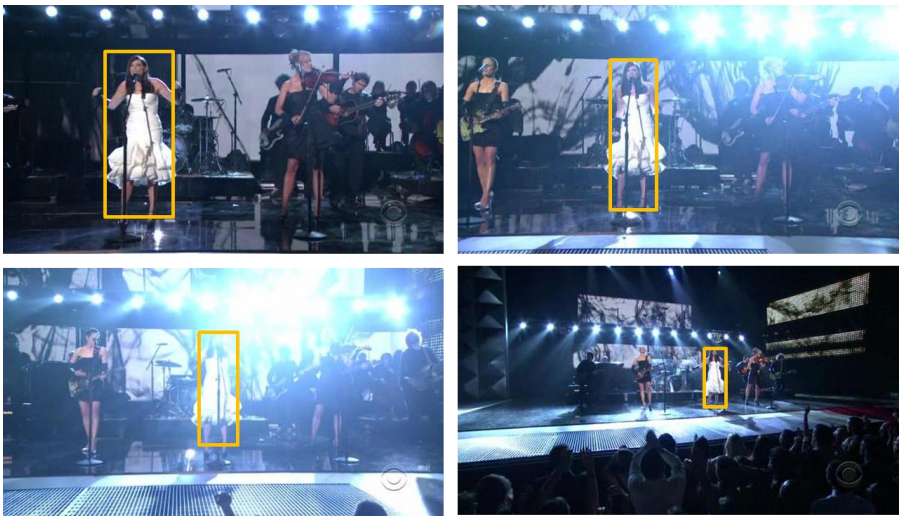


Figure 1.1: Appearance variation of a target in a video acquired by a single camera. The target in the scene, which is shown by the rectangle bounding box, undergoes changes in scale, illumination, shape and view angle together with large amount of variation in background clutter in the different frames.

searches in a large region of the state space [108]. In order to tackle the different challenges, a wide variety of tracking techniques have been proposed such Mean-shift [17], Kalman filter [4], Particle filter [43], Kanade-Lucas-Tomasi (KLT) [62]. In general, the variation between trackers can be in one of the following key components: appearance representation and modelling, motion representation and modelling, and searching and matching strategy [110]. This thesis mainly considers Bayesian trackers (details are presented in Chapter 3).

When different challenges appear simultaneously, a single tracker with a fixed model may be insufficient to achieve satisfactory performance and results in a tracking failure. For a tracker in the incorrect state estimation condition, it is difficult to recover the correct state estimation operation even after the tracking challenges disappear. Tracking failure at any instant of time remains to be a long-term failure due to the use of first-order Markov processes [21]. Except [48, 87, 120, 25, 24, C1], most trackers do not explicitly detect and correct failures or handle tracking challenges. Therefore, a performance measure to detect tracking failures is needed and a tracking correction step is desirable to obtain a robust tracking [15, 61, 16].

Robust visual tracker is characterised by its ability *to avoid* or *to recover from* tracking failures. The former property is known as *ante-failure robustness*, whereas the latter is known as *post-failure robustness* [132]. A possible design approach to avoid and to recover from the failures is a self-correcting tracking, Track-Evaluate-Correct (TEC) framework (Fig. 1.2). As a

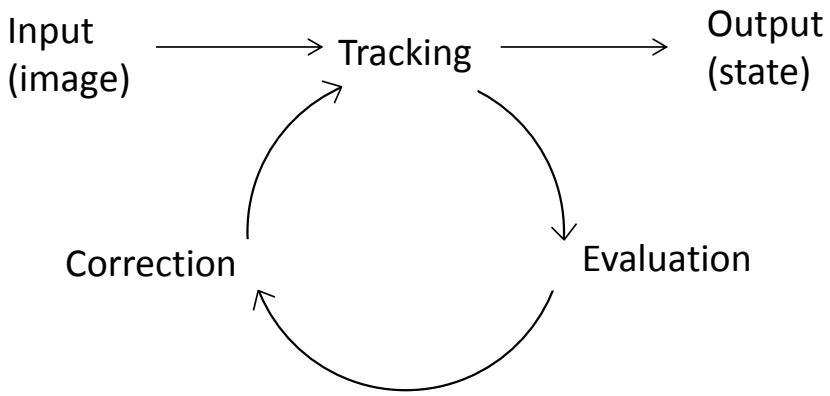


Figure 1.2: Self-correcting tracking, Track-Evaluate-Correct (TEC) framework. Tracking is aided by an online evaluation block to identify challenges and failures, and a correction block to avoid and to recover from failures.

formal definition, a self-correcting tracker is characterised by its state-awareness and tracking model (parameter) correction properties. The state-awareness depends on probabilistic properties of instantaneous knowledge over the state and allows the tracker to reason about its tracking quality. The state-awareness is expressed in the form of quality assessment scores or classification classes. The correction on a tracker is done in accordance to the state-awareness and allows a tracker to actively change one or more of its models (parameters) to resume correct state estimation behaviour. The design of self-correcting framework largely involves how to employ the evaluation and correction blocks together with the existing state-of-the-art tracker.

Performance measures, the criteria to judge the track quality, play an important role for the correction to proceed on the tracker. Since comparing the tracker's output to the ground-truth data is not applicable for real-time systems and the ground-truth data are not available for most applications [91, 64], there is the need for an efficient and robust framework for online track verification [48, 86]. There exist different methods to carry out online the performance measure of a tracker. From the tracker output, trajectory characteristics, feature characteristics, and hybrid of trajectory and feature characteristics are used to obtain the performance results [115, 90]. Corrections are achieved mainly in the form of model update and fusion. Model update allows a tracker to adapt for changes in tracking environment such as illumination changes and amount of clutter [15]. Fusion allows correction of a tracker by switching and combining different tracking components and trackers [132]. In self-correcting tracking, the information to be fused as correction of the tracker are obtained directly from a source running in parallel [48, 112] or gathered online whenever the fusion is required [C1]. Context information in the scene can be exploited

for the fusion to recover tracking failures and verify tracking result [123].

1.2 Contributions

We designed a TEC framework where we embed appropriate evaluation and correction blocks for trackers to obtain robust tracking. A generic formulation of self-correcting tracking is made based on Dynamic Bayesian Network (DBN) in order to examine its operation and different correction methods in a tracker. The detailed contributions of the thesis are:

1. **Generic representation of self-correcting tracking**

   We provide a generic formulation of a self-correcting target tracking using a DBN model. In the DBN model of the self-correcting tracking, discrete variables represented by additional hidden layers are incorporated over a DBN model of a baseline tracker that allow evaluation and correction on a tracker. The discrete variables value identifies the level of track quality and the type of corrections made to a tracker. An approximate inference is made by utilising the TEC framework. To model interacting units in the self-correcting tracking, a coupled DBN model is proposed. The generic DBN formalisation allows us to develop a self-correcting tracking for an intended tracking application.

2. **Improve tracking of model points on an extended object**

   We use the TEC framework to improve Bayesian filtering of model points on an extended object [J1]. We propose a quality measure criterion for evaluation of each model point track to produce a decision for correction. The quality measure is based on examining trackers state and modelling of failure sources. Unlike evaluation methods proposed in [C1, C2, 47, 87], modelling the source of failures enable the tracker to make correction for avoiding expected failures. We used data obtained from optical motion capture systems in order to show the improvement in tracking performance of our proposed method compared to the state-of-the-art methods.

3. **Utilise correlation information for correction**

   We make model point trackers to assist each other based on their correlation model in the TEC framework [J1]. Correction of low-quality model point trackers involves an estimation of a probable true state and a re-initialisation of the tracker using the correlation model with other point trackers. The correlation between point trackers is modelled from observed trajectory histories adaptively. Unlike the method presented in [C1, 120], an online modelling

and a correlated trajectory selection criterion are used for effectively recovering low-quality trackers.

4. **Tracker-level fusion for self-correcting tracking**

   We propose self-correcting tracking using tracker-level fusion of complementary trackers [J2]. In the fusion framework, we include online performance measures for individual trackers as enabling factor for tracker-level fusion. Most existing methods enable the fusion by considering the measurement likelihood for score or reliability assessment [53, 132, 128, 51, 127]. However, measurement likelihoods are affected by scene clutter and are dependent on the discriminative capability of the target appearance modelling [89].

5. **Collaboration strategy for tracker-level fusion**

   We define a tracker collaboration strategy in the tracker-level fusion using prior states (prior correction) based on the performance measures of trackers [J2]. The basic idea of the prior correction is to improve a tracker with low-quality performance. A poorly performing tracker prior is partially or fully replaced with the prior of a well-performing tracker according to their relative performance levels. In addition to this, we set a criterion for the appearance model update for trackers based on the performance measure to minimise drifting.

1.3 Organisation of the thesis

This thesis is organised as follows:

Chapter 1: The introduction and motivation for the thesis are described in Sec. 1.1 and followed by discussion of the contributions in Sec. 1.2.

Chapter 2: The introduction of the chapter is provided in Sec. 2.1. Related works on a tracking system are described in Sec. 2.2, Sec. 2.3 and Sec. 2.4. Performance measures and correction methods for target tracking are discussed in Sec. 2.5 and Sec. 2.6. Data fusion methods for tracking are described in Sec. 2.7. Discussions on related work to obtain self-correcting tracking are presented in Sec. 2.8.

Chapter 3: The introduction of the chapter is provided in Sec. 3.1. Tracking of targets in Bayesian framework is explained in Sec. 3.2. The proposed DBN model and filtering equations

for self-correcting tracking are presented in Sec. 3.3, Sec. 3.4 and Sec. 3.6. Based on the DBN model, tracking corrections are explained in Sec. 3.5. Summary of the chapter is given in Sec. 3.7.

Chapter 4: The introduction of the chapter is given in Sec. 4.1. The proposed TEC framework based probabilistic tracking of an extended object with set of model points is presented in Sec. 4.2, Sec. 4.3, Sec. 4.4 and Sec. 4.5. Experimental analysis and validation, and comparison with other state-of-the-art methods are provided in Sec. 4.6. The chapter is summarised in Sec. 4.7.

Chapter 5: The chapter is introduced in Sec. 5.1. The proposed tracker-level fusion framework is described in Sec. 5.2 and Sec. 5.3. Trackers and their appearance model update in the fusion framework are explained in Sec. 5.4 and Sec. 5.5. The DBN model discrete variables for the fusion framework are described in Sec. 5.6. Experimental analysis and validation, and comparisons to other-state-of-the art methods are discussed in Sec. 5.7. Summary of the chapter is given in Sec. 5.8.

Chapter 6: Summary of achievements and future research directions of the thesis work are presented in Sec. 6.1 and Sec. 6.2.

Chapter 2

Related work

2.1 Introduction

Tracking involves estimating the state of a target using measurements taken from a camera. For the state estimation there exist numerous trackers with different formulations. The trackers might encounter failure when the target of interest undergoes challenges such as occlusions and photometric changes. In order to allow trackers to avoid or recover from failures, a self-correcting tracking framework can be considered. The self-correcting framework requires an online performance measure for track quality assessment and a correction for correcting a tracker in the case of tracking challenges and failures (Fig. 1.2). In this chapter, reviews on the state-of-the-art for tracking methods together with associated performance measures and correction techniques are presented. Table 2.1 gives the characteristics of trackers that use performance measures and/or correction techniques. Reviews on key components of a tracking system are presented in Sec. 2.2, Sec. 2.3 and Sec. 2.4. Related works on performance measures for track quality assessment are discussed in Sec. 2.5. In Sec. 2.6, state-of-the-art trackers using correction techniques for achieving robust tracking are presented. Sec. 2.7 presents data fusion techniques in tracking as a means of enabling self-correcting tracking. Finally, Sec. 2.8 provides a discussion of the presented literature.

Table 2.1: Comparison of trackers with performance measure and correction technique. State representations P: point based, A: area based and O: other, "-": not mentioned or not explicitly stated, JPDAF: Joint Probability Data Association Filter.

Ref.	Tracker	State	Performance measure	Correction method	Comments
[91]	Unscented Kalman Filter	P	detection performance, track quality and track accuracy (innovation error)	-	relies on sensor's measurement accuracy
[C2]	JPDAF [97]	P	trajectories correlation between trackers	-	states of trackers providing correlation information are not checked a priori
[89]	Particle Filter	A	spatial state uncertainty and time-reversed constraint	-	applied to trackers using multiple hypotheses strategy
[105]	Probabilistic Trackers	P	model validation, change detection	-	largely dependent on the system characteristics
[14]	Any Tracker	A	log polar transformation of colour changes	-	sensitive to illumination changes and background colour
[124]	Mean-Shift	A	-	correlation information	correlation information fused with no stated criteria
[86]	Particle Filter	A	-	appearance model update	update performed with no performance measure of the tracker
[134]	Particle Filter	A	-	appearance and motion model update	update done with no performance measure of the tracker
[103, 40]	Matching	P	feature detection quality	birth and death of features	spatial information is not preserved
[48]	Median-Flow [47]	P&A	comparison of displacement changes between features and object states	re-initialise from online learned detector knowledge	largely dependent on the accuracy of the detector, expensive for multi-target tracking
[47]	KLT	P	time-reversed constraint	remove failure features	spatial information is not preserved
[87]	Mean-Shift	A	compare to first detected model	information from detector	evaluation not robust for appearance changes
[C1]	JPDAF [97]	P	time-reversed constraint	trajectory correlation between trackers	non-adaptive correlation model
[120]	Particle Filter	O	misdetection	correlation information	correlation model learned off-line
[50]	Particle Filter	A	trained classifier for occlusion detection	use knowledge of unoccluded region	detecting of occlusions is used as performance measure
[J1]	JPDAF [97]	P	predefined failure models and tracking challenges	trajectory correlation between trackers	performance measure based on the baseline tracker, adaptive correlation modelling for correction

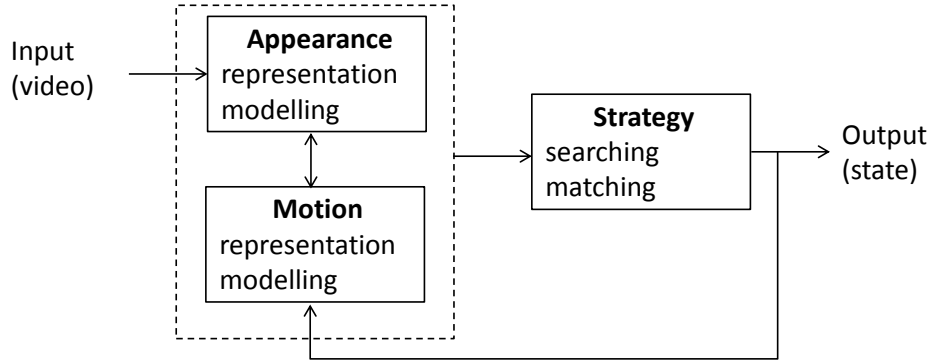
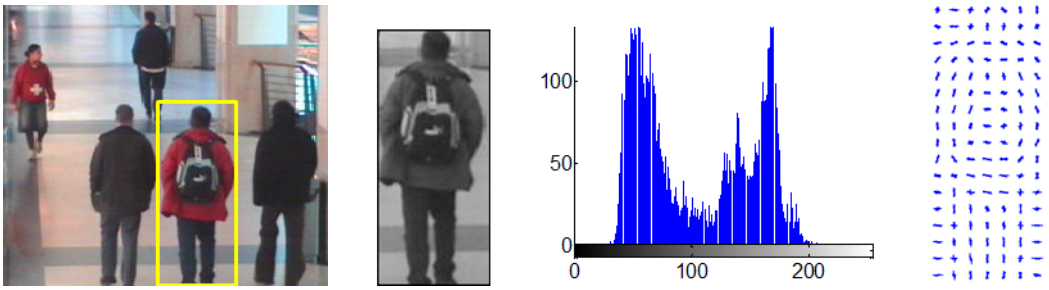


Figure 2.1: A generalised pipeline of a visual tracker with key components.

2.2 Appearance models

As the core of a tracking system (Fig. 2.1), the target's representation and appearance model play an important role in visual tracking [64, 96]. The target representation can be in the form of a point, set of points, parts, a contour and a bound box [125, 64, 96]. For the intended representation, the appearance of the target is measured in the image in the form of pixel values and their properties, and is represented by using schemas such as template [10], histograms [76] and feature vectors [127] (Fig. 2.2). The set of pixels representing a target can be organised either in the form of a global (holistic) descriptor or a local descriptor [99, 12]. Global descriptors identify the target by considering pixels in the overall target, while local descriptors are constructed from specific parts of the target. In both of the descriptors, different properties of pixels can be considered as features for modelling the appearance of the target. In some appearance modelling techniques, the pixels in the background (context-information) can also contribute in order to describe or to discriminate the target in the form of classification task [3, 130]. The classification task uses pixels on the target and on the background as positive and negative samples, respectively.

The features taken from the image can be categorised as low-level, medium-level and high-level features [64]. Low-level features include colours [76], gradients [65] and motion patterns [12, 93]. Colours are directly obtained from the pixels in an image, and different colour spaces (e.g. RGB and HSV) exist to identify the target of interest. Gradients are obtained by taking the difference in the intensity values of pixels on the target (sometimes from background too), while motion patterns are obtained as changes in pixel values between different images in the video, such as optical flow. An image processing technique can be applied to pixels in order to obtain medium-level features such as edges [39], interest points [31] and uniform regions [41].



(a) (b) (c) (d)

Figure 2.2: Target appearance representations: (a) sample image containing target (indicated by yellow bounding box), (b) gray-image template (2D-array), (c) gray-image intensity-based histogram and (d) gray-image intensity-based gradient feature vectors.

High-level features are used to directly identify and discriminate the target of interest from the background by utilising appearance features. Example of high-level features can be extracted from a head-detector used in applications such as crowd counting or density estimation [98].

2.3 Motion models

Target's motion over time is described by a motion model. The motion model helps to reduce computational complexity by predicting the likely state of a target between consecutive frames [53, 110]. Motion prediction is done using constant velocity [12, 65], constant acceleration [128], coordinated -turn [108] and random walk [53, 52] models. The motion model allows the tracker to perform robust tracking when the model properties match with the motion characteristics of the target. The constant velocity, acceleration and coordinated turn motion models result robust in tracking when a target moves behind occlusions. The random walk model is widely used for targets with no predefined motion pattern. Motion models can be adaptive such as an adaptive velocity model with adaptive nose [134]. Similarly, observation data driven motion model is introduced for handling abrupt motion changes of the target [77]. The motion models can be designed and represented to describe the changes in scale, rotation and deformation properties of the target depending on an adopted state representation method. Among the rich motion representations methods, an affine motion model contains the centre, scale, rotation, aspect ratio and the skew angle of a rectangular bounding box [107].

2.4 Searching and matching strategies

Trackers such as Mean-Shift [17] and KLT [62] are designed free of a constrained motion model, rather they implicitly embed the motion model in their search strategies. The search strategy defines a method to find and match the best state of the target in the current frame [96]. The search strategies in tracking can be either deterministic or stochastic. The search strategy is accompanied by a matching method for selecting the best candidate of the target. The search strategy for matching the features of the target from frame to frame can be formulated as an optimisation problem of an objective function. In deterministic methods, the optimisation problem is formulated as differential algorithms, and is solved by using gradient decent and its variants. The Mean-Shift and KLT trackers are examples of deterministic trackers. Stochastic methods use Bayesian formulations to optimise the objective function. Kalman filter [4] and particle filter [43] are the most common Bayesian trackers. Although deterministic methods are computationally efficient and have low complexity, the capability to escape local minimum solutions makes stochastic trackers preferable [110]. Recent advances and an experimental survey on trackers are available in [121, 96, 125].

2.5 Performance measures

Performance measures judge the quality of the tracker either based on the algorithm used or the tracker output (Fig. 2.3) [64]. In analytical methods, the tracker components and the complexity of the algorithm are examined without the need of tracks. In empirical methods, one or more output variables of a tracker are compared with a predefined characteristic and reference data in order to quantify the track quality. The track quality can be quantified either offline or online. For an offline performance measure, manually collected ground-truth data is used as a reference in empirical discrepancy methods [91]. For the case of an online performance measure, the output of the tracker is judged in empirical standalone methods. Online performance measures play an important role in obtaining self-evaluation decision for correcting a tracker. In this thesis we review related works on online performance measure as part of building self-correcting tracking.

Various types of online performance measures have been proposed related to the diverse tracking strategies. For the online evaluation, characteristics of the tracker output include trajectory properties [48, 47, 115], objects colour differences and boundary contrasts with background [14, 26, 69], observation likelihood [91, 105, 69] and innovation errors or covariances

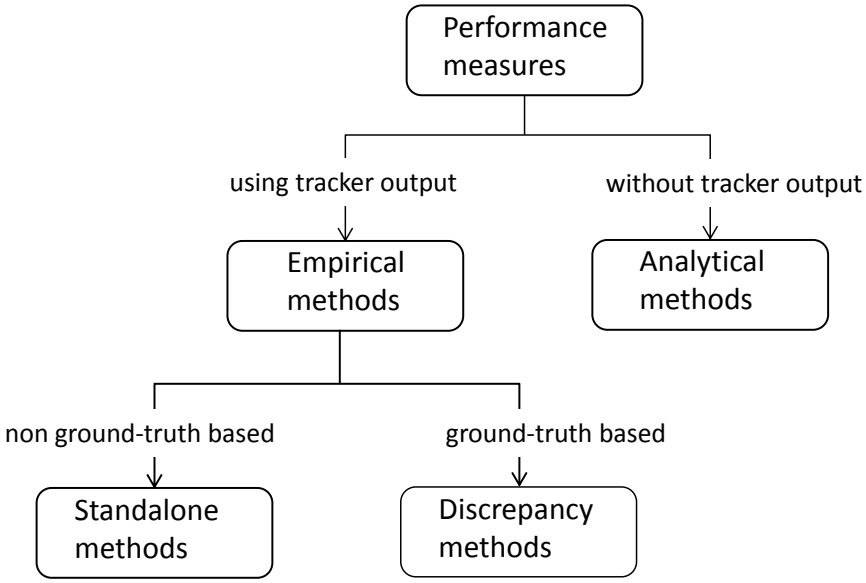


Figure 2.3: Performance measures in object tracking (this figure is an adaptation of [64]). Among the different techniques, Empirical standalone methods are used for track quality assessment for realisation of self-correcting trackers.

of the states [91, 89]. These characteristics are compared with threshold values and predefined target models. Moreover, in the case of fusion of multiple trackers or tracking of an extended object with model points or parts, the multiple estimated states are compared in order to obtain the evaluation [47, 133, C2].

2.5.1 Trajectory and feature properties

Trajectory-based evaluation methods consider properties such as smoothness, length, change of direction, similarity with the predefined model [89, 81] and similarity with a reverse tracking result [115]. Detection of failures for a tracker is a form of performance measure with binary decisions. A failed tracker is assumed to have irregular (random) changes in the tracker output states [116]. The irregularity in the trajectory is measured from a combination of the complexity, the smoothness and the scale consistency scores. The irregularity assumption for detection of failures is a predefined characteristic for the target; however, these characteristics cannot be applied to a generic target variable motion pattern. Trajectory consistency between the forward and the backward track is measured for quality assessment based on the assumption that correct tracking is achieved independent of time direction [47, 115, C1].

Histogram difference between the tracker output and a prior known reference of the target is used as colour-oriented performance measures. In addition to this, properties of temporal colour

differences between tracks are also considered as a performance measure. The occurrence of dramatic change in colour properties of the track has been used to detect failures [42]. Although these performance measures are easy to implement, they can produce false positive results when there are illumination changes [89, 26]. Moreover, the use of colour similarities for guiding the state estimation in the tracker as well as for the evaluation of the tracker at the same time cannot represent appropriate track quality evaluation. Features of colour change in log-polar transformed track image are used for detecting failures instead of comparing colour similarity directly [14]. Authors defined failure of the tracker as the movement of the centre of the bounding box from the target to the background, and this property is shown to be easily measured in the log-polar transformed image.

In multi-face tracking based on a face detector, combinations from static observations on the tracker state and dynamic observations on the temporal evolution of features are used to detect tracking failures [25, 24]. The static observations are made based the properties of the face detector output, trajectory (refereed as tracking memory) and the likelihood with respect to the state estimated. The dynamic observations are based on the changes in the state distribution and observations likelihood over consecutive frames. Abrupt increase and decrease of the dynamic observations are associated with tracking failures.

2.5.2 Model validation

Performance measures are considered equivalent to online model validation of the assumed dynamic system in the tracker such as motion models. Model validation for the probabilistic trackers can be directly estimated from their innovation error, likelihood, covariance magnitudes and their corresponding cumulative sums [105, 115]. However, similar measurements available from other objects or clutter usually produce inaccurate performance measures by using the model validation approaches. Negative expected log likelihood of the states is proposed as a performance measure metric in order to detect particular failures from slow model changes (drifting) [105]. Entropy characteristic of the estimated posterior distribution is also considered for track quality estimation [60]. In particle filter, the entropy is measured from the weights of the samples, and a large value of entropy is associated with poor tracking performance. A more generic and robust performance measure for probabilistic tracker is proposed based on the spatial uncertainty of samples from the posterior distribution [89, 90, 65].

2.5.3 Multiple states

Relativity-based evaluation is used as performance measure for selection and comparison of set of trackers for fusion. In the fusion of multiple trackers, relationship among trackers and majority voting schema are exploited to analyse the relative performance of trackers. Relative likelihoods from trackers are used for obtaining the quality of trackers [52, 126]. The tracker with the highest probability is selected as the best performing tracker or assigned with the highest weight for fusion. Relativity-based evaluation methods avoid the need of a threshold value for failure detections. The difference between results from individual trackers and the fused results from all trackers is examined for identifying failed trackers [133]. A pair-wise track correlation among set of trackers, together with track consistency between two successive frames from individual trackers, is used to rank performance of trackers in order to have an optimal combination [32].

For tracking an extended object by a set of model points and parts, the performance measure is based on the model point's and part's detection and matching quality [103, 40, 69] or their trajectory properties [48, C1, 47]. Individual allocated tracker for each model point or part can be referred as local trackers. Performance measure of extended object track is performed either with the local trackers [47] or with the global-object tracker [48]. A local tracker is considered to have a low-quality performance when no new observations are matched with it [103]. However, matching-based performance evaluation generates false positives and negatives due to similarities between descriptors, clutter and misdetections. In feature points tracker, Sum-of-Squared-Differences (SSD) metric is used on surrounding image patch for the performance measure [74]. Trajectory-based performance measures use time-reversed tracking to obtain a backward trajectory for the comparison with the original one [115]. The differences between the forward and the backward trajectories are used as a performance measure [C1, 47, 115]. However, tracking back to the initial (or previous) frame leads to delays; moreover it is expensive for long-term trackers and for targets with large number of model points or parts, such as markers for the human body in motion capture system [8]. Trajectory correlation among local trackers is exploited in order to obtain the performance of extended target tracking [C2]. From the observed trajectory correlation, reference data is estimated for local trackers using the states of others. The evaluation compares error distances between the output state and the estimated reference data.

2.6 Correction

The correction step involves changing a tracking parameter [134], using a suitable tracking principle over a baseline tracking algorithm [50] in order to deal with challenges. The baseline tracker is a tracker without an explicit modelling of evaluation and correction blocks in it.

2.6.1 Adaptive modelling

Changing of parameters and models helps a tracking system to adapt changes in property over time for the target and tracking environment, and allows occurrence of tracking failures to be minimised [15]. The parameter changing is a form of correction in a tracker which is usually done in accordance with the performance measure. The motion and observation models in a tracker contain the parameters that can be tuned for correct state estimation. Trackers that tune their parameters are referred to as adaptive [44, 101]. In the case of adaptive appearance model, every time the tracker update its appearance model, the update is a form of correction to the predefined model. Adaptive trackers change the tracker model based on the outcome of the most recent frames. It is worth noting that changing a tracker model based on its outputs without any sort of quality measure, can be referred as blind updates, are prone to accumulate errors and cause drifts to background [110, 109]. Considering inter frame modelling changes for motion and appearance in tracking, an adaptive strategy for motion and appearance models are proposed [134]. The motion model uses adaptive velocity with adaptive noise modelling based on first-order predictor of states in consecutive frames. The adaptive appearance model uses a mixture of appearance models obtained from the tracker output with a constraint that past mixture components are exponentially forgotten. Assuming the effectiveness of a strong appearance model compared to a strong motion model in visual tracking, emphasis has been given to adaptive appearance models [86, 44, 36, 101, 88].

2.6.2 Re-initialisation

Tracker re-initialisation is used as a correction in order to recover from its failure [48, 87, 11]. The re-initialisation is a model resting technique that is similar to breaking the Markovian chain in Bayesian trackers [9]. This approach requires a mechanism to obtain the (expected) true state of the target independent of the tracker [48, 87]. Contextual dynamics between a target and features in the background are used for correcting the tracker in the cases of occlusions and/or camera

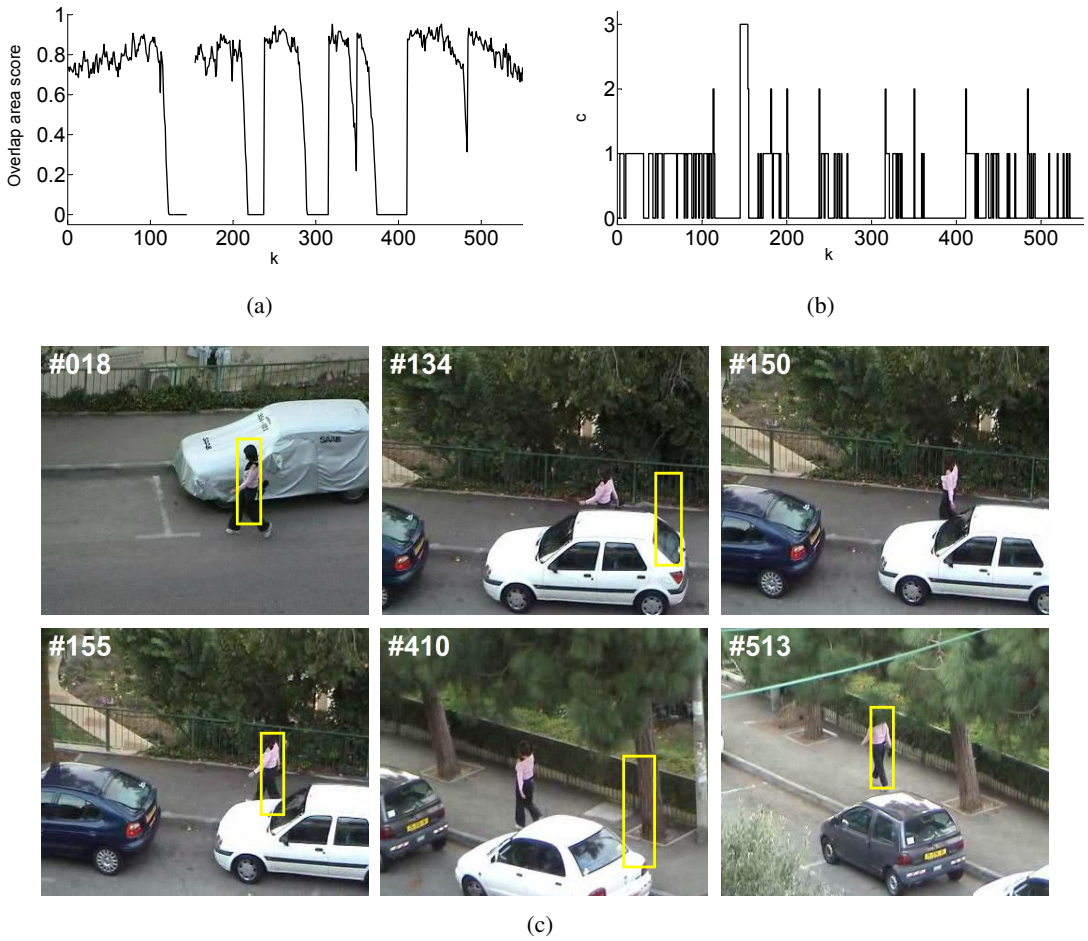


Figure 2.4: Performance of Tracking-Learning-Detection (TLD) framework [48]. (a) Area overlap score (Sec. 5.7.3) of TLD compared to ground truth over frame k. (b) Modes of operation for TLD: $c = 0$- result from tracker alone, $c = 1$ - result from fusion of tracker and detector, $c = 2$- re-initialisation of a tracker and $c = 3$- no result. Failures in a tracker, corresponding to small values of overlap in (a), are corrected by a detector in the re-initialisation mode of operation. (c) Sample tracking results.

motion [119]. Detector has been used for target re-accusation and re-initialisation of trackers in the cases of occlusions in Tracking-Learning-Detection (TLD) framework [48]. The detector is learned online while the tracker is well on the target. The output from TLD can either be from the tracker alone or fusion of the tracker with the detector depending on their performances. TLD has four modes of operations: result from tracker alone, fusion of the tracker with detector, re-initialisation of a tracker from detector and no track estimate (Fig. 2.4). Failures in the tracker are corrected by the detector in the form of re-initialisation.

Similarly to TLD, a face detector has been incorporated with a face tracker for re-initialisation capability of the tracker [87]. In a face detector based multi-face tracking, correction of failed target tracking is done in the form of association with the new initialised targets [25]. The associ-

ation of removed and created targets is done using person identification. The person identification model for correction is built from observations during the tracking process before failure.

2.6.3 Multiple states

In the extended object tracking, the state estimation can be done in the form of local model points and parts. The states of model points and parts combined to estimate the global state of the object. Terminating uncertain or low-quality local track in the extended object is used to achieve robust tracking [47]. The termination process avoids likely tracking failures and incorrect state information by the low-quality local trackers that contribute the state estimation for the global object. This approach usually leads to the loss of structural information of the object when model points are assumed to be a persistent part of the object. The structural information of the object plays an important role as an input for High-level tasks such as activity analysis. Terminating poor performing local trackers can be accompanied with initiating new local trackers [103, 40]. For such tracking method the initiation is another form of correction. The track initiation criteria give focus to the stability of the added new model point. However, it is crucial to consider the structural information to add the new model point for accurate tracking of the extended target.

In a tracking system consisting of multiple trackers, correlation information among the trackers has been used as a correction technique [C1, 124, 120]. In 3D articulated human motion tracking, correlation information is obtained from symmetric portions of the human body [120]. This information is used to estimate the motion prior and constrains the proposal distribution of the particle filter when no measurement data is available. The model used to obtain correlation information is learned offline, which limits the generality of the solution. Although not explicitly stated as a correction technique, a similar type of correlation information is used to obtain robust tracking with a *context-aware tracker* [124]. The source of correlation information is tracking other objects in the scene (auxiliary objects). In the context-aware-tracker the concept of performance measure is not exploited before fusing correlation information from auxiliary objects. Correction in a tracking system is mainly done in the form of fusion. Fusion helps the individual trackers to cooperate each other based on the score assigned to the trackers as a performance measure. In the cooperation, trackers correct their own posterior state in reference to the fused state. Details of fusion in tracking are discussed in Sec. 2.7.

2.7 Fusion in tracking

Data fusion has been widely used for enabling robust visual tracking [132, 67]. The fusion principle can be directly related to the design of self-correcting framework. The score (weight) assessment to the units to fuse is similar to online performance measure. The fused output that is used as an input for next frame in Bayesian trackers (Sec. 3.2) is similar to applying the correction step for the less weighted unit in the fusion framework. Fusion can be implemented at different levels for tracking: *Component-Level Fusion* (CLF) [108, 65] on motion and appearance models using a single tracker, and *Tracker-Level Fusion* (TLF) [52] on trackers in a multi-tracker scenario. In this section, we review the related work on fusion for visual tracking.

Figure 2.5 shows block diagram for the two levels of fusion (CLF and TLF), and Table 2.2 summarises visual trackers using different fusion techniques.

2.7.1 Selection of units to fuse

The selection of units to be considered for fusion is mainly based on complementary performance properties. The complementary properties allow the units to overcome any unforeseen challenge mutually or independently (in piecewise selection manner). In addition to this, complementary properties of the units help to avoid the use of redundant tracking hypothesis [65]. The complementary performance properties are observed from robustness of the units to tracking challenges [32, 112, 51]. A comparison based on the simulated real-world challenges allows one to observe the complementary performance of trackers and tracker components for the fusion [110, 96, 73]. Holistic and local appearance representations have complementary performances to occlusions, and illumination and pose changes [113]. Zero-order and constant velocity motion models have complementary performances to occlusions and sudden pose changes of the target [45]. In TLF variations in searching and matching strategies can be considered in addition to the complementary performance properties of motion and appearance models for the trackers. In order to account for the long-term occlusions, a detector with a sliding-window search strategy can be assumed to have a complementary characteristic to a search strategy in a tracker for target reacquisition [48].

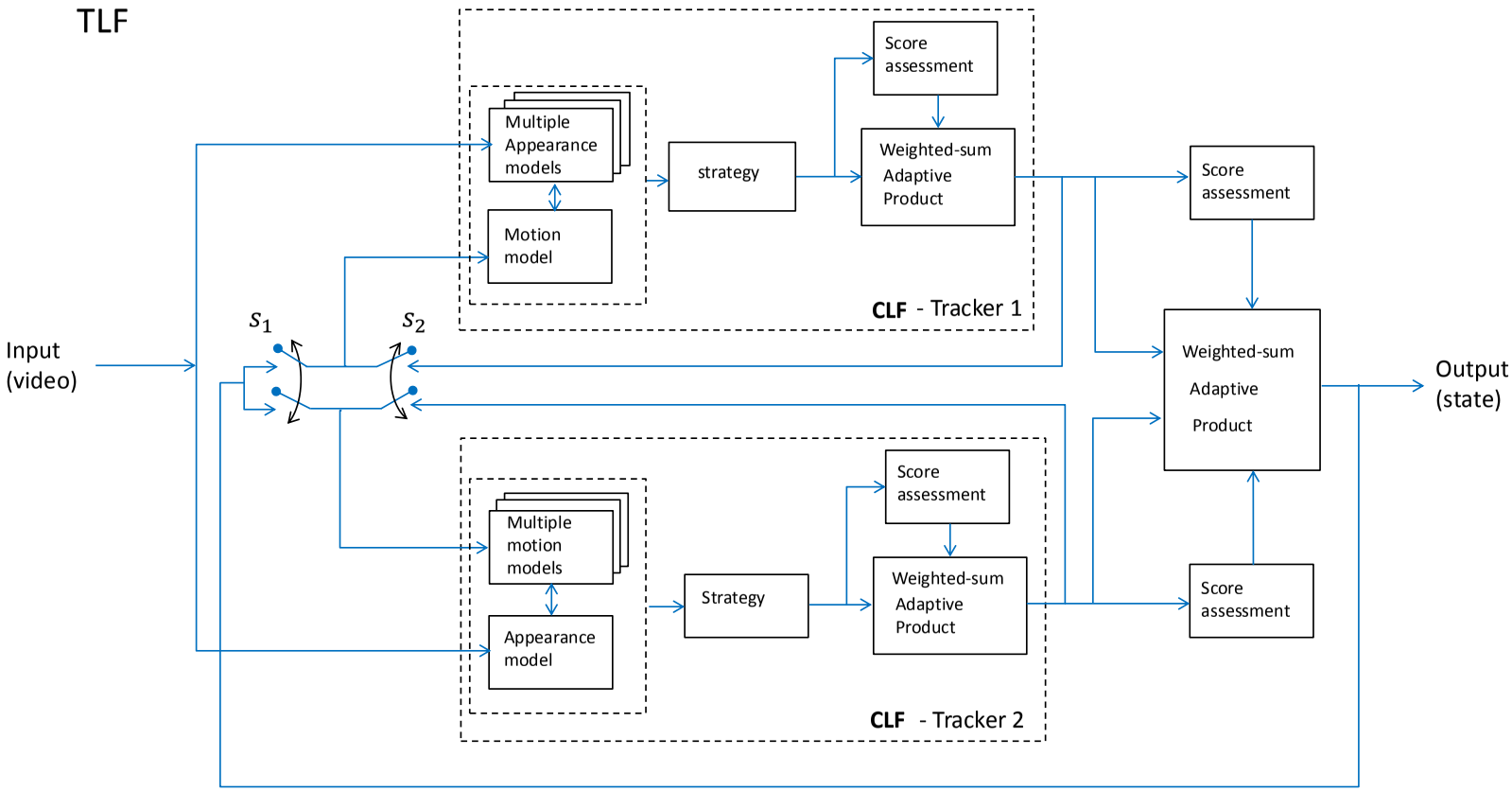


Figure 2.5: Block diagram of fusion in tracking. The diagram shows the use of multiple motion and appearance models in Component-Level Fusion (CLF) and the use of multiple trackers in Tracker-Level Fusion (TLF). In TLF the individual trackers might work collaboratively (switch $S_1 = ON$) or individually (switch: $S_2 = ON$).

Table 2.2: Visual trackers that use fusion. Key- CLF: component-level fusion, MF: multiple features (appearance models), MM: multiple motion models, TLF: tracker-level fusion, PF: particle filter, KF: Kalman filter, IVT: incremental visual tracking, ST: Superpixel tracker, SVM: Support Vector Machine, HOG: Histograms of Oriented Gradients, IMM: interactive multiple model.

Fusion level		**Ref.**	**Tracker(s)**	**Observation models/ Features**	**Motion models**	**Fusion technique**
CLF	MF	[65]	PF	part-wise colour and edge histograms	constant velocity	product rule
		[113]	PF	holistic and local sparse features from gray-image intensity	Gaussian random walk	product rule on the confidences
		[127]	PF	subspaces features and HOG features	adaptive Gaussian random walk	product rule on the likelihoods with co-training model update
		[12]	KF	holistic features from colour, motion and shape, and local colour histograms	constant velocity	weighted-sum rule
		[33]	PF	HSV colour and edge features	constant velocity	adaptive weighted-sum and product rules
	MM	[45]	KF	HOG features	zero-order and constant velocity	IMM framework - based on error covariance from Kalman filters
		[108]	PF	colour histogram	constant velocity and turn motion models	IMM framework - averaging
		[128]	PF	edges	constant velocity, acceleration and coordinated-turn model	likelihood-based weighted sum
TLF		[51]	multiple PF	hue, saturation, intensity and edge templates	Gaussian random walk with smooth and abrupt changes	IMM based on the likelihood value
		[52]	multiple PF	hue, saturation, intensity and edge templates	Gaussian random walk with K-Harmonic mean clustering	IMM-sampling of trackers and interaction by their likelihood
		[112]	IVT[86], ST	intensity and superpixel confidence map	Gaussian random walk	coincidence in state estimation between trackers and re-initialisation
		[102]	two SVM	colour histogram and HOG	full search in the whole image	classifiers error with sample exchange between trackers for co-training
		[32]	13 trackers	variable features	variable motion models	no interaction between trackers, optimisation-based combination
		[J2]	PF_1[76], PF_2[107]	holistic colour histogram, intensity based sparse features	constant velocity and Gaussian random walk	based on performance measure of trackers with prior correction

2.7.2 Fusion methods

Fusion of units is done based on different techniques such as product rule [113, 54], weighted-sum rule [12] and the Interactive Multiple Model (IMM) framework [108, 66]. The weighted-sum rule is a voting scheme where independent decisions from each model are combined in the form of mixtures as

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \sum_{i=1}^{M} \eta^i p(\mathbf{x}_k^i|\mathbf{Z}_k^i), \tag{2.1}$$

where $p(\mathbf{x}_k^i|\mathbf{Z}_k^i)$ represents the probability density estimate for target state from i^{th} unit and η^i is the weight assigned, while M is the number of the units to fuse. In $p(\mathbf{x}_k^i|\mathbf{Z}_k^i)$, variables $\mathbf{x}_k$ and $\mathbf{Z}_k$ represent target state and measurements, respectively (Sec. 3.2). The probability distributions from different fused components are usually integrated into a single distribution in the form of product rule as

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \prod_{i=1}^{M} p(\mathbf{x}_k^i|\mathbf{Z}_k^i). \tag{2.2}$$

The weighted-sum rule maximises the results of individual models and is robust to fused units that contain noise (such as measurements from an occlusion), while the product rule considers the integration of models and is effective for units with independent modalities [132]. An adaptive combination of weighted-sum rule and product rule can also be used for an effective fusion [33, 117]. IMM allows fusion of different models by switching between model associations [52, 112]. The model associations can be obtained by weighted-sum rule or product rule of the different fused units.

Co-training has also been proposed for fusion of learning-based and classifier-based trackers. In co-training, different classifiers cooperate by providing negative samples for their individual training [102] or use individual estimates for incrementally updating their target model in a criss-cross fashion [127]. The fused output from the co-trained units is estimated using the weighted-sum or product rules. Co-training is appropriate for features obtained from independent modalities such as voice and appearance [48].

2.7.3 Component-level fusion

A tracker using a single motion model and measurement model is more vulnerable to tracking challenges and encounters failure. In order to cope with the challenges multiple appearance mod-

els or multiple motion models can be used in CLF. Multiple appearance models can be used to handle changes in target appearance and measurement uncertainty [65, 113, 12]. The appearance model can also be obtained as a fusion of multiple visual cues [65, 33]. The visual cues represent appearance attributes of the target. Features containing colour and edge information, which can be either holistic or local representations for the target, are used to model the appearance in the form of histograms or templates. Most of the proposed methods exploit the strengths of both holistic and local features [113, 12]. Holistic appearance models can be robust to out-of-plane rotations and deformations, while local appearance models can be robust to partial occlusions, scaling and in-plane rotations [110, 112]. CLF using multiple appearance models includes part-wise colour histograms together with holistic-edge histograms [65], sparse features in the form of local and holistic templates (including background) [113], and local-colour histograms with motion, shape and colour features of the whole target [12].

Multiple motion models are utilised in CLF to cope with uncertainty and variable motion of the target [108]. Combinations of constant velocity and acceleration models, Gaussian random walk with various covariances and coordinated turn models are used to implement motion variations [45, 128]. In Multiple Model (MM) particle filter, a bank of parallel Extended Kalman filters is used to construct the proposal distribution for accurate tracking [108]. The Extended Kalman filters have different motion models: constant velocity and acceleration, and clockwise and anticlockwise coordinated turn motion models. For handling sudden stops and direction changes, a zero-order motion model and a constant velocity motion model can be integrated for pedestrian tracking [45].

2.7.4 Tracker-level fusion

TLF is carried out using output from multiple trackers. Due to tracking challenges, it is possible that both of the appearance and motion models of the target changes simultaneously. Since CLF uses either multiple motion models with single appearance model or multiple appearance models with single motion model, it might not be possible to address such changes when using CLF. Fusion of multiple trackers allows us to handle the simultaneous change in motion and appearance of the target by incorporating multiple motion and appearance models in the trackers [132]. TLF can be considered as a generalisation of CLF, regardless of computational complexity. Within TLF framework, individual trackers can operate either in an independent [32], or in an interactive (cooperative) manner [112, 52, 54]. The two modes of operations in TLF are shown by the

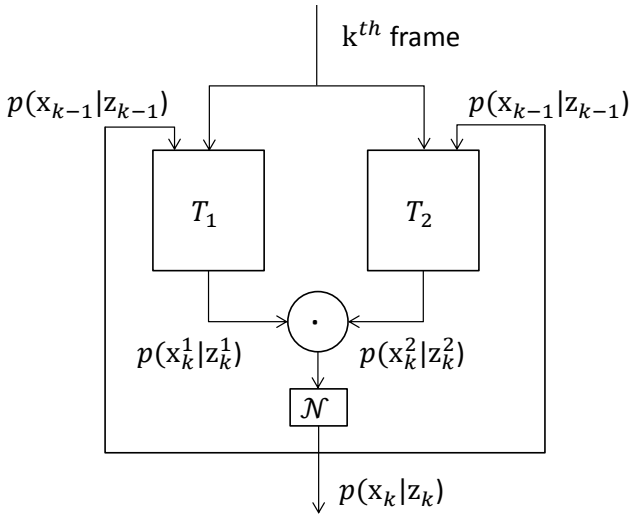


Figure 2.6: Closed box fusion of trackers T_1 and T_2 (this figure is an adaptation of [54]). The superscripts in variables identify trackers. The block $\mathcal{N}$ is used for normalising the posterior distribution.

switches in Fig. 2.5. Fusion of independent working trackers aims only for an optimal combination of tracks for boosting the overall performance compared to individual trackers. The fused result is not used as a feedback for improving the performance of individual trackers and thus the fusion operation can be done offline. Individual trackers in the fusion are considered as a black box without requiring details of trackers [32]. Optimal combination from the individual tracks is obtained by considering only the correlation among tracks.

Cooperating trackers in TLF aims to overcome a tracking failure mutually. A generalised fusion framework is proposed that treats trackers as a 'closed box' and combines the outputs from each tracker [54] (Fig. 2.6). The combinations of multiple motion models with multiple appearance models have been used in order to create multiple trackers that work in collaboration as a single compound tracker [53, 52]. Appropriate trackers, selected based on likelihood score, are sampled and communicate with other trackers for improving individual performance together with the overall tracking performance. A region tracker [86] (robust to target scale changes and in-plane rotations) is fused with an object tracker (superpixel tracker robust to out-of-plane rotations and deformations) in a complementary tracking framework [112]. The re-initialisation of a tracker from the estimated state of the other one is made by observing the overlapping area between the trackers' estimate bounding-boxs. TLF can also be in the form of cascaded fusion [111, 63], where the result of a tracker is used as an input for the other one with the aim of obtaining more accurate tracks. In the case of different state representation techniques by the

trackers, state conversion between trackers is a necessary step during fusion [54].

2.8 Discussion

In tracking, changes in properties of the target together with changes in the tracking environment can cause trackers to failures. In order to deal with the failures and obtain robust tracking results, an explicit modelling of tracking failures or handling of tracking challenges should be adopted for trackers. Failures and performance degradation in trackers can be detected using online evaluation techniques and recovering from failure can be done in the form of correction. The evaluation and correction techniques are directly related to the design of a tracker (Table 2.1), thus we have summarised state-of-the-art tracking strategies and tracker components as part of designing self-correcting tracking.

Performance measures on trackers have been discussed for failure detection and quality assessment in order to devise the self-correcting tracking. Changes in trackers parameters and characteristics of tracks are compared with known properties for the evaluation. Robust and generic evaluation methods for all tracking scenarios are very challenging, but it is an important problem that needs to be addressed [25, 48]. Different evaluation criteria and characteristics have been considered related to the wide variation of tracking strategies and state representations. Track characteristics such as trajectory smoothness and shape, colour differences with the background and errors from assumed tracking models have been considered for the evaluation. In general, the accuracy of a particular performance measure might vary for different trackers due to variations in state representation and tracking formulations used in the trackers. In order to improve the performance of a particular tracker, a suitable evaluation method can be adopted for better quality assessment and failure identification [103, 120, 48].

Trackers using correction in the form of model update and re-initialisation have also been discussed. The challenging issue behind correction is looking for information that helps to make appropriate changes to the tracker so that the tracker resumes its normal operation [48, 87]. Moreover, identifying the component of the tracker undergoing changes is a concern during correction. The use of recent tracker output as source of information without proving its correctness is susceptible to drifting problem [88]. In order to alleviate the drifting, the adaptive modelling approach needs to be assisted by an appropriate online performance measure. Information for correction can stem from different sources such as context information and other trackers running

in parallel (fusion) [124, 53]. As potential solution towards obtaining correction information, we have summarised related works on fusion for tracking (Table 2.2). The existence of wide range of tracker with different strategies and ability to cope with different challenges encourage the use of TLF [65]. For effective implementation of characteristics to avoid failures and to recover from failures, we have mentioned the requirement of complementary properties between trackers in the fusion framework. Appropriate online performance measures need to be addressed for the fusion units for better quality assessment. Moreover, the collaboration method plays an important role to improve the performance of individual tracker and at the same time to boost the overall fusion framework.

This thesis aims to enable trackers a self-correcting capability by embodying online performance measure and correction explicitly, TEC framework (Fig. 1.2). Specifically, for an extended object tracker with local model points we use covariance property and source of failure models to obtain quality of local tracker, and utilized correlation information between local trackers to correct failed tracks (Chapter 4). Accurate correction information is acquired by using adaptive online correlation modelling. In tracker-level fusion, we include appropriate online performance measure and collaboration strategy to obtain self-correcting capability among individual trackers (Chapter 5). We also utilized the online performance measure to guide and minimize drifting in model update for the individual trackers.

Chapter 3

Dynamic Bayesian Network framework for self-correcting tracking

3.1 Introduction

Trackers using a fixed motion model and appearance model are vulnerable to tracking challenges, and may encounter failures. In order to avoid and/or recover from these failures, trackers can be assisted by evaluation and correction in self-correcting tracking (Fig. 1.2). A Dynamic Bayesian Network (DBN) is an appropriate mathematical tool for representing and solving the inference problem of the state estimation in self-correcting tracking. A DBN model of a baseline tracker can be extended to contain the evaluation and correction units in order to obtain a DBN model of self-correcting tracking. In this chapter, we provide a generic formulation and representation of self-correcting tracking using the DBN model. In the DBN model of self-correcting tracking, we incorporate additional hidden layers to the DBN model of a baseline tracker. The additional hidden layers are represented by discrete variables that allow the self-correcting tracker to have evaluation and correction operations.

In this chapter, the baseline method of tracking in a Bayesian framework is presented in Sec. 3.2. The proposed DBN model for a generic self-correcting tracker with associated filtering equations is discussed in Sec. 3.3. Approximate inference for the DBN model based on the TEC framework is described in Sec. 3.4. Correction of a tracker is explained based on the filtering equations from the DBN model in Sec. 3.5. A Coupled DBN model for interacting units in self-correcting tracking is presented in Sec. 3.6. Finally, Sec. 3.7 summarises the chapter.

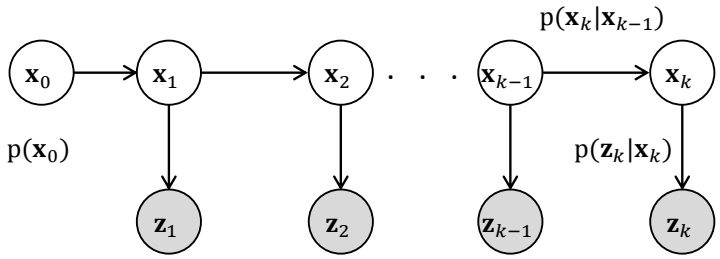


Figure 3.1: Graphical model representation of a generalised baseline Bayesian tracker [71]. Nodes represent a random variable and arrows show the dependence between variables. Observable variables are indicated by shading the nodes.

3.2 Bayesian tracking

Let $\mathbf{I} = \{I_k\}_{k=1}^K$ be an image sequence of K number of frames. Tracking involves estimating of the states $\mathbf{X} = \{\mathbf{x}_k\}_{k=1}^K$ of the target over time from a set of available observations $\mathbf{Z} = \{\mathbf{z}_k\}_{k=1}^K$ obtained from $\mathbf{I}$. A Bayesian network, which is a special class of Probabilistic Graphic Models (PGMs), allows us to represent conditional dependence between variables and to estimate hidden variables given other observed variables. A Dynamic Bayesian Network (DBN) is a type of Bayesian Network for handling changes in the value of a variable over time [71]. A DBN model of a generalised baseline Bayesian tracker is shown in Fig. 3.1. The model is used for analysing the evolution and the conditional dependence within and across the time slot for the target state [7, 84].

Tracking in Bayesian formulation is expressed as sequentially estimating $\mathbf{x}_k$, from the set of observations $\mathbf{Z}_k = [\mathbf{z}_k, \mathbf{z}_{k-1} \cdots \mathbf{z}_0]$. The state, $\mathbf{x}_k$, is estimated in the form of a posterior distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$ using two important steps: a prediction step followed by an update step. The prediction step is

$$p(\mathbf{x}_k|\mathbf{Z}_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1}) d\mathbf{x}_{k-1}, \tag{3.1}$$

where $p(\mathbf{x}_k|\mathbf{Z}_{k-1})$ is the probability distribution of the current state estimated from all previous observations and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the probability distribution for state transition between consecutive frames. $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is estimated from the assumed motion model of the target. The update step estimates

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{Z}_{k-1})}{p(\mathbf{z}_k|\mathbf{Z}_{k-1})}, \tag{3.2}$$

where $p(\mathbf{z}_k|\mathbf{x}_k)$ is the likelihood function obtained using the appearance model and $p(\mathbf{z}_k|\mathbf{Z}_{k-1})$ is a normalising function. The Bayesian filtering assumes a prior distribution of $p(\mathbf{x}_0)$ and Markovian assumptions for the sequential state estimation. The prior distribution allows us to define the identity of the target and is used to initialise tracking. The Markovian assumptions are: a first order process for the hidden states, *i.e.* $p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{Z}_{k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$, and a conditional independence between the current observation with the rest of the previous observations given the current state $\mathbf{x}_k$, *i.e.* $p(\mathbf{z}_k|\mathbf{x}_k,\mathbf{Z}_{k-1}) = p(\mathbf{z}_k|\mathbf{x}_k)$.

The posterior distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$ can have a linear Gaussian form as in the case of Kalman filters, or any arbitrary non-linear distribution as in the case of particle filters. From the distribution $p(\mathbf{x}_k|\mathbf{Z}_k)$, the best hidden state $\hat{\mathbf{x}}_k$ can be approximated using either Minimum Mean Square (MMS) or Maximum a Posteriori (MAP) formulations. In MMS, $\hat{\mathbf{x}}_k$ is the mean of distribution calculated as

$$\hat{\mathbf{x}}_k^{MMS} \approx E[p(\mathbf{x}_k|\mathbf{Z}_k)] = \int \mathbf{x}_k p(\mathbf{x}_k|\mathbf{Z}_k) d\mathbf{x}_k, \tag{3.3}$$

while in the case of MAP, $\hat{\mathbf{x}}_k$ is the best candidate selected as

$$\hat{\mathbf{x}}_k^{MAP} \approx arg \max_{\mathbf{x}_k} p(\mathbf{x}_k|\mathbf{Z}_k). \tag{3.4}$$

3.3 Dynamic Bayesian Network model of self-correcting tracking

3.3.1 DBN model

A self-correcting tracker should contain the evaluation and correction to improve a baseline tracker (Fig. 1.2). A DBN model for self-correcting tracker requires additional hidden layers for representing the evaluation and correction compared to the DBN model of the baseline tracker. The additional hidden layers result in a deeper hierarchical DBN for a self-correcting tracker as shown in Fig. 3.2. The DBN has a similar structure to that of Markov switching models such as Switching Linear Dynamical System (SLDS) [118, 78], or Change Point Models (CPM) [28, 7]. In SLDS and CPM the additional nodes are used for selection of models. In the case of the DBN model for self-correcting tracking, the additional nodes explicitly model evaluation and correction operations that also include model selection as done by SLDS and CPM.

The top hidden layer of the DBN model for the self-correcting tracker (Fig. 3.2) contains a discrete *performance evaluation variable*, $\mathbf{p}_k$, for providing information about the performance

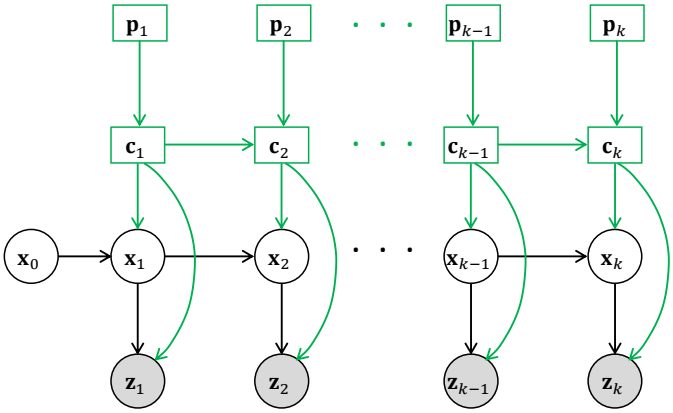


Figure 3.2: DBN graph of self-correcting tracker. The discrete evaluation variable, $\mathbf{p}_k$, and correction variable, $\mathbf{c}_k$, allow the tracker to tune the parameters in the models or to select different models, while $\mathbf{x}_k$ and $\mathbf{z}_k$ are the target state and observation variables, respectively. Circle and square nodes represent continuous and discrete variables, respectively.

of a tracker. The hidden layer just below $\mathbf{p}_k$ contains a discrete *correction variable*, $\mathbf{c}_k$, that allows the tracker to have a decision for correction. The discrete variables take values from the sets

$$\begin{aligned} \mathbf{p}_k &= \left\{ \mathrm{p} \in \mathbb{N}^0 : 0 \leq \mathrm{p} \leq (N_\mathbf{p} - 1) \right\}, \\ \mathbf{c}_k &= \left\{ c \in \mathbb{N}^0 : 0 \leq c \leq (N_\mathbf{c} - 1) \right\}, \end{aligned} \tag{3.5}$$

where $N_\mathbf{p}$ and $N_\mathbf{c}$ are the number of discrete values for $\mathbf{p}_k$ and $\mathbf{c}_k$, respectively. The $N_\mathbf{p}$ discrete labels for $\mathbf{p}_k$ represent the results of distinctive classes from the performance measure on the tracker. For instance good and poorly performing classes can be considered as two labels of $\mathbf{p}_k$. The $N_\mathbf{c}$ discrete labels for $\mathbf{c}_k$ identify the type of corrections done on a tracker such as motion or observation model selection, and re-initialisation.

The DBN model has a joint probability distribution of

$$p(\mathbf{x}_{0:K}, \mathbf{c}_{0:K}, \mathbf{p}_{0:K}, \mathbf{z}_{0:K}) = \prod_{k=0}^{K} p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{c}_k) p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{c}_k) p(\mathbf{c}_k|\mathbf{c}_{k-1}, \mathbf{p}_k) p(\mathbf{p}_k). \tag{3.6}$$

In the joint probability distribution $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{c}_k)$ and $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{c}_k)$ are the measurement and motion models, respectively, which are dependent on the correction variable $\mathbf{c}_k$. The correction variable has also prediction model of $p(\mathbf{c}_k|\mathbf{c}_{k-1}, \mathbf{p}_k)$.

3.3.2 Inference: filtering equations

For the DBN model shown in Fig. 3.2, the filtering task is estimation of the probability distributions $p(\mathbf{x}_k|\mathbf{Z}_k, \mathbf{c}_k)$, $p(\mathbf{c}_k|\mathbf{Z}_k, \mathbf{p}_k)$ and $p(\mathbf{p}_k|\mathbf{Z}_k)$. Due to the discrete characteristics of $\mathbf{p}_k$ and $\mathbf{c}_k$, the

distributions $p(\mathbf{x}_k|\mathbf{Z}_k,\mathbf{c}_k)$, $p(\mathbf{c}_k|\mathbf{Z}_k,\mathbf{p}_k)$ and $p(\mathbf{p}_k|\mathbf{Z}_k)$ need to be evaluated for each discrete value as $p(\mathbf{x}_k|\mathbf{Z}_k,\mathbf{c}_k=c)$, $p(\mathbf{c}_k=c|\mathbf{Z}_k,\mathbf{p}_k=\mathrm{p})$ and $p(\mathbf{p}_k=\mathrm{p}|\mathbf{Z}_k)$, respectively [30].

The distribution $p(\mathbf{x}_k|\mathbf{Z}_k,\mathbf{c}_k=c)$ is estimated in the recursion from the probability density function $p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1},\mathbf{c}_{k-1}=\hat{c})$, where $\hat{c}$ is the correction variable value at $k-1$. In principle, the recursion is similar to the Bayesian formulations of the baseline tracker (Eq. 3.1 and 3.2). The function $p(\mathbf{x}_k|\mathbf{Z}_k,\mathbf{c}_k=c)$ is estimated using a prediction step

$$p(\mathbf{x}_k|\mathbf{Z}_{k-1},\mathbf{c}_k=c)=\int p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{c}_k=c)p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1},\mathbf{c}_k=c)d\mathbf{x}_{k-1}, \tag{3.7}$$

and then an update step:

$$p(\mathbf{x}_k|\mathbf{Z}_k,\mathbf{c}_k=c)\propto p(\mathbf{z}_k|\mathbf{x}_k,\mathbf{c}_k=c)p(\mathbf{x}_k,|\mathbf{Z}_{k-1},\mathbf{c}_k=c). \tag{3.8}$$

In the filtering equations the densities $p(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{c}_k=c)$ and $p(\mathbf{z}_k|\mathbf{x}_k,\mathbf{c}_k=c)$ are in their appropriate form as described in Eq. 3.6. However, in Eq. 3.7 the probability density function $p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1},\mathbf{c}_k=c)$ shows non-normality in its form which arises from the possible changes of values between $\mathbf{c}_{k-1}$ and $\mathbf{c}_k$. The density $p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1},\mathbf{c}_k=c)$ is estimated by using a finite mixture of the prior $p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1},\mathbf{c}_{k-1}=\hat{c})$ as

$$p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1},\mathbf{c}_k=c)=\sum_{\hat{c}=0}^{N_\mathbf{c}-1}p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1},\mathbf{c}_{k-1}=\hat{c})w_{\hat{c}c}, \tag{3.9}$$

where $w_{\hat{c}c}$ is the weight for the prediction of $\mathbf{c}_k=c$ from the previous value $\mathbf{c}_{k-1}=\hat{c}$. The weight is estimated proportional to the distribution $p(\mathbf{c}_{k-1}=\hat{c}|\mathbf{Z}_{k-1},\mathbf{p}_{k-1}=\hat{\mathrm{p}})$. Similarly to the filtering equation for $p(\mathbf{x}_k|\mathbf{Z}_k,\mathbf{c}_k=c)$, the probability density $p(\mathbf{c}_k=c|\mathbf{Z}_k,\mathbf{p}_k=\mathrm{p})$ can be estimated using prediction and update steps [30].

Exact inference techniques for the DBN model in Fig. 3.2 are complex and computationally expensive due to the fact that the distributions $p(\mathbf{x}_k|\mathbf{Z}_k,\mathbf{c}_k)$, $p(\mathbf{c}_k|\mathbf{Z}_k,\mathbf{p}_k)$ and $p(\mathbf{p}_k|\mathbf{Z}_k)$ require recursions. The recursions contain integrations over $\mathbf{x}_k$ together with summation over the values of discrete variables($\mathbf{p}_k$ and $\mathbf{c}_k$). For cases such as Gaussian probability density functions for models in the tracker, the complexity of the filtering equation grows exponentially due to the additional discrete variable values [7, 30]. For reducing the complexity an approximation is considered in the filtering equations. The most widely used approximation is based on the use of only the best value $\mathbf{c}_k$ in the filtering equations [49, 80]. The best value of $\mathbf{c}_k$ is selected from

the maximum likelihood value of $p(\mathbf{z}_k|\mathbf{x}_k,\mathbf{c}_k)$. We use our proposed TEC framework to obtain an approximate filtering equations as explained in Sec. 3.4.

3.4 Approximate inference using the TEC framework

The TEC framework implements a self-correcting tracker with an explicit performance measure to estimate the evaluation variable that, in turn, is used to determine the correction variable. To overcome the complexity of exact-variable inference in the DBN model, an approximation is needed. The step-wise operations of tracking, evaluation and correction in the TEC framework are implicit assumptions that allow us to obtain the approximate filtering equations. At each k, the evaluation and correction variables are estimated after obtaining the result of the hidden state $\mathbf{x}_t$. The step-wise variable inference corresponds to freezing a subset of the network nodes, i.e. $\mathbf{p}_k$ and $\mathbf{c}_k$ with their most probable values in order to estimate $\mathbf{x}_k$ [70, 75]. Estimation of $\mathbf{x}_k$ involves the use of the prediction of $\mathbf{c}_k$ from time $k-1$ as the most probable value.

3.4.1 Estimation of the evaluation variable

From the TEC framework assumption, the distribution for evaluation variable $p(\mathbf{p}_k = \mathrm{p}|\mathbf{Z}_k)$ is obtained after the estimation of the target state at time k, i.e. when $\mathbf{x}_k$ is available. The probability distribution $p(\mathbf{p}_k = \mathrm{p}|\mathbf{Z}_k)$ is estimated as

$$p(\mathbf{p}_k = \mathrm{p}|\mathbf{Z}_k) = f_{\mathrm{p}}(\mathbf{x}_k, I_p), \tag{3.10}$$

where $f_{\mathrm{p}}(\cdot)$ is a performance measure function that assigns the probability density for $\mathbf{p}_k$. When $\mathbf{c}_k$ is not available, the distribution of $\mathbf{p}_k$ is dependent on $\mathbf{x}_k$ (see the DBN model, Fig. 3.2). I_p is any information other than the current estimated state of the tracker, such as pre-defined threshold values and reference data, which are used by $f_{\mathrm{p}}(\cdot)$. Determining $f_{\mathrm{p}}(\cdot)$ together with I_p for the tracker plays an important role as part of implementing the self-correcting tracking. The performance measure $f_{\mathrm{p}}(\cdot)$ is a non-linear operation and returns one label of the track quality at each frame k.

3.4.2 Estimation of the correction variable

The value of $\mathbf{c}_k$ is estimated from the value of $\mathbf{p}_k$ (Fig. 3.2). The distribution $p(\mathbf{c}_k = c|\mathbf{Z}_k, \mathbf{p}_k = \mathrm{p})$ is estimated according to the formulation

$$p(\mathbf{c}_k = c|\mathbf{Z}_k, \mathbf{p}_k = \mathrm{p}) = f_c(\mathbf{c}_k, \mathbf{p}_k), \tag{3.11}$$

where $f_c(\cdot)$ is a function that maps the probability density values of $\mathbf{p}_k$ to $\mathbf{c}_k$. The characteristic of $f_c(\cdot)$ controls the growth in hypotheses of the filtering equation due to the different values of $\mathbf{c}_k$. For instance, $f_c(\cdot)$ with a Kronecker delta function characteristic can be used to select one of the values of $\mathbf{c}_k$ at each time k. Selection of a single value for $\mathbf{c}_k$ is equivalent to maximum likelihood or maximum a posteriori formulation of multiple models [80].

3.4.3 Approximate filtering equations

At each k, the state $\mathbf{x}_k$ is estimated before the estimation of $\mathbf{p}_k$ and $\mathbf{c}_k$. However, the equations in the estimation of $\mathbf{x}_k$ are dependent on the discrete variable $\mathbf{c}_k$ (Eq. 3.7 and 3.8). As an approximation, in order to estimate $\mathbf{x}_k$, the prediction values of the discrete variables from time $k-1$ can be considered as the most probable value at time k.

Let the prediction of the correction variable be represented as $\mathbf{c}_{k|k-1}$. In the TEC framework based estimation, the posterior probability density function $p(\mathbf{x}_k, |\mathbf{Z}_k, \mathbf{c}_{k|k-1})$ is calculated in a similar way to the Bayesian formulations given in Eq. 3.7 and 3.8. The prediction step estimates $p(\mathbf{x}_k|\mathbf{Z}_{k-1}, \mathbf{c}_{k|k-1})$ as

$$p(\mathbf{x}_k|\mathbf{Z}_{k-1}, \mathbf{c}_{k|k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{c}_{k|k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1}, \mathbf{c}_{k|k-1}) d\mathbf{x}_{k-1}. \tag{3.12}$$

The update of a tracker involves estimating the final state $p(\mathbf{x}_k|\mathbf{Z}_{k-1}, \mathbf{c}_{k|k-1})$ as

$$p(\mathbf{x}_k|\mathbf{Z}_k, \mathbf{c}_{k|k-1}) \propto p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{c}_{k|k-1}) p(\mathbf{x}_k, |\mathbf{Z}_{k|k-1}, \mathbf{c}_{k|k-1}). \tag{3.13}$$

3.5 Tracking correction

Correcting a tracker involves the use of different tracking models that are suitable for the current observation and target's characteristics. Additionally correction can be done in the form of changing of prior information of the target (i.e. a form of re-initialisation). For a generic inter-

pretation, let us consider that the discrete value $c = 0$ represents the status of tracking without correction (i.e. the baseline mode of operation), while $c \neq 0$ represents the mode of operation with corrections and changes made on the baseline method. The correction step aims to modify the tracker T and to improve the accuracy of the estimated states $\mathbf{x}_k$ as

$$\overline{T}, \overline{\mathbf{x}}_k = \begin{cases} T, \mathbf{x}_k, & \text{if } \mathbf{c}_k = 0, \\ \Theta(\mathbf{x}_k, T, \mathbf{c}_k, I_c) & \text{otherwise} \end{cases} \tag{3.14}$$

where $\Theta(\cdot)$ is the transformation made to obtain the corrected tracker $\overline{T}$ and the improved states $\overline{\mathbf{x}}_k$. Similarly to I_p (Eq. 3.10), I_c represents valid information, such as the true state of the target from a detector [48] to assist the correction technique.

$\overline{T}$ is obtained by changing the parameters and models for the tracker. The correction can be associated to the Bayesian filtering equations discussed in Eq. 3.12 and 3.13. In the prediction equation the motion model probability density is conditioned on the variable $\mathbf{c}_k$, and can be expressed as

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{c}_{k|k-1} = c) = \begin{cases} p(\mathbf{x}_k|\mathbf{x}_{k-1}) & \text{if } c = 0, \\ p(\overline{\mathbf{x}}_k|\mathbf{x}_{k-1}) = f_{\Theta_m}(\mathbf{x}_k, I_c) & \text{if } c \neq 0, \end{cases} \tag{3.15}$$

where $p(\overline{\mathbf{x}}_k|\mathbf{x}_{k-1})$ is the corrected motion model and f_{Θ_m} is a function to estimate the corrected motion probability density. f_{Θ_m} typically implements motion models different from the baseline tracker, or adopts the motion model of the baseline tracker based on the different values of $\mathbf{c}_k$.

Correction on the observation model $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{c}_{k|k-1})$ is performed at the update step according to

$$p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{c}_{k|k-1} = c) = \begin{cases} p(\mathbf{z}_k|\mathbf{x}_k) & \text{if } c = 0, \\ p(\mathbf{z}_k|\overline{\mathbf{x}}_k) = f_{\Theta_o}(\mathbf{x}_k, \mathbf{Z}_k, I_c) & \text{if } c \neq 0, \end{cases} \tag{3.16}$$

where $p(\mathbf{z}_k|\overline{\mathbf{x}}_k)$ is the corrected observation model and f_{Θ_o} represents a function to estimate the corrected observation model. Similarly to f_{Θ_m}, f_{Θ_o} can be implemented by using different observation models from the baseline tracker or it can be an adaptive version of the observation model in the baseline tracker.

The posterior estimate at time k is used as prior information at time $k+1$. The prior density provides vital information for the execution of the tracker. In the case of a poor estimation of the

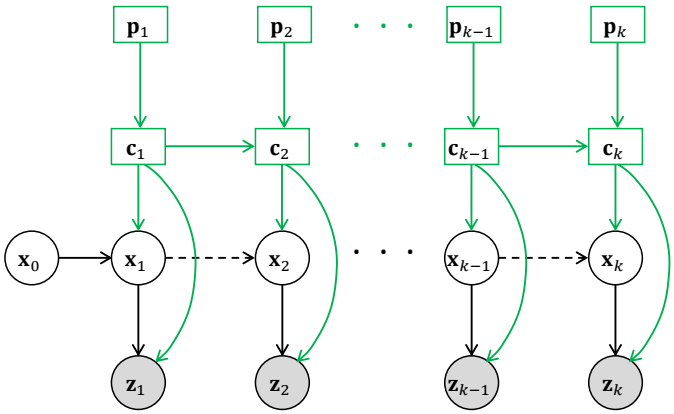


Figure 3.3: Reset model for re-initialisation of a tracker (prior correction). The reset model is a particular case of the general DBN shown in Fig. 3.2. In the model, for some values of $(\mathbf{c}_k, \mathbf{p}_k)$ the Markov chain between $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ can be discontinuous (indicated by weak lines between the states).

state, the posterior knowledge is incorrect, and this may lead a tracker failure. In the update step (Eq. 3.13), the tracker estimates the posterior density $p(\mathbf{x}_k|\mathbf{Z}_k, \mathbf{c}_{k|k-1})$. At each time k, the posterior $p(\mathbf{x}_k|\mathbf{Z}_k, \mathbf{c}_{k|k-1})$ needs to be corrected to $p(\mathbf{x}_k|\mathbf{Z}_k, \mathbf{c}_k)$ based on the current estimate values of $\mathbf{c}_k$. We refer to this correction schema as prior density correction, which can be expressed as

$$p(\mathbf{x}_k|\mathbf{Z}_k, \mathbf{c}_k = c) = \begin{cases} p(\mathbf{x}_k|\mathbf{Z}_k, \mathbf{c}_{k|k-1}) & \text{if } c = 0, \\ p(\overline{\mathbf{x}}_k|\mathbf{Z}_k, \mathbf{c}_{k|k-1}) = f_{\Theta_p}(\mathbf{x}_k, \mathbf{Z}_k, I_c) & \text{if } c \neq 0, \end{cases} \tag{3.17}$$

where $p(\overline{\mathbf{x}}_k|\mathbf{Z}_k, \mathbf{c}_{k|k-1})$ is the corrected prior and f_{Θ_p} is a function used to estimate the corrected prior density. A re-initialisation of a tracker is a prior correction, and it involves breaking the Markov chain between consecutive state estimates $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$, which results in a reset model as shown in Fig. 3.3 [9]. In the figure, the weak link between states $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ is to show the independence of the current state from the previous state estimate whenever $c \neq 0$.

3.6 Coupled DBN model of interacting units

Incorporating multiple trackers and tracking components as a single system helps to attain self-correcting tracking in the form of collaboration (Sec. 2.7). A coupled DBN (CDBN) allows us to represent the operation of two or more interacting units. The filtering equation can be obtained from the inference of variables in the CDBN. In the CDBN, the different units interact based on hidden variables from the individual units [71]. Figure 3.4 shows a CBDN model of two interacting trackers i and j in order to obtain a self-correcting tracker. Trackers i and j have

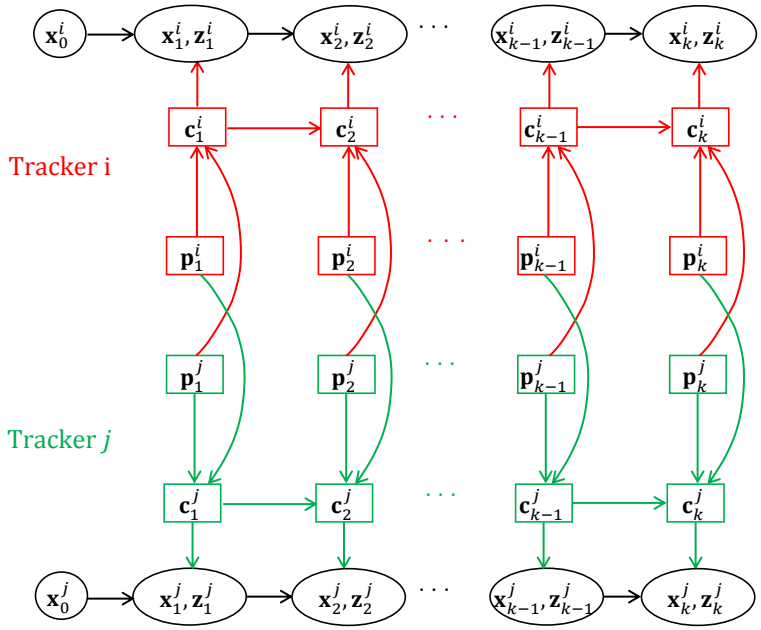


Figure 3.4: Coupled DBN model of interacting trackers i and j to enable self-correcting tracking.

their own observations $\mathbf{z}_k^i$ and $\mathbf{z}_k^j$, state estimations $\mathbf{x}_k^i$ and $\mathbf{x}_k^j$, evaluation variables $\mathbf{p}_k^i$ and $\mathbf{p}_k^j$, and correction variables $\mathbf{c}_k^i$ and $\mathbf{c}_k^j$, respectively. The individual trackers estimate the target state $\mathbf{x}_k^i$ and $\mathbf{x}_k^j$ and evaluation variables $\mathbf{p}_k^i$ and $\mathbf{p}_k^j$ based on their respective variables. However, the trackers interact based on their individual evaluation variables $\mathbf{p}_k^i$ and $\mathbf{p}_k^j$ in order to estimate the correction variable $\mathbf{c}_k^i$ and $\mathbf{c}_k^j$. The estimated correction variables determine modes of operation for each tracker. The mode of operation for trackers includes the weights assigned to each tracker for the fusion and the level of cooperation between trackers for avoiding failures to obtain robust tracks.

In order to obtain the inference of variables in CDBN, similar strategies can be followed as described in Sec. 3.3 and Sec. 3.4. For the units i and j, evaluation variables are determined independently as of Eq. 3.10

$$\begin{aligned} p(\mathbf{p}_k^i = \mathrm{p}|\mathbf{Z}_k) &= f_{\mathrm{p}}(\mathbf{x}_k^i, I_p), \\ p(\mathbf{p}_k^j = \mathrm{p}|\mathbf{Z}_k) &= f_{\mathrm{p}}(\mathbf{x}_k^j, I_p). \end{aligned} \tag{3.18}$$

However, the correction variables of each unit is estimated as

$$\begin{aligned} p(\mathbf{c}_k^i|\mathbf{Z}_k, \mathbf{p}_k^i, \mathbf{p}_k^j) &= f_c^i(\mathbf{c}_k^i, \mathbf{p}_k^i, \mathbf{p}_k^j), \\ p(\mathbf{c}_k^j|\mathbf{Z}_k, \mathbf{p}_k^i, \mathbf{p}_k^j) &= f_c^j(\mathbf{c}_k^j, \mathbf{p}_k^j, \mathbf{p}_k^i). \end{aligned} \tag{3.19}$$

In the CDBN, unlike Eq. 3.11, the estimation of the probability density function for $\mathbf{c}_k$ for each unit is a function of the $\mathbf{p}_k$ from the other unit. f_c does not fulfil commutative properties for $\mathbf{p}_k^i$ and $\mathbf{p}_k^j$, and the superscript of f_c identifies the estimation of $\mathbf{c}_k$ for the particular interacting unit.

3.7 Summary

This chapter presented a generic formulation and representation of self-correcting target tracking using a DBN. In the representation, we added nodes to the DBN model of a baseline tracker resulting in a deep hierarchical DBN model. The additional nodes (that we referred to as evaluation and correction variables) are discrete variables and control update, selection and fusion of models. For the DBN model, inference of variables is explained as filtering equations. Due to the additional nodes, exact inference among variables is computationally expensive and the need for an approximations has been discussed. We utilised our proposed TEC framework (Fig. 1.2) in order to obtain approximate inference between the variables in the DBN model. Different schemes of making correction in a tracker or tracker's model are explained based on the filtering equations of the proposed DBN model.

The chapter explained a generic means of obtaining self-correcting trackers using DBN. Table 3.1 summarises the main components in order to model self-correcting tracking using the TEC framework. The components of the self-correcting tracking can be designed for a particular baseline tracker. The design includes determining appropriate performance measure and correction functions for the baseline tracker. Chapters 4 and 5 present details of our proposed self-correcting trackers using a correlation-based correction for an extended object and a tracker-level fusion framework, respectively. For the proposed trackers, in Sec. 4.5 and Sec. 5.6 we discuss the characteristics of discrete variables following the DBN model developed in this chapter.

Table 3.1: Main components for modelling self-correcting tracking using TEC framework.

TEC framework components	**Symbol**	**Equation**	**Comment**
Baseline tracker	T		estimates the target state using Eq. 3.1 and 3.2
Evaluation variable estimator	$f_{\mathrm{p}}(\cdot)$	3.10	estimates evaluation variables that correspond to the labels from tracking performance
Extra information for evaluation	I_p	3.10	used by $f_{\mathrm{p}}(\cdot)$ for estimating evaluation variables
Correction variable estimator	$f_c(\cdot)$	3.11	estimates the type/label of correction on a tracker
	$f_c^i(\cdot), f_c^j(\cdot)$	3.19	superscript values identify interacting units i and j
Correction function	$\Theta(\cdot)$	3.14	changes the models used in T based on the estimated correction variable
Extra information for correction	I_c	3.14	used by $\Theta(\cdot)$ for correcting T

Chapter 4

Correlation-based self-correcting tracking

4.1 Introduction

The target state representation varies from one application domain to another. An extended object can be represented by a set of model points estimated from multiple independent measurements. Local appearance is modelled using pre-selected points on the object, which we refer to as *model points*, such as markers in a motion capture system [8], or features extracted from images using Scale-Invariant Feature Transform (SIFT) [100]. Tracking model points is achieved by estimating the state of each point individually, which generates a challenge for data association [106]. Moreover, performance degradation or failures in tracking can be generated by the challenges related to data association, missed and false detections, illumination changes and occlusions [110]. Therefore, a performance measure to detect tracking failures and a tracking correction step are desirable to obtain a robust tracking [15, 61, 16]. To this end, we use the TEC framework for the Bayesian filtering of model points [J1] (Fig. 4.1). In the proposed framework, we make model point trackers to assist each other based on their evaluation and trajectory correlation in order to obtain self-correcting tracking.

In this chapter, we first discuss the algorithm for tracking model points in Sec. 4.2. We then present the proposed performance measure and correction technique for the tracker in Sec. 4.3 and Sec. 4.4, respectively. Sec. 4.5 explains the DBN model discrete variables for the framework. Sec. 4.6 discusses the experimental validation and analysis, and Sec. 4.7 summarises the chapter.

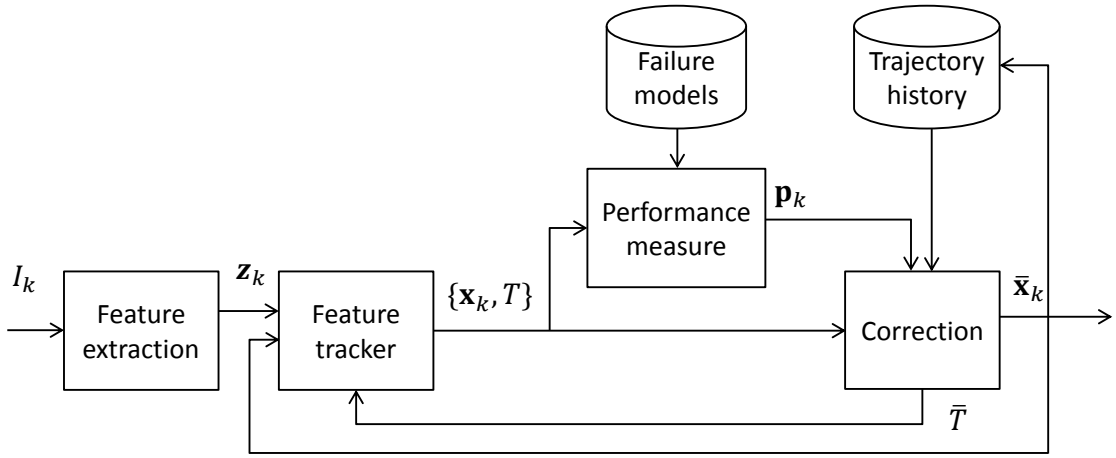


Figure 4.1: Block diagram of the proposed TEC framework for tracking of model points. From the image I_k, the feature extractor produces a set of measurement $\mathbf{z}_k$ and the baseline tracker estimates the set of tracks $\mathbf{x}_k$. A performance measure on the state $\mathbf{x}_k$ and tracker T gives a set of binary decisions $\mathbf{p}_k$ that expresses the track quality. Based on $\mathbf{p}_k$ a correction step produces a more accurate estimate $\bar{\mathbf{x}}_k$ and a corrected tracker $\bar{T}$. Failure models and the knowledge of the past trajectory information allow evaluation and correction in the framework.

4.2 Extended object tracker

In an extended object tracking the state $\mathbf{x}_k$ consists of the state (and identity) of each model point. Therefore, the state $\mathbf{x}_k$ can be expressed as

$$\mathbf{x}_k = \left\{ \mathbf{x}_k^i : 1 \leq i \leq N_k, i \in \mathbb{N}^+ \right\}, \tag{4.1}$$

where $\mathbf{x}_t^i$ is the state of the i^{th} model point and N_k is the number of estimated model points. For implementations with initiation and termination of model points, N_k is variable over time. Similarly, at each time the sensor or feature extractor produces M_k point measurements

$$\mathbf{z}_k = \left\{ \mathbf{z}_k^{\widehat{i}} : 1 \leq \widehat{i} \leq M_k, \widehat{i} \in \mathbb{N}^+ \right\}, \tag{4.2}$$

where $\mathbf{z}_k^{\widehat{i}}$ is the $\widehat{i}^{th}$ model point measurement. The measurements $\mathbf{z}_k$ are unlabelled and are affected by potential misdetections and clutter.

Individual allocated trackers for each model point are *local trackers* T^i. For tracking the local points, Bayesian filtering is used as a baseline tracker (Sec. 3.2) [97]. An analytical online solution to the integral in Eq. 3.1 is obtained by assuming a linear-Gaussian model as a Kalman filter. For non-linear and non-Gaussian models numerical approximations such as Monte Carlo methods are applied [22]. For marginalising analytically the state of the extended object, which

is proportional to the number of local trackers, we choose Kalman filter as the baseline tracker, T^i, for each model point. Moreover, we assume that the motion of model points is not complex (highly non-linear), and a linear motion model and a linear measurement model with Gaussian noise are adequate for state estimation [97, 38].

For a D-dimensional tracking problem, the state for each local point $\mathbf{x}_k^i$ in the Kalman filter are represented as

$$\mathbf{x}_k^i = [x_{1,k}^i, \dot{x}_{1,k}^i, ...x_{D,k}^i, \dot{x}_{D,k}^i]. \tag{4.3}$$

where x and $\dot{x}$ are the position and velocity components of the state. For the state models a constant velocity is assumed. The prediction model (state transition) is defined by the Gaussian distribution as

$$p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i) = \mathcal{N}(F_k\mathbf{x}_{k-1}^i, Q_k), \tag{4.4}$$

where $\mathcal{N}(\cdot)$ is a D-dimensional Gaussian probability density function with mean $F_k\mathbf{x}_{k-1}$ and covariance matrix Q_k. The constant velocity model is defined by the matrix F_k as

$$F_k = I_D \otimes \begin{bmatrix} 1 & \Delta k \\ 0 & 1 \end{bmatrix}, \tag{4.5}$$

where I_D is a $D \times D$ identity matrix and $\otimes$ represents the tensor product on matrices. Let us assume that the measurement $\widehat{\mathbf{z}_k^i}$ is associated with the local tracker T^i and the measurement model is defined as

$$p(\widehat{\mathbf{z}_k^i}|\mathbf{x}_k^i) = \mathcal{N}(H\mathbf{x}_k^i, R_k), \tag{4.6}$$

where R_k is the covariance matrix that represents the uncertainty in the measurement and H is the observation matrix, defined as

$$H = I_D \otimes \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{4.7}$$

Data association uncertainty arises between the local trackers, particularly when the object undergoes deformation or it is articulated. The uncertainty is due to the arrival of multiple measurements in a validation region (Fig. 4.2). The region is determined by the *gate probability* P_g of the predicted Gaussian distribution, $p(\mathbf{x}_k^i|\mathbf{Z}_{k-1})$. The multiple measurements obtained in the

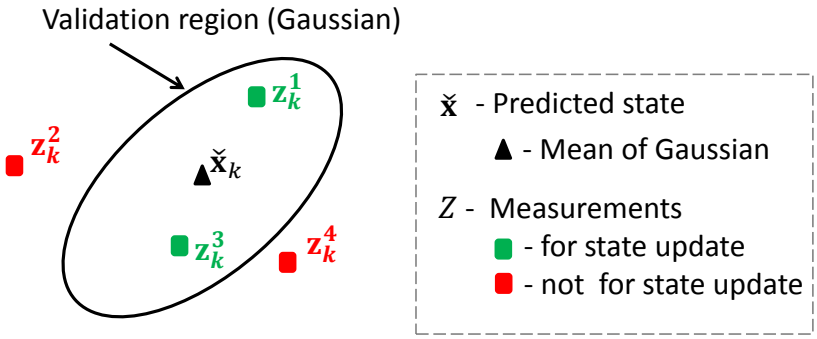


Figure 4.2: Data association problem with multiple measurements in the validation region at the update stage of the Kalman filter.

validation region are the result of other local point trackers or clutter. Optimal solutions for such scenarios are characterised by exponential complexity to the number of measurements [5]. A suboptimal solution is to apply a Joint Probability Data Association Filter (JPDAF) [29]. The JPDAF is implemented by using a Probability Data Association Filter (PDAF) to each model point independently. The separate PDAF implementation to a model point considers the other model points' measurements as clutter. PDAF allows the tracker to take into account the association uncertainty for measurements in the validation region. The soft decision of PDAF is based on *Minimum Mean Square Error* (MMSE), between the predicted state and measurements in the validation region. JPDAF is effective for tracking a known number of targets, which is considered in our framework, and in the presence of clutter [5]. However, JPDAF performance degrades significantly when neighbouring targets and clutter produce persistent interference, and when misdetections occur. We improve the performance of JPDAF in such conditions with the proposed TEC framework.

4.3 Quality measure

Sources of performance degradation are clutter (false measurements), absence of measurements (misdetections) and association errors. Fig. 4.3 shows the proposed tracking quality estimation for a model points tracker taking into consideration the aforementioned sources of failure. We model the sources of failure as information, which are represented by I_p in Table 3.1, in order to estimate the track quality. The details of the quality estimation are described below.

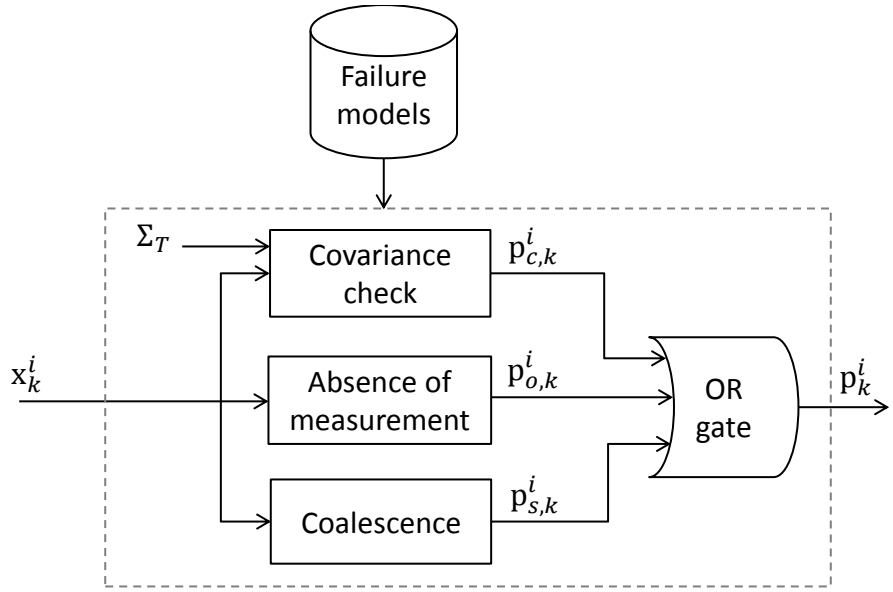
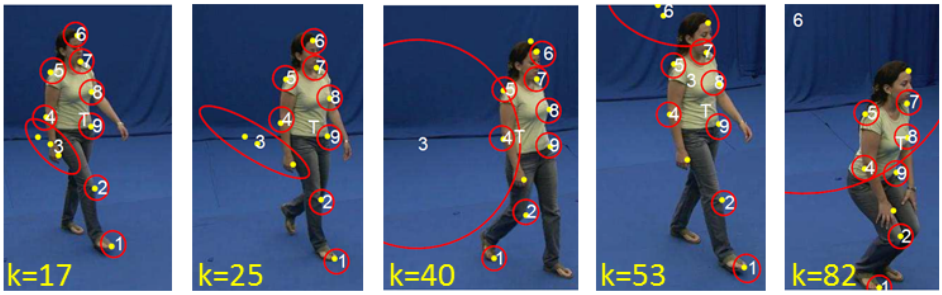


Figure 4.3: Performance measure of local trackers based on predefined failure models. Individual performance measures from the covariance check ($p^i_{c,k}$), absence of measurement ($p^i_{o,k}$) and coalescence ($p^i_{c,k}$) are combined to obtain the overall performance measure of a tracker p^i_k ($\mathbf{\Sigma}_T$ is a covariance threshold). The estimated quality values are used for the correction.

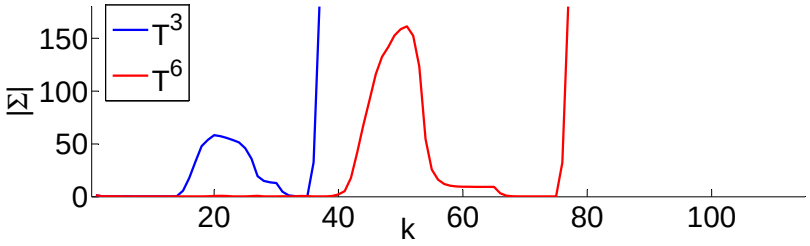
4.3.1 Addressing performance degradation

An increase in the *error covariance* of the Bayesian tracker is an indication of low confidence for the local tracker [105]. The increase is a result of associating more than one measurement data $\mathbf{z}_k$ in the validation region of the Kalman filter at the update step. The local points in the extended object come close to each other and lead to the observation of one model point in the validation region of the other model point trackers. The multiple measurements in the validation region are also a result of clutter. Let T^i represent i^{th} local point tracker. Fig. 4.4 shows the covariance characteristics for local point trackers under clutter (the position of the points are manually collected from a walking person for simulation). T^3 and T^6 have clutter at frames 17 and 50, respectively. The association uncertainties from the clutter increase the covariances of the local point trackers and finally lead to a track loss.

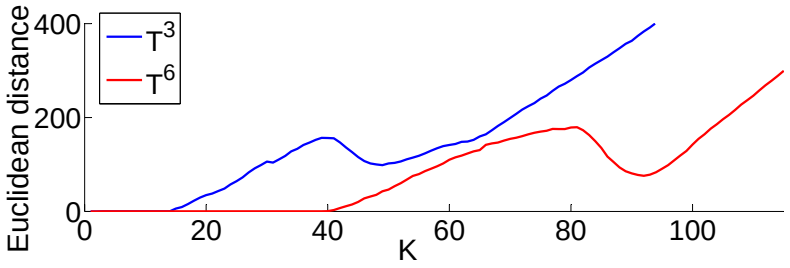
Let $\mathbf{\Sigma}$ be the covariance matrix in the Kalman filter and $p_{c,k}$ be the quality measure against the error in covariance (in the case of double subscript for p, the first subscript identifies the particular type of performance measure). We estimate $p_{c,k}$ by observing the magnitudes of the covariance matrix $|\mathbf{\Sigma}|$ as



(a)



(b)



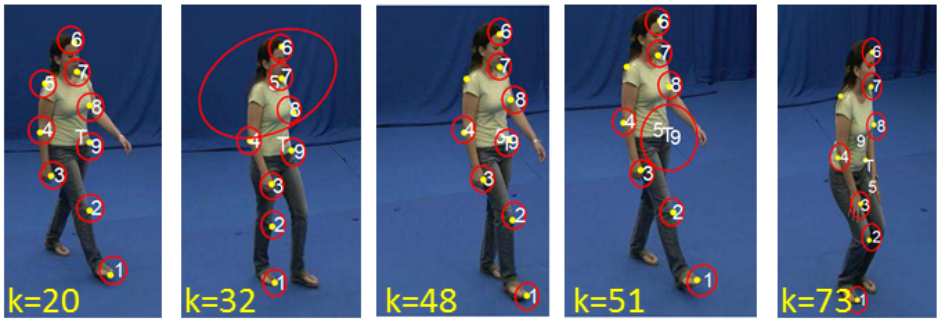
(c)

Figure 4.4: Increase in covariance of local point trackers due to multiple measurements in the validation region. (a) sample track, (b) covariance magnitude and (c) track error. T^3 and T^6 incur track losses due to clutter appearance in their validation region. The Euclidean distance is measured between local point track and ground-truth position.

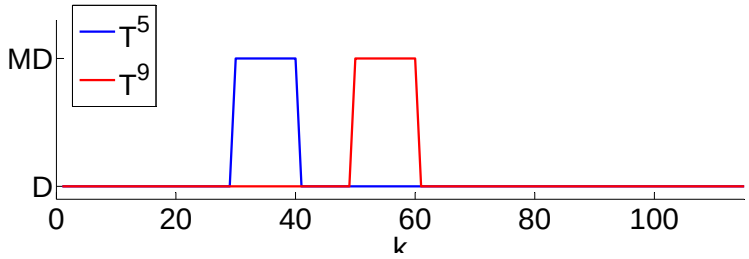
$$\mathrm{p}_{c,k} = \begin{cases} 1 & \text{if } |\mathbf{\Sigma}| \geq \mathbf{\Sigma}_T \\ 0 & \text{otherwise,} \end{cases} \tag{4.8}$$

where Σ_T is a threshold covariance used to estimate the trajectory. The term $|\mathbf{\Sigma}|$ is obtained by considering the individual matrix components value. For selecting the threshold, factors such as spatial distributions of points in the extended object are considered.

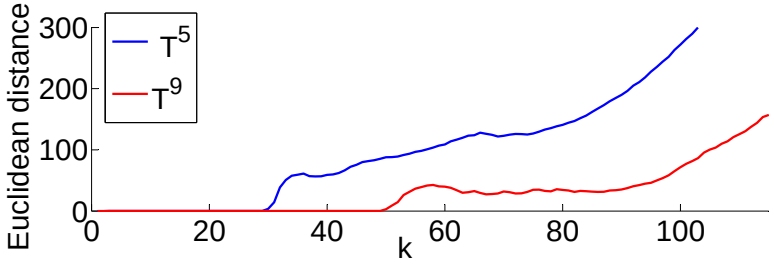
Misdetections due to sensor resolution or the feature extraction process cause an absence of measurement for the update stage. It is not possible to distinguish online if the absence of measurement is either from a tracking failure or from a misdetection. For such a reason, any absence of measurement at the update stage is used to estimate the track quality as a performance



(a)



(b)



(c)

Figure 4.5: Performance of local point trackers against misdetections. (a) sample track, (b) misdetections duration: D=detection and MD=misdetection, and (c) track error. A failure happens in T^5 and T^9 due to the misdetections of their model point measurement. The Euclidean distance is measured between local point track and ground-truth position.

due to absence of measurement $p_{o,k} = 1$. Fig. 4.5 shows the results of T^5 and T^7 when they undergo misdetections at frames 30 and 50, respectively. Both trackers failed in the subsequent frames.

Coalescence occurs when two or more local trackers are tracking the same model point. For example, after a crossover between targets, one tracker assumes the wrong model point and continues to track. Coalescence also occurs due to an increase in covariance, that in turn is caused by a misdetection and clutter, and the tracker starts to consider data from other model points than its own. This happens in multiple target tracking due to the inconsideration of joint associations for the measurements and trackers. The other performance measures mentioned, $p_{c,k}$ and $p_{o,k}$, do not handle such situations. The increase in covariance may not be noticeable before it is detected

by $\mathrm{p}_{c,k}$. Therefore, a performance measure against coalescence $\mathrm{p}_{s,k}$ is used to estimate the track quality. We estimate $\mathrm{p}_{s,k}$ by examining the local trackers and their associated measurements. Whenever a single measurement alone is associated to two or more trackers, we set $\mathrm{p}_{s,k} = 1$.

4.3.2 Strong and weak trackers

Considering the aforementioned low-quality tracking performance and sources of failure, the performance measure conducted on the output of the local trackers produces two classes: strong trackers and weak trackers. Weak trackers are not certain about the state they have estimated. Let $\mathrm{p}_k = 1$ and $\mathrm{p}_k = 0$ represent the decision given for weak and strong trackers, respectively. For each model point tracker the performance value $\mathrm{p}^i_k = \{0, 1\}$ is determined as

$$\mathrm{p}^i_k = \mathrm{p}^i_{c,k} \vee \mathrm{p}^i_{o,k} \vee \mathrm{p}^i_{s,k}, \tag{4.9}$$

where $\vee$ is a logical OR operator for combining performance measures (Fig. 4.3).

It is important to note that a weak tracker, for instance with large covariance, does not always imply tracking failure. The increase in covariance leads to the inclusion of other model points and also exposes it to a large amount of clutter in its validation region. Correcting this uncertainty, i.e. the low confidence, will avoid associating multiple points in the subsequent frames to improve the accuracy of the model point trackers. The information to make corrections on the identified weak trackers is obtained from either an offline constructed model [120] or learned online [48]. Online learned methods are based on decisions from the performance measure. Our correction step for weak trackers is based on states of strong trackers, and details are described in the next section.

4.4 Correction

The correction step estimates the most probable state of the weak tracker from the state of strong trackers. A correlation model from trajectories is used to estimate the state of one tracker from the other. The state estimation using the correlation model enables $\Theta(\cdot)$, where trajectories from trackers represent I_c (Eq. 3.14). Trajectories allow us to observe motion correlation of the state of model points over time. The correlation existing between local trackers of a generic extended object, such as articulated structure, is time dependent and has different degrees of correlation. The degree of correlation is the confidence in accurately estimating the state of one local tracker

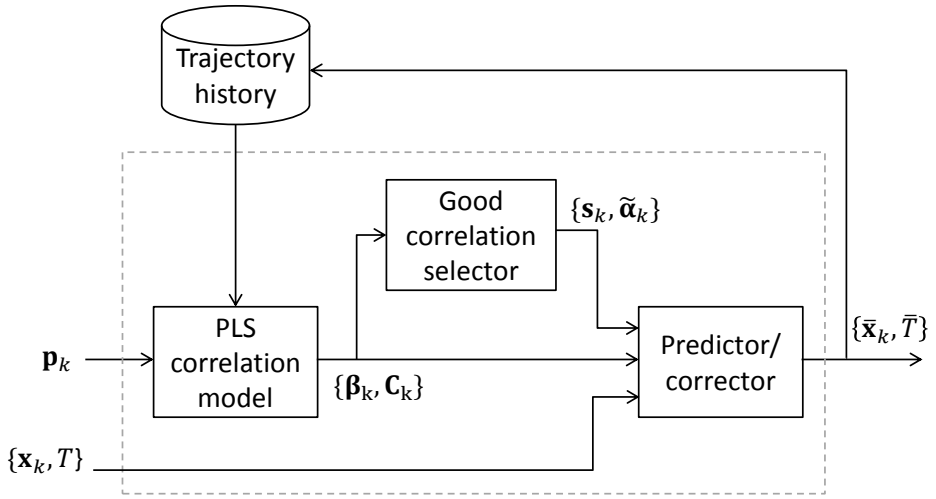


Figure 4.6: Correction step. The correlation model $\boldsymbol{\beta}_k$, constructed from short windowed trajectories, and its modelling error covariance $\mathbf{C}_k$ allow to select correlated local trackers $\mathbf{s}_k$ and their corresponding weight $\widetilde{\boldsymbol{\alpha}}_k$ according to the degree of correlation. The selected trackers with a high degree of correlation allow the estimation of the corrected state $\overline{\mathbf{x}}_k$ and tracker $\overline{T}$.

from the state of the others. The adaptive modelling of this correlation over time is important for estimating the probable state of the weak tracker.

The overall correction step comprises of three sub-steps (Fig. 4.6): correlation model constructor, correlated trackers selector and predictor/corrector. A *correlation model* is constructed for each set of weak trackers against a set of existing strong local trackers. From the correlation model, a *degree of correlation* between the weak tracker and the strong trackers is estimated by assigning normalised weights to the strong trackers. Finally, the state of the weak tracker is *corrected* by combining the outputs from the aforementioned two components.

4.4.1 Partial Least Square regression

Correlation is modelled using Partial Least Square (PLS) regression [120]. PLS regression models the relation between dependent and independent variables. The independent variables are the predictors and the dependent variables are the responses. Trajectories from strong and weak trackers are used as predictor and response variables, respectively.

In order to formulate the correlation model at each frame, let $\mathbf{w}_k$ and $\mathbf{s}_k$ be the set of indices for weak and strong local trackers, respectively. $\mathbf{w}_k$ and $\mathbf{s}_k$ are obtained from their performance evaluation $\mathbf{p}_k$ according to

$$i : 1 \leq i \leq N_k \in \begin{cases} \mathbf{w}_k & \text{if } p^i = 1 \\ \mathbf{s}_k & \text{otherwise.} \end{cases} \tag{4.10}$$

$\mathbf{w}_k = \{w_i : 1 \leq i \leq N_{W_k}, i \in \mathbb{N}^+\}$ and $\mathbf{s}_k = \{s_j : 1 \leq j \leq N_{S_k}, j \in \mathbb{N}^+\}$, where $N_{\mathbf{w}_k}$ and $N_{\mathbf{s}_k}$ are the number of weak and strong trackers, respectively, such that $N_{\mathbf{w}_k} + N_{\mathbf{s}_k} = N_k$. N_k is the number of local trackers, which is equal to the number of model points. For describing the correlation model, the pair between weak trackers, w_{ik}, and strong tracker, s_{j_k}, is used, and the combination of all the trackers is considered in Sec. 4.4.2. Trajectories are constructed from the position components of the tracker state. Let the trajectories at the current frame $\mathbf{\Gamma}_k^{w_i}$ and $\mathbf{\Gamma}_k^{s_j}$ for a weak and a strong tracker, respectively, be defined as

$$\begin{aligned} \mathbf{\Gamma}_k^{w_i} &= [\mathbf{x}_{k-1}^{w_i}, \mathbf{x}_{k-2}^{w_i}, ..., \mathbf{x}_{k-m}^{w_i}], \quad w_i \in \mathbf{w}_k, \\ \mathbf{\Gamma}_k^{s_j} &= [\mathbf{x}_{k-1}^{s_j}, \mathbf{x}_{k-2}^{s_j}, ..., \mathbf{x}_{k-m}^{s_j}], \quad s_j \in \mathbf{s}_k, \end{aligned} \tag{4.11}$$

where m is the trajectory length corresponding to a temporal window for modelling the correlation. The state $\mathbf{x}_k$ is a $D \times 1$ vector, and hence $\mathbf{\Gamma}_k$ has a dimensions of $D \times m$.

PLS regression allows the correlation model to be learnt by using the observed trajectories as a training data. The learnt model is used to estimate the correct state of a weak tracker by using the state of a strong tracker. The correlation model $\boldsymbol{\beta}_k^{w_i s_j}$ between a weak tracker w_i and strong tracker s_j is estimated using PLS as

$$\boldsymbol{\beta}_k^{w_i s_j} = \mathbf{\Gamma}_k^{s_j} V (U^{\mathrm{T}} \mathbf{\Gamma}_k^{s_j} \mathbf{\Gamma}_k^{s_j\,\mathrm{T}} V) U^{\mathrm{T}} \mathbf{\Gamma}_k^{w_i}, \tag{4.12}$$

where U and V are the component matrices for the predictor variable ($\mathbf{\Gamma}_k^{s_j}$) and the response variable ($\mathbf{\Gamma}_k^{w_i}$), respectively using principal component analysis. The superscript T represents the transpose operator [85].

The correlation model allows the estimation of the corrected state of weak trackers $\overline{\mathbf{x}}_k^{w_i}$ (Eq. 3.14) given the state of strong trackers through a probability density function $p(\overline{\mathbf{x}}_k^{w_i} | \mathbf{x}_k^{s_j}, \boldsymbol{\beta}_k^{w_i s_j})$. The probability density is estimated from a motion model for predicting the weak tracker w_i from the strong local tracker s_j, and is given as

$$\overline{\mathbf{x}}_k^{w_i} = \mathbf{A}_k^{w_i s_j} \mathbf{x}_k^{s_j} + \mathbf{b}_k^{w_i s_j}. \tag{4.13}$$

The matrix $\mathbf{A}$ and translation vector $\mathbf{b}$ are obtained directly from the model $\boldsymbol{\beta}^{w_i s_j}$ ($\boldsymbol{\beta}^{w_i s_j} =$

$[\mathbf{A}_k^{w_i s_j}, \mathbf{b}_k^{w_i s_j}]$). $\mathbf{A}^{w_i s_j}$ and $\mathbf{b}^{w_i s_j}$ have dimensions of $D \times D$ and $D \times 1$, respectively.

Due to absence of perfect correlations in the training data (i.e. between $\mathbf{\Gamma}_k^{s_j}$ and $\mathbf{\Gamma}_k^{w_j}$), the model $\boldsymbol{\beta}_k^{w_i s_j}$ has a residual fitting error $\mathbf{E}_k^{w_i s_j}$, and is calculated as

$$\mathbf{E}_k^{w_i s_j} = \mathbf{\Gamma}_k^{w_i} - (\mathbf{A}_k^{w_i s_j} \mathbf{\Gamma}_k^{s_j} + \mathbf{B}_k^{w_i s_j}), \tag{4.14}$$

where $\mathbf{B}_k^{w_i s_j}$ is a matrix of dimension $D \times m$ obtained by concatenation of m vectors of $\mathbf{b}_k^{w_i s_j}$. $\mathbf{E}_k^{w_i s_j}$ has a dimension $D \times m$, where the D rows correspond to the dimension of the states and the m columns correspond to the trajectory history considered from the current frame of reference.

PLS estimates the correlation model by minimising the variance of $\mathbf{E}_k^{w_i s_j}$. As a result, the mean of $\mathbf{E}_k^{w_i s_j}$ is very small compared to the individual m errors of the trajectory component. Assuming the small mean error value of $\mathbf{E}_k^{w_i s_j}$, the covariance matrix $\mathbf{C}_k^{w_i s_j}$ for estimating the state of weak trackers from the state of strong trackers in the training data is calculated as

$$\mathbf{C}_k^{w_i s_j} = \frac{1}{m} \mathbf{E}_k^{w_i s_j} \mathbf{E}_k^{w_i s_j \mathrm{T}}. \tag{4.15}$$

$\mathbf{C}_k$ has dimensions $D \times D$.

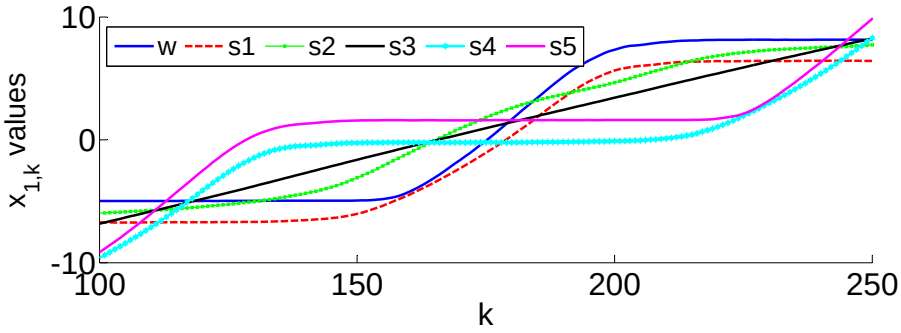
The state $\overline{\mathbf{x}}_k^{w_i}$ is determined from the state of a strong tracker, $\mathbf{x}_k^{s_j}$, as

$$p(\overline{\mathbf{x}}_k^{w_i} | \mathbf{x}_k^{s_j}) = \mathcal{N}(\mathbf{A}_k^{w_i s_j} \mathbf{x}_k^{s_j} + \mathbf{b}_k^{w_i s_j}, \mathbf{C}_k^{w_i s_j}), \tag{4.16}$$

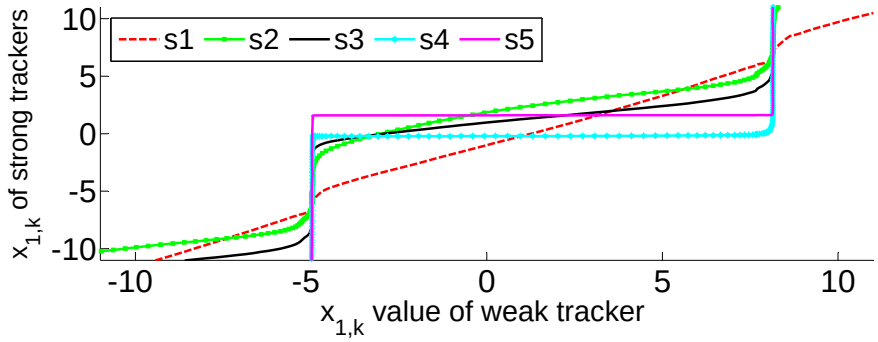
where $\mathcal{N}$ is a D-dimensional Gaussian density function (Eq. 4.4). The correction of a weak tracker is done in a Gaussian function form, due to the fact that the baseline tracker is the Kalman filter. The mean value of the Gaussian is the most probable position as estimated by other trackers' states, and the covariance is proportional to $\mathbf{E}^{w_i s_j}$.

4.4.2 Correlated trackers selection

The degree of correlation for a particular weak tracker $w_i \in \mathbf{w}_k$ to a set of strong trackers $\mathbf{s}_k = \{s_j\}_{j=1}^{N_{S_k}}$ varies from one strong tracker to another. For articulated structures, subsets of model points with good correlation are those laying in the same rigid structure relative to the overall object. In order to obtain the most accurate state estimation of the weak tracker, the best correlations need to be identified among the set of strong trackers. Fig. 4.7 shows an example of a trajectory for a particular weak tracker w and a list of available strong trackers ($s1$, $s2$, $s3$, $s4$ and $s5$): among the existing strong trackers $s1$ shows the best correlation to the weak tracker while $s5$



(a)



(b)

Figure 4.7: Trajectory correlations between trackers. In the plot, the $x_{1,k}$ values represent the trajectory values obtained by considering the first component of the state vector $\mathbf{x}_k$ (Eq. 4.3 and 4.11). (a) A weak tracker and a set of strong trackers. (b) A weak tracker against a list of strong trackers.

is the worst.

For each candidate weak tracker $w_i \in \mathbf{w}_k$, a new set of strong trackers

$$\mathbf{g}_k^{w_i} = \left\{ g_n^{w_i} : 1 \leq n \leq N_{S_k}, n \in \mathbb{N}^+ \right\}, \tag{4.17}$$

are selected from the original list $\mathbf{s}_k$ (the superscript identifies the set particular to weak tracker w_i). The new array $\mathbf{g}_k^{w_i}$ contains indices of strong trackers in decreasing order of correlation with tracker w_i. $\mathbf{g}_k^{w_i}$ is obtained by considering the PLS correlation model covariance $\mathbf{C}_k^{w_i s_j}$ (Eq. 4.15). Strong trackers in $\mathbf{g}_k^{w_i}$ are ordered according to the formulation

$$|\mathbf{C}_k^{w_i g_{(n)}^{w_i}}| < |\mathbf{C}_k^{w_i g_{n+1}^{w_i}}|, \quad 1 \leq n \leq N_{S_k}. \tag{4.18}$$

Ranking the correlation level is important since the level is directly proportional to the prediction accuracy. Fig. 4.8 shows how the correlation between two local tracks is estimated from the

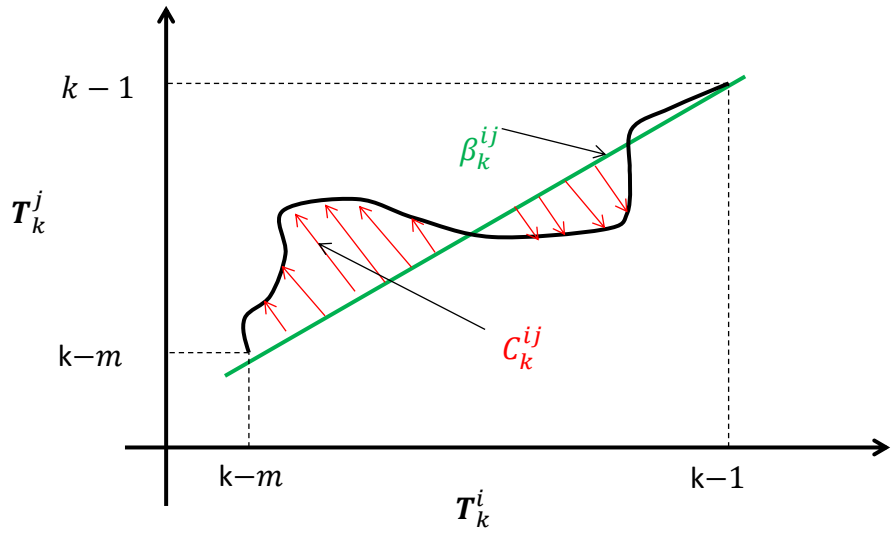


Figure 4.8: Trajectories $\boldsymbol{\Gamma}_k^{s_j}$ and $\boldsymbol{\Gamma}_k^{w_i}$ from a strong and a weak tracker, respectively, used for training data in order to construct the correlation model $\boldsymbol{\beta}_k^{w_i s_j}$. The correlation model has a covariance $\mathbf{C}_k^{w_i s_j}$. The fitting error from the constructed model is used to measure the correlation between strong and weak local trackers.

regression model. The magnitude of the red arrows indicates how large the fitting error between the constructed model and the observed trajectories is. The correlation level is determined by considering the magnitudes of these residual errors for the trackers.

For accurate state estimation, only the first γ elements of $\mathbf{g}_k^{w_i}$ are selected as predictor by considering the magnitude of the covariance in the PLS regression model:

$$|\mathbf{C}^{w_i g_p^{w_i}}| < C_T, \quad 1 \leq p \leq \gamma, p \in \mathbb{N}^+, \tag{4.19}$$

where C_T is a threshold covariance magnitude. The scale of the object and the spatial distribution of the model points are considered as factors for determining the magnitude of C_T for a particular application. Additionally, the optimised value of the state covariance in the baseline tracker (Eq. 4.4) allows C_T to be selected.

The selected γ strong local trackers from $\mathbf{g}_k^{w_i}$ are assigned a weight $\boldsymbol{\alpha}_k$ which is inversely proportional to the magnitude of their covariance in the PLS model

$$\alpha_k^{w_i g_p^{w_i}} \propto |\mathbf{C}^{w_i g_p^{w_i}}|^{-1}. \tag{4.20}$$

The estimated weight gives the goodness of correlation level and accuracy of prediction from the learned model. In order to have a proper probability density, the weights for the selected local

strong trackers are normalised at each time step as

$$\widetilde{\alpha}_k^{w_i g_p^{w_i}} = \frac{\alpha_k^{w_i g_p^{w_i}}}{\sum_{p=1}^{\gamma} \alpha_k^{w_i g_p^{w_i}}}. \tag{4.21}$$

4.4.3 Correction

For each weak tracker w_i, the selected γ strong trackers jointly contribute to estimate $\overline{\mathbf{x}}_k^{w_i}$ as

$$p(\overline{\mathbf{x}}_k^{w_i}) = p(\overline{\mathbf{x}}_k^{w_i} | \mathbf{x}_k^{g_{1:\gamma}^{w_i}}) = \mathcal{N}(\mathbf{x}_{k,mean}^{w_i}, \mathbf{C}_k^{w_i}), \tag{4.22}$$

where

$$\mathbf{x}_{k,mean}^{w_i} = \sum_{p=1}^{\gamma} \widetilde{\alpha}_k^{w_i g_p^{w_i}} (\mathbf{A}_k^{w_i g_p^{w_i}} \mathbf{x}_k^{g_p^{w_i}} + \mathbf{b}_k^{w_i g_p^{w_i}}),$$

and

$$\mathbf{C}_k^{w_i} = \sum_{p=1}^{\gamma} \widetilde{\alpha}_k^{w_i g_p^{w_i}} \mathbf{C}_k^{w_i g_p^{w_i}}.$$

We used the joint prediction from γ trackers rather than the use of only prediction from the best correlated tracker (i.e. equivalent to maximum likelihood) in order to minimize the effect of an inaccurately learnt model and also to reduce the effect of fast changing correlation model between trackers.

It is worth mentioning that the set of strong tracker indices for the available two or more weak trackers are the same at each step of correction. However, the selected γ strong predictor trackers for a particular weak local tracker are different and are assigned with weights according to their constructed regression models. Eq. 4.22 represents the function $\Theta(\cdot)$ in Eq. 3.14, and the state $p(\overline{\mathbf{x}}_k^{w_i})$ is used as the corrected state output and prior information for the execution of the baseline tracker. The correction is a form of Kalman filter *re-initialisation*, which estimates $\overline{\mathbf{x}}_k$ and changes the prior information of T in order to obtain $\overline{T}$.

4.5 Discrete variables of the DBN model

The mapping function f_c^i described in Table 3.1 (Eq. 3.19) to obtain the probability distribution for correction variable, $\mathbf{c}_k^i$, from its evaluation variable, $\mathbf{p}_k^i$, and other model point tracker j evaluation variable, $\mathbf{p}_k^j$, is shown in Fig. 4.9. The correction variable has two discrete values $\mathbf{c}_k^i = \{0, 1\}$, where the values 1 and 0 correspond to the mode of operation with and without cor-

	$[\mathrm{p}^i, \mathrm{p}^j]$			
	[0,0]	[0,1]	[1,0]	[1,1]
c^i	0	0	1	0
c^j	0	1	0	0

Figure 4.9: Mapping function f_c^i (Table 3.1) to obtain the values of $\mathbf{c}_k^i$ for tracker i from the value of its $\mathbf{p}_k^i$ and the value of $\mathbf{p}_k^j$ from tracker j.

rection, respectively. For the weak local tracker w_i, i.e. when $p(\mathbf{c}_k^{w_i} = 1) = 1$ and $p(\mathbf{c}_k^{w_i} = 0) = 0$, correction is made by complete re-initialisation of a tracker. The prior correction for the tracker is formulated as,

$$p(\mathbf{x}_k^{w_i}|\mathbf{Z}_k, \mathbf{c}_k^{w_i} = c) = \begin{cases} p(\mathbf{x}_k^{w_i}|\mathbf{Z}_k) & \text{if } c = 0, \\ p(\overline{\mathbf{x}}_k^{w_i}|\mathbf{x}_k^{\mathbf{g}^{w_i}}), & \text{if } c = 1, \end{cases} \tag{4.23}$$

where $p(\overline{\mathbf{x}}_k^{w_i}|\mathbf{x}_k^{\mathbf{g}})$ is the re-initialisation of tracker w_i from list of strong trackers $\mathbf{g}^{\mathbf{w_i}}$ described in Eq 4.22. In the correlation based-self correcting tracking, the motion model and observation model are independent of $\mathbf{c}_k$ resulting in

$$\begin{aligned} p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{c}_{k-1}^i) &= p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i), \\ p(\mathbf{z}_k|\mathbf{x}_k^i, \mathbf{c}_{k-1}^i) &= p(\mathbf{z}_k|\mathbf{x}_k^i). \end{aligned} \tag{4.24}$$

Figure 4.10 shows the performance of two local trackers from the baseline tracker T^1 and T^2, and their correlation-based self-correcting tracker $\overline{T}^1$ and $\overline{T}^2$, respectively. In the self-correcting tracker, the selected discrete correction variable values c, i.e. $p(\mathbf{c}_k = 1)$, are shown to indicate the times in which correction is made on the baseline tracker (4.10(b) and 4.10(d)). For T^1, corrections made around $k = 55$ has no significance over the baseline tracker, however the corrections made around $k = 180$ play an important role for recovering the baseline tracker from failures. For tracker T^2, the first correction made on the baseline tracker allows to recover from its failure.

4.6 Experimental results and analysis

We present a quantitative performance evaluation of the proposed TEC framework by tracking markers from a motion capture system. Tracking of the markers involves matching corresponding

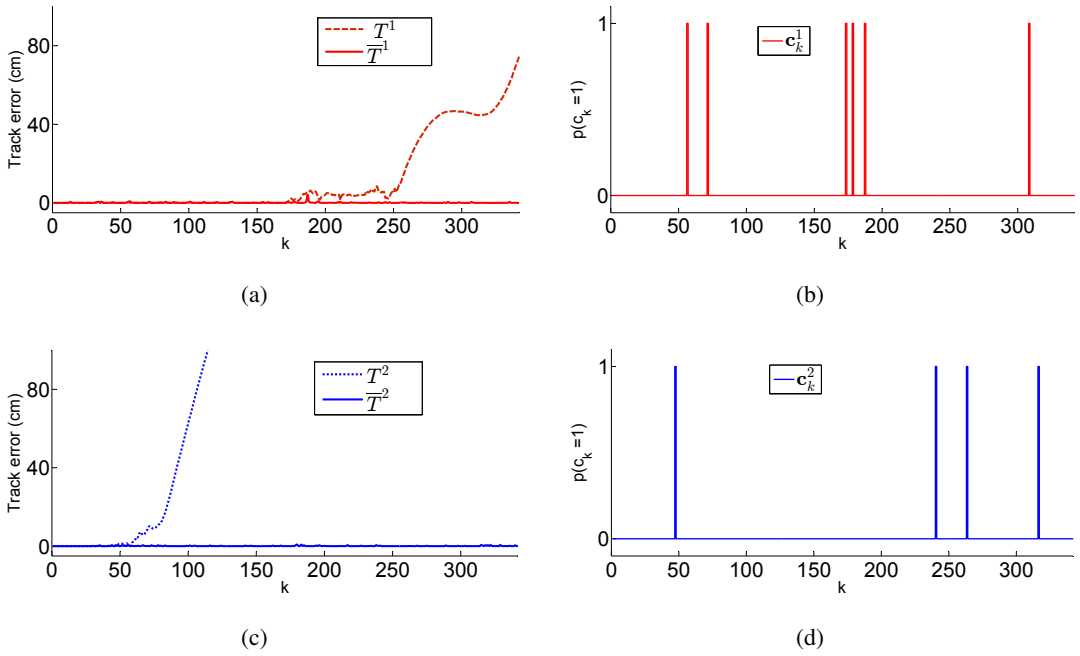


Figure 4.10: Comparison of tracking errors in the baseline tracker (T) and in the correlation-based self-correcting tracker ($\overline{T}$) for local point tracking. Track errors are euclidean distance between the track estimate and ground-truth state. Tracker 1: (a) tracker error and (b) values of $\mathbf{c}_k^1$ with $p(\mathbf{c}_k^1 = c) = 1$ in self-correcting tracking. Tracker 2: (c) tracker error and (d) values of $\mathbf{c}_k^2$ with $p(\mathbf{c}_k^2 = c) = 1$ in self-correcting tracking.

marker indices at different frames from the list of unlabelled points. Fig. 4.11 shows a typical motion capture system [34]. At each frame, in the measured two-dimensional and three-dimensional points there are misdetections and false detections (clutter) of markers. Misdetections are the result of occlusions and wrong spatial alignment of the two-dimensional points detected by cameras in the motion capture system. False detections are common when multiple cameras are used and their detected two-dimensional points are cluttered. Using the markers, we compare the results of TEC framework with the baseline tracker JPDAF [23][1] [5]. JPDAF and TEC for 30 targets run on average 30 and 26 frames per second, respectively, (without clutter) and 16 and 12 frames per second (with 100 as the amount of clutter) on an Intel i5-3570k, 3.4GHz CPU with 8GB RAM on Windows 7 (non-optimised MATLAB2013b code). We also compare TEC with other state-of-the-art multi-target trackers: two online methods - Kalman filters using Hungarian algorithm and nearest-neighbour data associations HDAF [18][2] and NNDAF, respectively, and two offline

[1]code:-`http://www.mathworks.co.uk/matlabcentral/fileexchange/34146`, Last accessed: March 2014

[2]code:-`http://studentdavestutorials.weebly.com/multi-bugobject-tracking.html`, Last accessed: March 2014

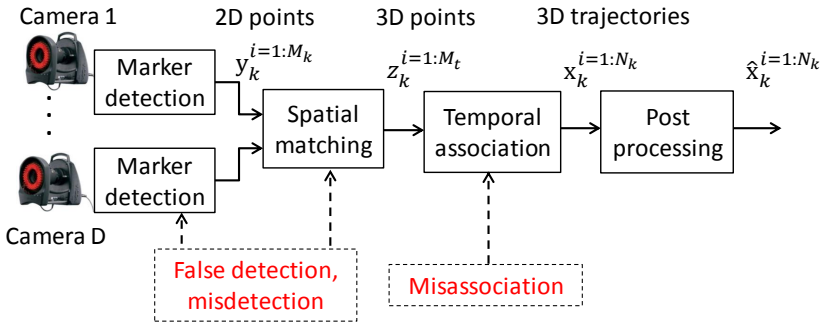


Figure 4.11: Motion capture pipeline. 2D points $\mathbf{y}_k^{i=1:M_k}$ obtained by each camera are used to estimate the absolute 3D position of each marker indices. The obtained 3D points $\mathbf{z}_k^{i=1:M_k}$ are associated and tracked at each time to create the markers' trajectories. Misdetections, false detections and data association problems generate errors. Finally, off-line post-processing remove the outliers.

methods - Hungarian-assessment-based particle tracker PT [104][3] and motion dynamics-based assessment for tracking targets with similar appearances SA[4] [19].

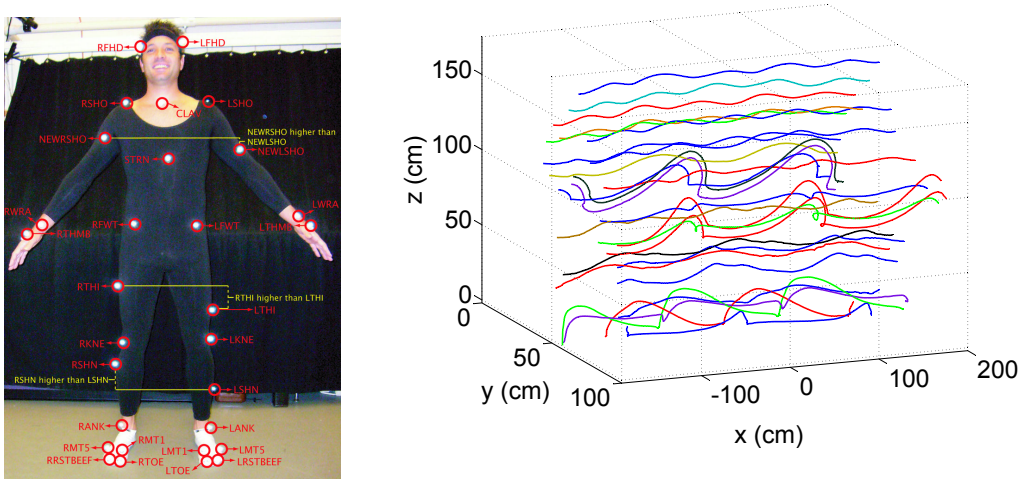
4.6.1 Experimental setup

We use the Carnegie Mellon University (CMU) [1] and HumanEva [2] Motion Capture database. The data is from a human performing different actions. For the CMU dataset, the front snapshot of the human body with the marker indices are shown in Fig. 4.12. The subject has 41 markers, and the data include absolute position (three-dimensional) and orientation information. We select 39 markers, and their three-dimensional absolute positions are read from C3D file format in the database. Two of the markers are discarded in the experimentation since their initial frame measurements are either incorrect or not available. Similarly, for the HumanEva dataset we use 39 markers.

The comparison of TEC with JPDAF is done by using three-dimensional marker points, while the comparison with other state-of-the-art trackers PT, SA, HDAF and NNDAF (including JPDAF) is done on two-dimensional marker points as the trackers implementation is for two-dimensional targets. The two-dimensional markers are obtained by removing the less variable coordinate component followed by removing markers indices that overlap and are very close to each other in their two-dimensional view for the whole sequence. The labels obtained from the dataset are used as a ground-truth for evaluating tracking results. Moreover, sources of failures

[3]code:-`http://www.mathworks.co.uk/matlabcentral/fileexchange/34040`, Last accessed: March 2014

[4]code:-`https://bitbucket.org/cdicle/smot`, Last accessed: March 2014



(a) (b)

Figure 4.12: Markers, their positions on the body and an example of their evolution over time. (a) Labels of markers on the object (image from CMU human motion data capture [1]). (b) Ground-truth three-dimensional trajectories for selected markers.

Table 4.1: Dataset used for validating the proposed Track-Evaluate-Correct approach. Legend-3D: three-dimensional dataset from CMU [1] and HumanEva [2], 3D-m: 3D with misdetection, 3D-c: 3D with clutter, 2D-m and 2D-c: two-dimensional of the 3D dataset with misdetections and clutter, respectively, FL: frame length, NM: number of markers, SF: source of failure

Dataset	**Motion**	**FL**	**NM**	**SF**
3D	Walking	342	39	-
	Running	172	39	
	Dancing	867	39	
	WalkingEva	1877	39	
	JoggingEva	1701	39	
3D-m	3D			Misdetection
3D-c	3D			Clutter
2D	Walking	342	27	-
	Running	172	23	
	Dancing	320	29	
2D-m	2D			Misdetection
2D-c	2D			Clutter

are added to the original data (Table 4.1). 3*D* is the original dataset obtained from CMU and HumanEva, while 3*D-m* and 3*D-c* represent the data with imposed misdetections and clutter, respectively. 2*D-m* and 2*D-c* are two dimensions of the original 3*D* dataset with misdetections and clutter, respectively. The added clutter appears only in the rectangular volume occupied by the object for the 3*D* dataset and in the image plane view of the object for the 2*D* dataset. We test the experimentation for a variable amount of clutter, misdetection length durations and

Table 4.2: Parameters and values used to generate *DS-m* and *DS-c* from Dataset *DS*. Legend- φ: uniformly-distributed pseudo-random number generator function, Δ: difference between maximum and minimum values of the marker states, x: position component of the object state in the 3D space.

Dataset	Parameters	Symbol	Value/function
3D-m	duration of misdetection	L_{md}	$1 \leq L_{md} \leq 15$ frames
	misdetected marker number	N_{md}	10
	misdetected marker index	I_{md}	Ψ
3D-c	amount of clutter	N_c	$100 \leq N_c \leq 600$
	volume occupied by the object	V	$\Delta x_1 * \Delta x_2 * \Delta x_3$
	state of clutter	$\mathbf{x}_c$	$V * \varphi$
2D-m	probability of misdetection	$\%MD$	0%, 1%, 5%, 10%
	fixed amount of clutter	$\%N_c$	50%
	misdetected marker index	I_{md}	φ
2D-c	percentage of clutter	$\%N_c$	$0\%, 50\%, 100\%, 200\%$
	state of clutter	$\mathbf{x}_c$	φ
	fixed probability of misdetections	$\%MD$	5%

probability of misdetections (see Table 4.2).

We use the same tracking parameters for each type of motion sequences, and optimal parameters are selected by examining the results. The matrices Q_k and R_k in the Kalman filters of JPDAF are set according to the formulation in [6] (variances of 20cm and 5cm, respectively). The gate probability for the Gaussian distribution is set to 0.99. As the quality measure for covariance grow, we use $\mathbf{\Sigma}_T$ between $2\mathbf{\Sigma}_{int}$ and $4\mathbf{\Sigma}_{int}$ ($\mathbf{\Sigma}_{int}$ is the initial covariance matrix in the Kalman filter estimated from Q_k). A trajectory length of $m = 12$ is used to train the correlation model. Long trajectories are expensive in terms of computation and do not model appropriately fast changing correlations between states of model points. We use $\gamma = 4$ for selecting the number of best predictor trackers, and zero weights are assigned to the rest of the trackers (Eq. 4.21). Kalman filters in HDAF and NNDF are set similarly to JPDAF. The minimum association threshold for HDAF is set to $10cm$. For SA tracker, we use a singular threshold value of 0.3, a time window of 80 frames and ADMM method. For PT, we use the default parameters as provided in the code and tracklets shorter than 30 frames are removed.

4.6.2 Evaluation measures

Tracking results are compared to the labelled ground-truth data based on Optimal Sub-Pattern Assignment (OSPA) [92]. Let $\mathbf{O} = \left\{\mathbf{o}_k^{\bar{i}}\right\}_{\bar{i}=1}^{G_k}$, $\bar{i} \in \mathbb{N}^+$ be the labelled ground-truth data, where G_k

Table 4.3: Summary of parameters used for correlation based correction of model point trackers. The term Σ_{int} is initial covariance matrix of Kalman filter.

Parameter name	Symbol	Value	Reference
Threshold covariance	Σ_T	$2\Sigma_{int}$ - $2\Sigma_{int}$	Eq. 4.8
Trajectory length	m	12	Eq. 4.11
Number of Predictor trackers	γ	4	Eq. 4.22

is the number of targets at each time. The OSPA distance $\mathbf{D}_k$ is estimated according to

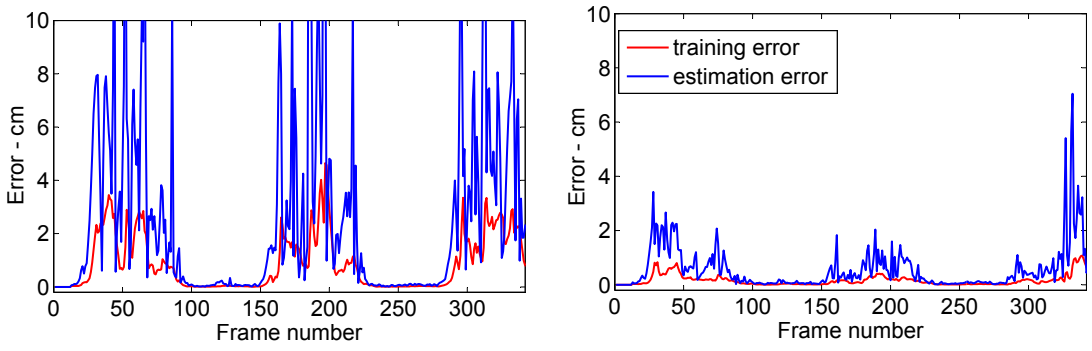
$$\begin{aligned}&\mathbf{D}_k(\mathbf{x}_k,\mathbf{o}_k) = \\ &= \left[\frac{1}{max(N_k,G_k)}\left(\min_{\pi\in\Pi_{G_k}}\sum_{l=1}^{N_k} d_{\grave{c}}(\mathbf{x}_k^i,\mathbf{o}_k^{\bar{i}})^a + |G_k - N_k|.\grave{c}^a\right)\right]^{1/a},\end{aligned} \tag{4.25}$$

where Π_{G_k} is a set of permutations of length N_t taken from the $\{1,2\cdots G_k\}$, $d_{\grave{c}}(\mathbf{x}_k^i,\mathbf{o}_k^{\bar{i}}) = min(\grave{c}, d(\mathbf{x}_k^i,\mathbf{o}_k^{\bar{i}}))$ is a cut of distance with $\grave{c} > 0$, $d(\mathbf{x}_k^i,\mathbf{o}_k^{\bar{i}})$ is the base distance (i.e. Euclidean) and a is the order of OSPA metric. We use the MATLAB code provided in [92] for estimating $\mathbf{D}$, with $\grave{c} = 10cm$ by considering the spatial distance between markers, and $a = 1$.

To characterise the overall tracker performance, we also count the number of *False Positive* FP for local track output components. For a frame length of ν, the FP level on a local tracker i is

$$FP = \begin{cases} 1 & \text{if } d_k^i \geq d_{Th}, k \in \nu \\ 0 & \text{otherwise,} \end{cases} \tag{4.26}$$

where q_{Th} is the thresholds of error value. In the reported results, we use $d_{Th} = 10\ cm$ and $\nu > 10$. We calculate FP for online trackers only. The online trackers considered are initialised based on the ground truth information in the first frame, and hence association between the trackers result and the ground truth is known. However, the offline trackers are based on optimization over all the sequences and from the result the association with the ground truth is not known. The offline trackers also produce a large number of trajectories compared the number of targets in the ground truth information particularly in the presence of clutter. In such cases, association of the tracker's result with the ground truth is not known and we cannot estimate the FP.



(a) (b)

Figure 4.13: PLS regression model (training) errors and estimation errors for marker RTOE (see Fig. 4.12) using the states of the other marker indices: (a) using LTOE; (b) using RWRA.

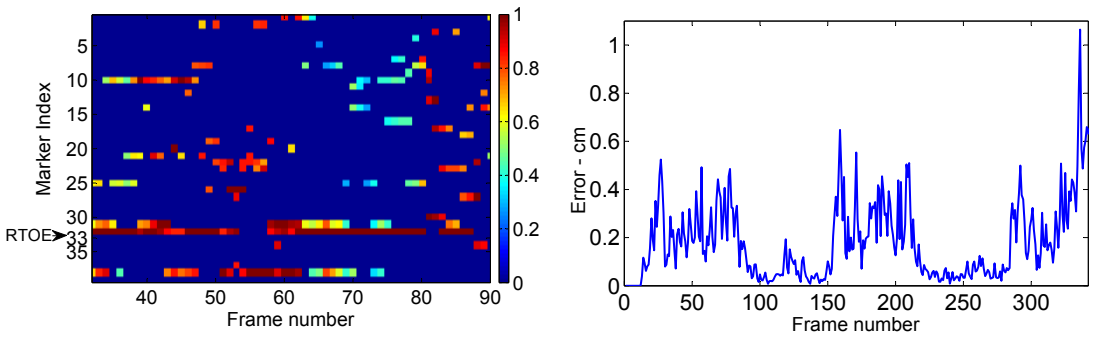
Table 4.4: Mean and standard deviation of errors (cm) for the PLS regression model over the walking sequence for different marker selected (see fig. 4.12). Legend: μ=mean, σ=standard deviation, P= predictors variables (row space), R=response variables (column space).

P/R	RTOE		LTOE		RWRA		LKNE	
	μ	σ	μ	σ	μ	σ	μ	σ
RTOE	-	-	1.61	2.94	1.65	2.17	1.43	1.95
LTOE	2.61	3.78	-	-	1.14	1.59	1.09	1.61
RWRA	0.57	0.83	0.27	0.32	-	-	0.18	0.18
LKNE	0.86	1.08	0.34	0.48	0.29	0.24	-	-
LSHO	0.51	0.66	0.38	0.65	0.24	0.23	0.19	0.16
RFHD	0.51	0.67	0.36	0.50	0.24	0.26	0.19	0.14
LWRA	0.41	0.43	0.71	1.31	0.43	0.71	0.34	0.53

4.6.3 Discussion

The first set of experiments examines how accurately the PLS regression model can estimate the state of one marker from the states of other markers. Fig. 4.13 shows the results of the state estimation accuracy with the corresponding regression model covariance (Eq. 4.20). The covariance of the regression model is estimated from the training error according to Eq. 4.15, which measures estimation accuracy. The results indicate that different local point trackers have different state estimation accuracy to a specific local point tracker and the accuracy is variable in time. This property is noticeable for deformable objects and articulated structures, such as human motion which is considered here. Table 4.4 gives the mean and standard deviation errors for estimating the state of one marker from the state of other markers.

Figure 4.14 shows the result of state estimation of marker index RTOE (see Fig. 4.12) by weighting adaptively the estimation of other markers. The weights assigned to each predictor



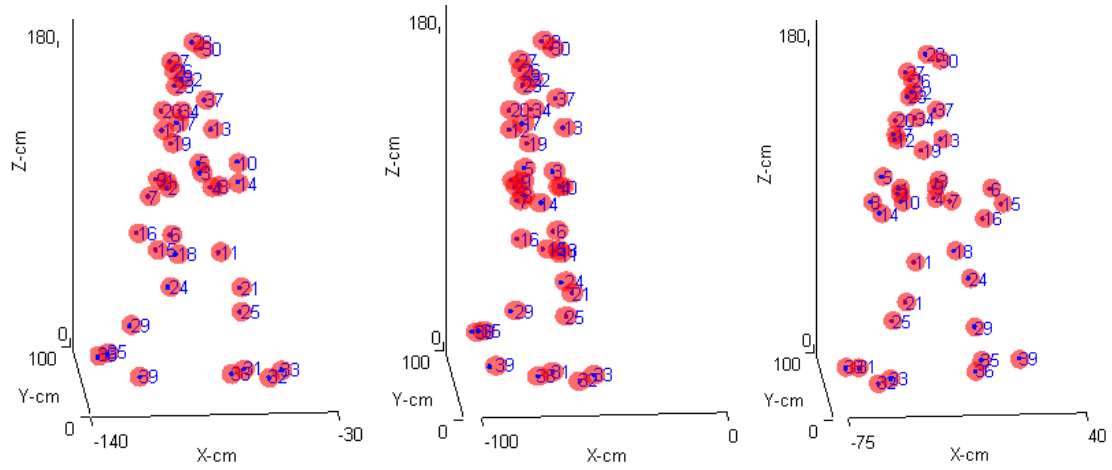
(a) (b)

Figure 4.14: Selection of best predictor indices. (a) Heat map of weights calculated for predictor marker indices for estimating the state of marker RTOE. (b) Estimation error of marker RTOE from the weighted estimate of all the other markers.

trackers (i.e. strong trackers) are indicated in the heat map (Fig. 4.14(a)). Due to the dynamic changing nature of human motion, the importance of adaptive modeling of the correlation is shown from the scattered weights assigned to the predictor marker indices in the heat map. From the map it is possible to see that for RTOE the marker index 32, just above it, is more correlated for large frame number durations. Fig. 4.14(b) shows the state estimation error of RTOE from the weighted prediction of all the other markers. The weighted estimate shows smaller error than the estimate errors by individual indices (Fig. 4.13).

Tracking on 3D data involves creating labels (data association) for each marker index at each frame in the sequence. For this dataset, the performance of JPDAF, without TEC, produces good tracking results. For the Dancing sequence $FP = 1$, while for the Walking, Running, WalkingEva and JoggingEva $FP = 0$. Fig. 4.15 shows sample tracking results for the walking sequence of 3D. The 3D ellipses represent the validation region of the marker trackers. Video results for the sequences described in Table 4.1 are available in `http://www.eecs.qmul.ac.uk/~andrea/tec.html`. The website contains videos showing the results of our proposed method and trackers used for comparison. The video results allow one to compare qualitatively the performance of our proposed framework with the competing methods.

Comparisons of JPDAF and TEC, using the CMU dataset, are shown in Fig. 4.16. The *first row* of the figure shows the result of performance for JPDAF on dataset 3D-m compared to TEC using the distance metric given in Eq. 4.25. Here, a misdetection duration $L_m = 5$ is considered. The results presented are averaged over 50 runs. The proposed approach shows a significant improvement of the error distance **D** compared to JPDAF in all types of sequences



(a) frame 80 (b) frame 100 (c) frame 140

Figure 4.15: Sample three-dimensional track results on the CMU 3D dataset for walking (Table 4.1) using JPDAF. The ellipses represent the validation regions of each local tracker for the markers. The number next to each ellipse is the label (index) of the marker.

considered. The obtained error distances are small, since only a small percentage of the overall markers undergoes low-quality tracking performance or tracking failure. The increase in the distance error with frame numbers is due to an increase in the number of failed local tracks in the motion capture sequence. The *second row* of Fig. 4.16 shows the performance of JPDAF and TEC using the number of FP for local tracks on 3*D-m*. The results are shown for variable misdetection length L_m (Table 4.2). The percentage of failures for markers are 19% and 5% for JPDAF and TEC, respectively, when averaged over all types of sequences in 3*D-m*. The result using TEC for the running dataset has not shown much improvement compared to the other sequences, particularly for longer misdetection durations. This particular case is a limitation to our correlation-based correction approach. The reason behind the limitation is due to rapid changes in correlation between marker trackers, which is caused by the rapid change in the dynamics of the markers. Additionally, the proposed correction schema is likely to fail with model points getting spatially close to each other for long time intervals.

For dataset 3*D-c*, the *third row* of Fig. 4.16 shows the performance of JPDAF and TEC using **D** at each frame of the sequence. The performance of tracking markers is improved by using TEC over JPDAF. The result reported is also averaged on 50 runs. The *fourth row* of Fig. 4.16 shows the performance of tracking by counting the number of FP for different amounts of added clutter. The amount of clutter is increased in such a way that it will lead the JPDAF to a likely failure. Averaging over all types of sequences in 3*D-c*, the percentage of failure for markers is

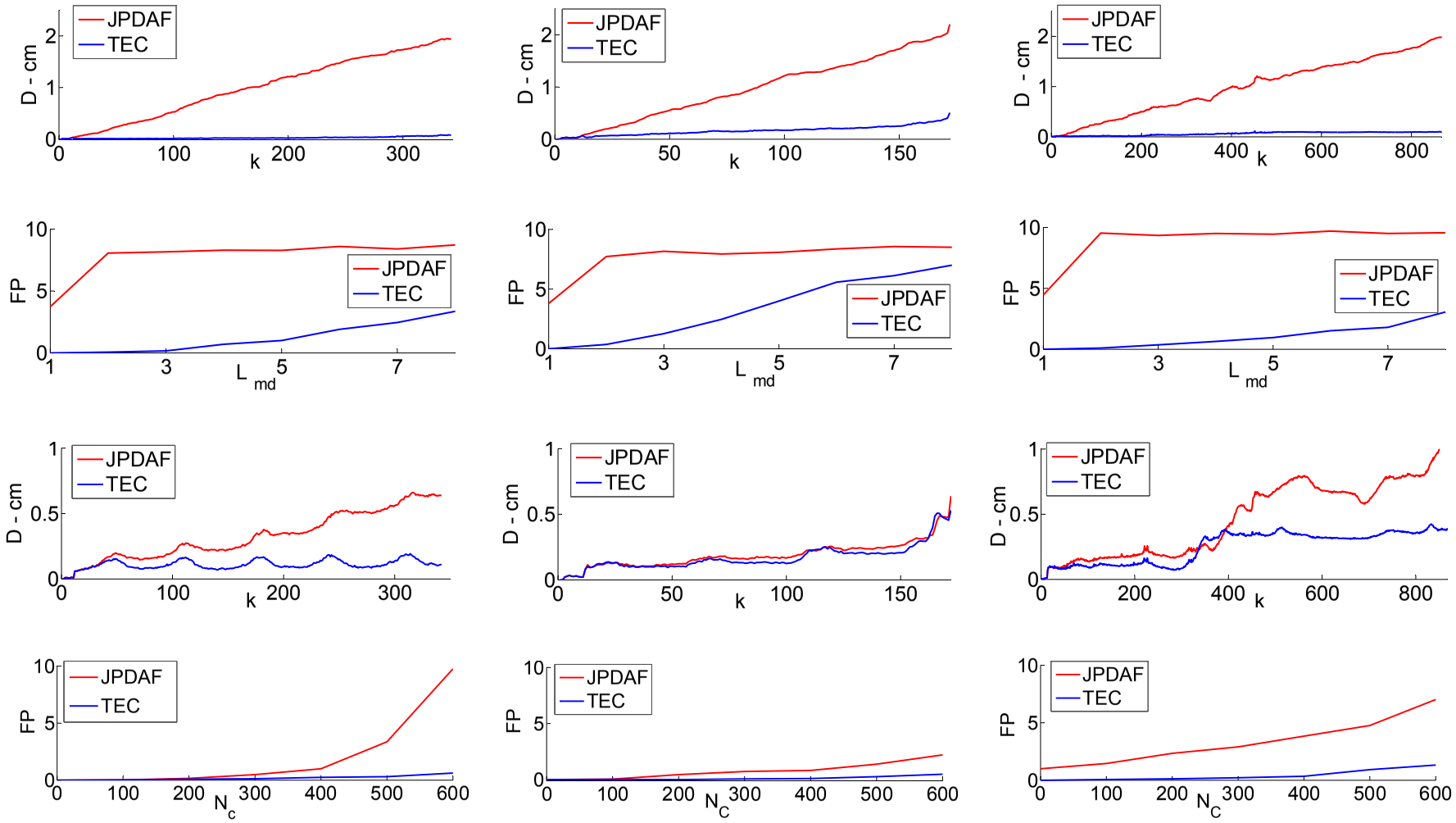
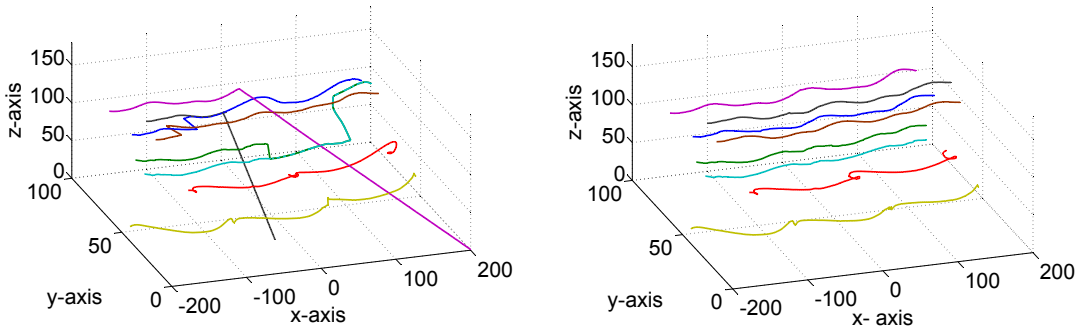


Figure 4.16: Comparisons of JPDAF and TEC on the 3*D* dataset from CMU. The results are averaged on 50 independent runs. *First row:* OSPA (Eq. 4.25) plots for misdetections duration of 5 frames. *Second row*: Number of false positive (*FP*) local tracks with different misdetection durations. *Third row:* OSPA plots for 500 points of added clutter. *Fourth row: FP* local tracks with variable amount of clutter. *First column:* Walking. *Second column:* Running. *Third column:* Dancing.



(a) (b)

Figure 4.17: Visualisation of selected marker trajectories on 3D dataset. (a) JPDAF and (b) TEC.

smaller than 1% for the TEC framework, while for the tracker alone it is 6%. The result shows that the TEC framework improves the tracking result of JPDAF in clutter. Trajectory output plots for JPDAF and TEC framework on the dataset 3*D* are shown in Fig. 4.17. In this figure, results from selected indices are shown to compare the improvements made by TEC over JPDAF. Tracks from JPDAF show failures due to drift caused by the presence of clutter and wrong association with other local trackers.

Similarly using the HumanEva dataset, comparisons of JPDAF and TEC are shown in Fig. 4.18. The improvement made by TEC on the JoggingEva dataset is not as good as that of the WalkingEva dataset due to the rapid changes in correlation between markers (*second row*, *second column*). Averaging over the sequences in the HumanEva dataset, TEC has 12%, while JPDAF has 20.4% of failures. The subjects in the HumanEva dataset make circular motion, unlike Walking and Running in the CMU dataset which are on a straight direction, and cause rapid changes in the correlation model. Failures in the HumanEva dataset are more numerous than those in the CMU dataset.

Figure 4.19 shows the performance of TEC compared to other state-of-the-art trackers using the datasets 2*D*-*m* and 2*D*-*c*. HDAF performs better if the measurements contain either clutter or misdetection. However, when both challenges exist at the same time the performance degrades. NNDAF performs well when there are no misdetections. Unlike JPDAF, HDAF and NNDAF make hard decisions in the association, therefore the presence of clutter and misdetections strongly influence their tracking performance.

As for the offline trackers: PT and SA perform poorly, since misdetection and clutter largely affect the overall optimisation accuracy. In addition to this, offline trackers produce two or more

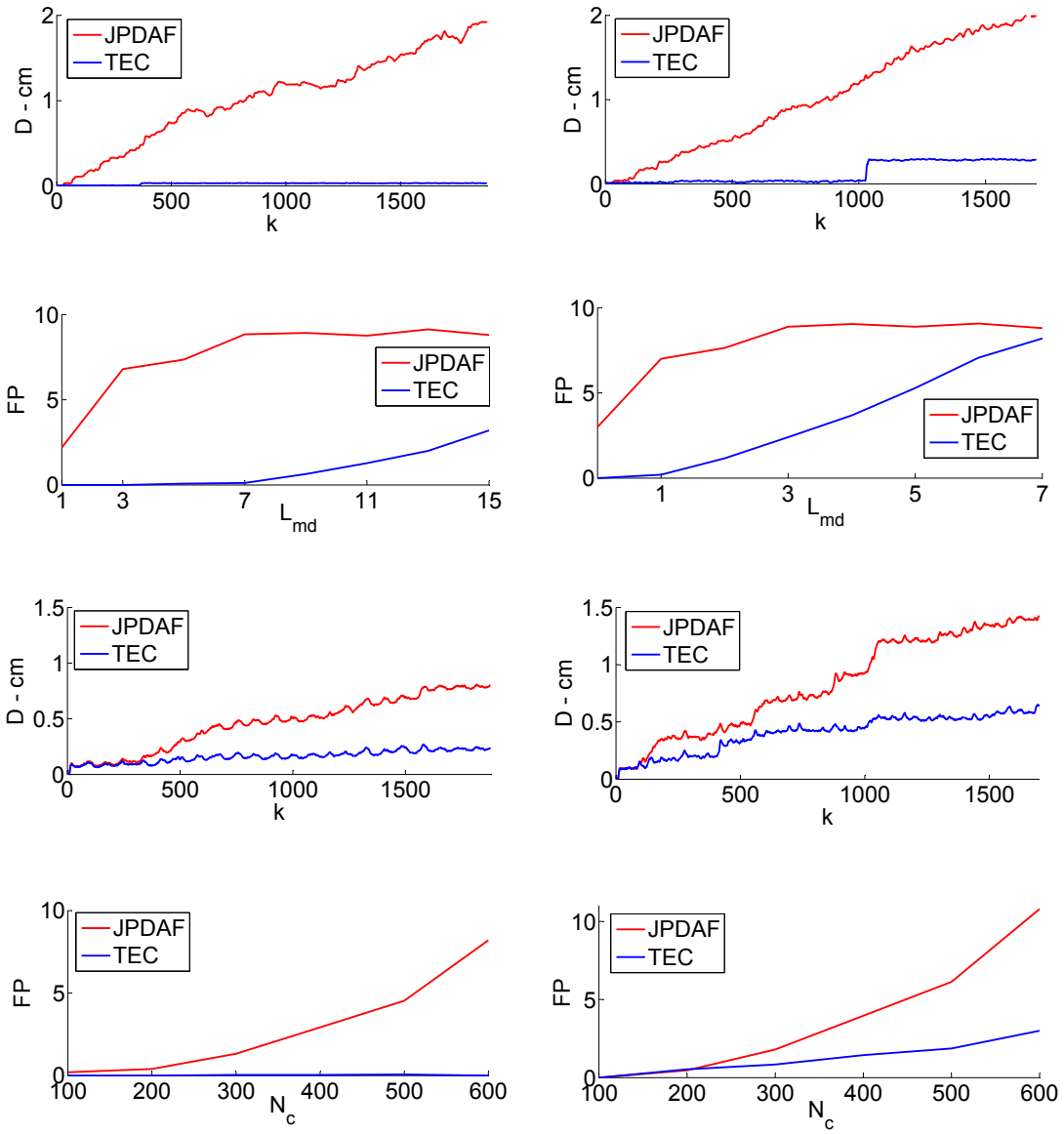
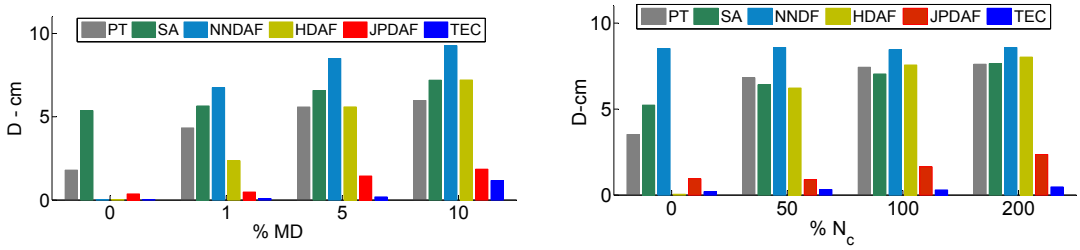


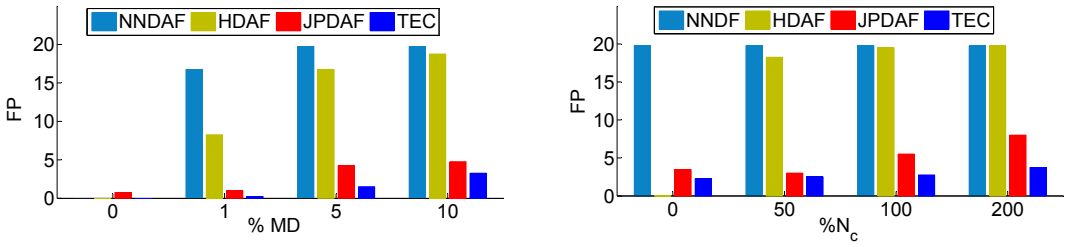
Figure 4.18: Comparisons of JPDAF and TEC using the 3*D* dataset from HumanEva. *First row:* OSPA (Eq. 4.25) plots for a misdetections duration of 5 frames. *Second row*: Number of false positive (FP) local tracks with different misdetection durations. *Third row:* OSPA plots for 500 points of added clutter. *Fourth row:* FP local tracks with variable amount of clutter. *First column:* WalkingEva. *Second column:* JoggingEva.

tracklets for a single marker and result in a higher **D** as compared with online trackers that produce tracks equal to the number of markers. JPDAF performs well with a large amount of clutter and presence of higher misdetections compared to the other trackers. However, using the proposed TEC framework, we have further improved the tracking results from JPDAF. For the dataset 2*D-m* (2*D-m*), TEC improves **D** and FP of JPDAF by 0.64 (1.14) and 1.5 (2.2), respectively. Comparison of online trackers in terms of number of FP is shown in Fig. 4.20. The



(a) (b)

Figure 4.19: Comparison of trackers performance using OSPA averaged over the different human motions. (a) Dataset 2*D-m* with misdetection *MD*; (b) Dataset 2*D-c* with amount of clutter N_c.



(a) (b)

Figure 4.20: Comparison of trackers performance using Number of FP averaged over the different human motions. (a) Dataset 2*D-m* with misdetections *MD*; (b) Dataset 2*D-c* with amount of clutter N_c.

number of FP for offline trackers is not given as the track association with ground-truth is not known a priori.

Table 4.5 compares the results obtained by all the trackers under consideration. The statistics given are averaged over the different types of sequences, the frame lengths and the separate multiple iteration runs. The number of failures and tracking errors in the 2D dataset are larger compared to the 3D dataset. The trajectories of markers in the 2D dataset crossover multiple times, thus increasing the data association problem compared to the 3D dataset. On average over all the the experiments, TEC improves the tracking performance of JPDAF by reducing **D** and FP by 0.66 and 2.87, respectively.

4.7 Summary

In this chapter, we presented a TEC framework to improve tracking performance of an extended object by a set of model points using a Bayesian tracker. The framework uses an on-line performance evaluation based on predefined failure models to decide whether the results obtained

Table 4.5: Comparisons of the results. For the best performing tracker values are indicated in bold and values not calculated are indicated by "-".

Dataset	**Method**	**D**-cm				**Number of FP**
		Average	**Min**	**Max**	**Var**	
3D-m	JPDAF	1.01	0.98	1.04	0.00	7.62
	TEC	**0.08**	0.03	0.17	0.00	**1.83**
3D-c	JPDAF	0.46	0.19	0.83	0.06	2.40
	TEC	**0.22**	0.11	0.40	0.01	**0.40**
2D-m	PT	4.43	0.57	6.52	3.45	-
	SA	6.20	3.75	8.00	1.24	-
	NNDAF	6.14	0.01	9.42	14.5	14.1
	HDAF	3.80	0.01	8.32	9.58	10.9
	JPDAF	1.01	0.14	2.87	0.71	2.7
	TEC	**0.37**	0.00	2.24	0.41	**1.2**
2D-c	PT	6.34	1.27	8.64	4.52	-
	SA	6.58	3.72	8.23	1.62	-
	NNDAF	8.53	8.01	8.83	0.06	19.8
	HDAF	5.46	0.02	8.71	11.6	14.4
	JPDAF	1.46	0.46	2.89	0.71	5.0
	TEC	**0.32**	0.00	0.80	0.10	**2.8**

by each local point tracker are weak or strong. A weak tracker corrects its state in the form of re-initialisation based on the assistance from the available strong trackers. Inferring the corrected state of the weak tracker from that of strong trackers is done by PLS regression using the short windowed trajectories of the trackers. For an accurate estimation of a weak tracker state, a weight is assigned to the estimations from strong trackers based on the observed trajectory correlation level. The level of correlation between local trackers is obtained from the regression model.

We used markers data from CMU [1] and HumanEva [2] motion capture database for experimental analysis and validation. In the data we introduced challenges such as misdetections and clutter as part of the experiment. The proposed framework has achieved improved in performance compared to the baseline tracker [5] used in the framework. Moreover, we have shown the improved in tracking performance by the framework compared to other state-of-the-art trackers [18, 104, 19].

For self-correcting tracking, in addition to identifying the quality of tracks the challenging task involves obtaining the information for correction in case of failures. In this chapter, we utilised the trajectory correlation information between local trackers of the extended target to recover tracking failures. In the case of tracking a single target, the information for correcting

the tracker might be obtained from another tracker or a detector that is able to provide the true state of the same target. Chapter 5 presents TLF as a possible direction to obtain the correction information and allows trackers to have self-correcting capability.

Chapter 5

Tracker-level fusion for robust tracking

5.1 Introduction

In self-correcting target tracking a correction step needs an appropriate source of information such as from another tracker for the same target or for auxiliary targets in the background in order to recover tracking failures. In this chapter, we present Tracker-Level Fusing (TLF) as a method such that the source of information for correction stems from running in parallel independent trackers, and the tracking failure from one tracker is corrected by another tracker [J2]. The proposed TLF framework is shown in Fig. 5.1. The fusion framework is based on the evaluation and correction paradigm (Fig. 1.2). In the framework, we propose a performance measure for trackers to guide the fusion and a tracker collaboration strategy using prior states (prior correction) based on the performance measures of trackers. In addition to this, we set a criterion for the appearance model update that uses the performance measure. Our proposed method is generic for Bayesian trackers: we use two Bayesian trackers with different formulations and show the robustness of the fusion framework using various sequences with challenges such as occlusions, illumination changes, shadows, motion changes and cluttered background. The results are shown to be more accurate compared to results from the individual trackers used in the framework and to state-of-the-art trackers that use tracker-level fusion.

In this chapter, we discuss the performance measure employed for the trackers in Sec. 5.2. The collaboration strategy and estimation of the fused output for the trackers in the framework are presented in Sec. 5.3. Sec. 5.4 explains the trackers used in the framework. Sec. 5.5 describes the

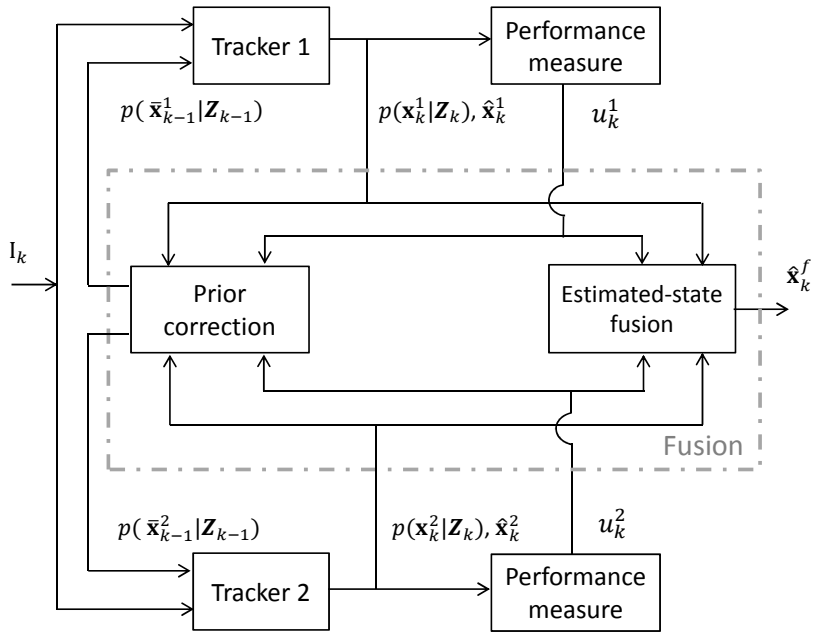


Figure 5.1: Block diagram of the proposed tracker-level fusion framework. A performance measure u^i_k is estimated for the output of each tracker $p(\mathbf{x}^i_k|\mathbf{z}_k)$. u^i_k is used to define the level of interaction between trackers and the fused output. At each frame k, trackers interact using their prior state $p(\mathbf{x}^i_{k-1}|\mathbf{z}_{k-1})$ so as to get the corrected version $p(\bar{\mathbf{x}}^i_{k-1}|\mathbf{z}_{k-1})$ for the next frame state estimation. $\widehat{\mathbf{x}}^f_k$ is the fused state estimate of the target in the framework.

proposed online performance measure based model update and Sec. 5.6 explains the DBN model discrete variables for the framework. Experimental analysis and validation of the framework are presented in Sec. 5.7. Finally, Sec. 5.8 summarises the chapter.

5.2 Performance measure

In this section, we present online quality assessment of trackers. The trackers considered in the fusion framework are based on particle filter (Sec. 5.4). Particle filters are widely-used probabilistic (Bayesian) methods in visual tracking [20]. In the particle filter $p(\mathbf{x}^i_k|\mathbf{Z}_k)$ (Eq. 3.1 and 3.2) consists of a set of particles $\left\{\mathbf{x}^{i,r}_k, \pi^{i,r}_k\right\}_{r=1}^{N_r}$, where $\mathbf{x}^{i,r}_k$ and $\pi^{i,r}_k$ are hypothesised states and their weights, respectively, and N_r is the number of particles [1]. This representation leads to an approximation of $p(\mathbf{x}^i_k|\mathbf{Z}_k)$ using a weighted sum of particles as

[1]When $\mathbf{x}_k$ has two superscripts the first one identifies the tracker, while the second represents the particle index.

$$p(\mathbf{x}_k^i|\mathbf{Z}_k) \simeq \sum_{r=1}^{N_r} \pi_k^{i,r} \delta(\mathbf{x}_k^i - \mathbf{x}_k^{i,r}), \tag{5.1}$$

where $\delta(\mathbf{x})$ is the Dirac delta function. For each particle, the state $\mathbf{x}_k^{i,r}$ is obtained from their previous state $\mathbf{x}_{k-1}^{i,r}$ using the motion model $p(\mathbf{x}_k^{i,r}|\mathbf{x}_{k-1}^{i,r})$, and the weight $\pi_k^{i,r}$ is assigned a value proportional to the observation likelihood $p(\mathbf{z}_k|\mathbf{x}_k^{i,r})$ as [63]

$$\pi_k^{i,r} \propto p(\mathbf{z}_k|\mathbf{x}_k^{i,r}). \tag{5.2}$$

We conduct online tracking quality assessment to quantify trackers' performance based on the spatial uncertainty of the particles [65, 89]. Spatial uncertainty analysis outperforms other methods such as those that directly use likelihood measurement [90]. The spatial uncertainty is calculated using a covariance matrix C_k^i for the particles and their weights [89]. C_k^i is estimated as

$$C_k^i = \sum_{r=1}^{N_r} \widetilde{\pi}_k^{i,r} (\mathbf{x}_k^{i,r} - \check{\mathbf{x}}_k^i)^T (\mathbf{x}_k^{i,r} - \check{\mathbf{x}}_k^i), \tag{5.3}$$

where

$$\widetilde{\pi}_k^{i,r} = \frac{\pi_k^{i,r}}{\sum_{r=1}^{N_r} \pi_k^{i,r}}$$

is the normalised weight of particle i and $\check{\mathbf{x}}_k^i$ is the mean state vector of the particles. The dimensionality of C_k^i is $d \times d$, where d is the dimension of the state vector. The state can have the form of a bounding box, defined as

$$\mathbf{x}_k^i = [x_k^i, y_k^i, h_k^i, w_k^i], \tag{5.4}$$

where (x_k^i, y_k^i) are the centre of the target in the image plane and h_k^i and w_k^i are its approximated width and height, respectively. We use the centre (x_k^i, y_k^i) of the target in the image plane for estimating the spatial uncertainty. The uncertainties of the other state parameter components such as width and height of the bounding box are mainly dependent on the assigned threshold values for the particular target property. The covariance matrix for the centre $\widehat{C}_k^i$ is a 2×2 matrix, and is given as

$$\widehat{C}_k^i = \begin{bmatrix} c_{xx}^i & c_{xy}^i \\ c_{yx}^i & c_{yy}^i \end{bmatrix}, \tag{5.5}$$

where c_{xx}^i and c_{yy}^i are variances of the state components x^i and y^i, respectively, and $c_{xy}^i = c_{yx}^i$ is the covariance between state components x^i and y^i [89]. The spatial uncertainty $\widetilde{u}_k^i$ is estimated from $\widehat{C}_k^i$ as

$$\widetilde{u}_k^i = \frac{\sqrt[2]{|\widehat{C}_k^i|}}{w_k^i h_k^i}, \tag{5.6}$$

where $|\cdot|$ is the determinant of a matrix, w_k^i and h_k^i are the width and height of target size for normalising the uncertainty. For reducing noise in $\widetilde{u}_k^i$, its value is smoothed according to

$$u_k^i = \rho \widetilde{u}_k^i + (1-\rho) u_{k-1}^i, \tag{5.7}$$

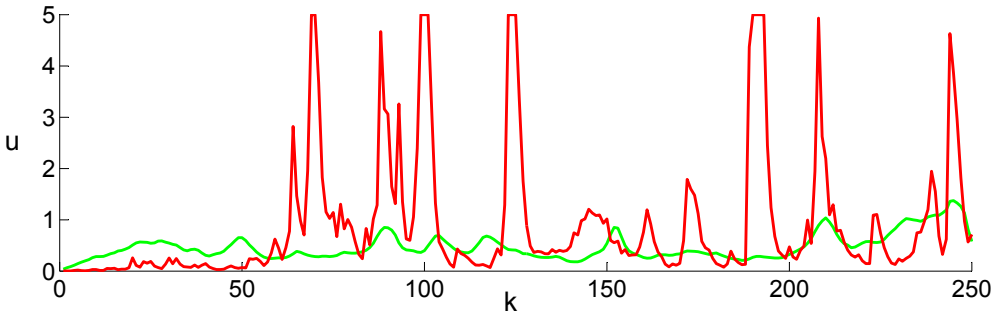
where $\rho \in [0,1]$ is a smoothing factor. The higher the value of ρ, the lower the smoothing effect. The changes of u_k^i over a sliding window can be analysed for classifying whether a tracker is on the target or is lost [89]. We directly utilise u_k^i as track-quality measure for fusion.

When there is a tracking failure (i.e. nearly all samples are far from the target) the weights and particles tend to be more spread compared to when the samples are on the target. This results in higher values of $C_k^i(u_k^i)$. Figure 5.2 shows an example of u_k^i as a quality measure of two trackers, T^1 [76] and T^2 [107]. An illumination change and a partial occlusion happen around $k = 65$ resulting in an increase of u_k for T^2 leading to a tracking failure, while T^1 continues to track well and its u_k remains constant and low.

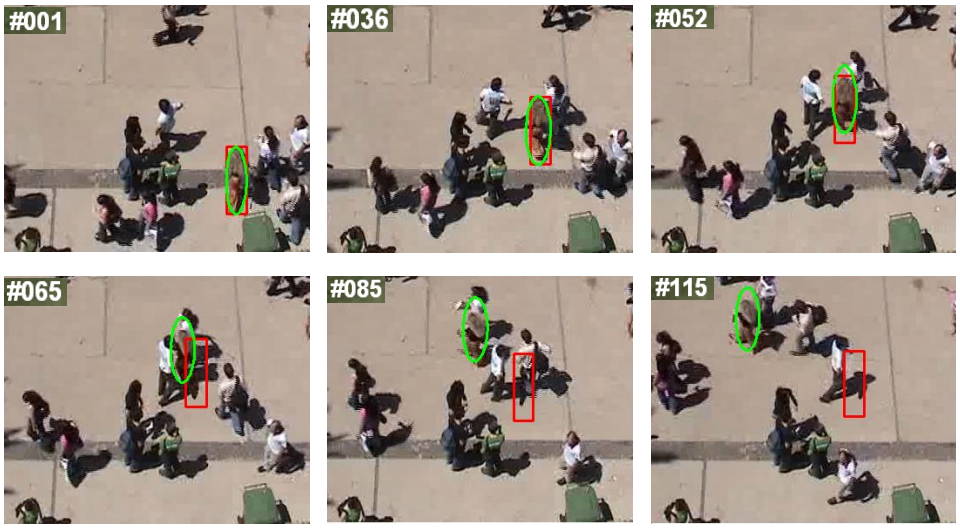
We quantise the performance result, u_k^i, into three quality levels $\mathrm{p}_k^i = \{\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3\}$. The quantisation process to obtain the quality level represents Eq. 3.10. These levels are defined based on two thresholds of the performance measure, namely u_{Th_1} and u_{Th_2}, as

$$\mathrm{p}_k^i = \begin{cases} \mathrm{p}_1 & \text{if} \quad u_k^i \leq u_{Th_1}, \\ \mathrm{p}_2 & \text{if} \quad u_{Th_1} < u_k^i < u_{Th_2}, \\ \mathrm{p}_3 & \text{if} \quad u_k^i \geq u_{Th_1}. \end{cases} \tag{5.8}$$

The first level p_1 has small values of uncertainty that represents a good track performance. The level p_3 has large uncertainty values that indicates low-quality performance. The level p_2 represents an average performance that lies between the two aforementioned levels. We discuss the



(a)



(b)

Figure 5.2: Distribution of the uncertainty measure u over frame k on STUDENTS dataset. (a) Uncertainty measure and (b) sample tracking results. —: T^1 [76] and —:T^2 [107]. Details of T^1 and T^2 are described in Sec. 5.4.

values of u_{Th_1} and u_{Th_2} in Section 5.7.4.

5.3 Fusion

The fusion unit consists of a collaboration strategy between trackers and a method to estimate the fused output from each trackers based on the performance measure (Fig. 5.1). The details of the sub-units are described as follows.

5.3.1 Prior state correction

Poorly performing (failed) trackers usually have an inaccurate (incorrect) prior of the target state that needs to be corrected. In our fusion framework a tracker assists the other by providing an appropriate prior $p(\mathbf{x}^i_{k-1}|\mathbf{Z}_{k-1})$. In the fusion framework, at frame k the corrected prior for a tracker i, $p(\bar{\mathbf{x}}^i_{k-1}|\mathbf{Z}_{k-1})$, described in Eq. 3.14 and Eq. 3.17, is determined as

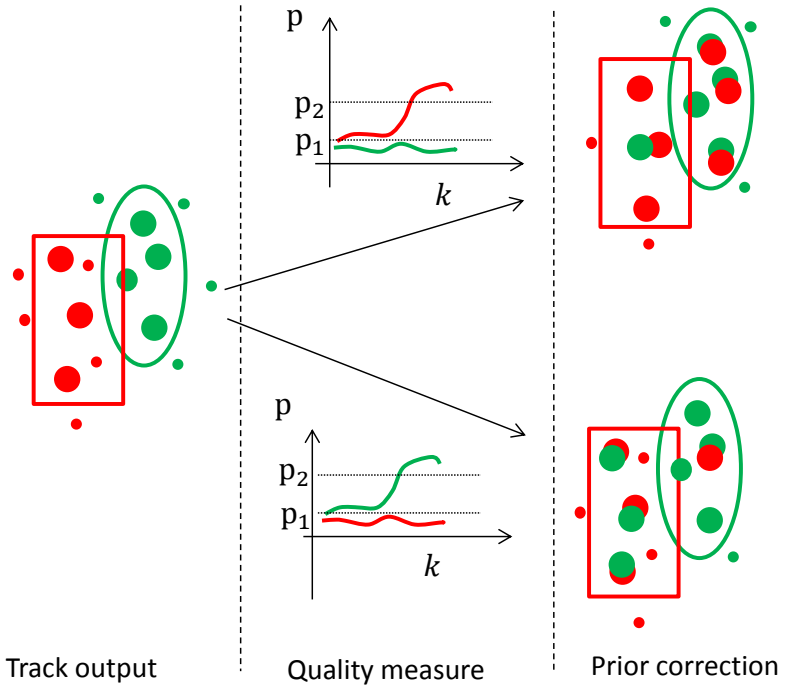


Figure 5.3: Prior state correction across trackers. Two sample cases are shown according to the quality measures obtained. The circles are particles in a $2D$ space and their size is proportional to their weights. Particles with small weights are replaced by particles with large weights. Colours identify the two trackers.

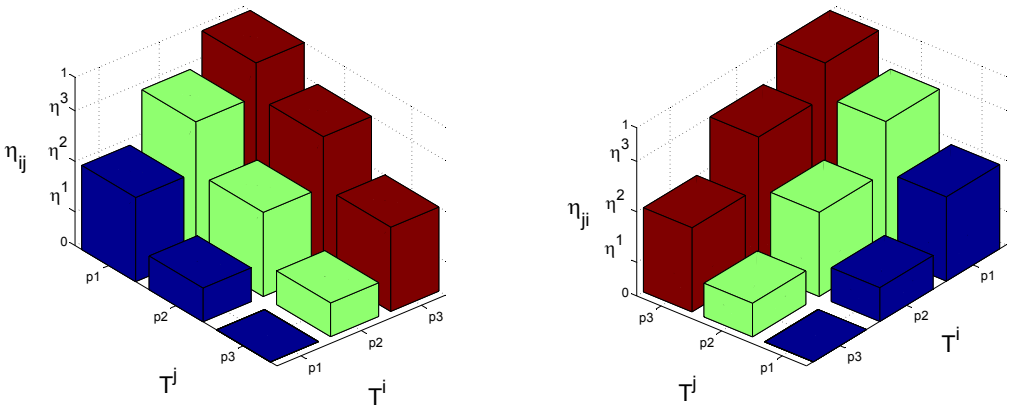
$$\begin{aligned} p(\bar{\mathbf{x}}_{k-1}^{i}|\mathbf{Z}_{k-1}) &= \sum_{j=1}^{M} \Lambda(p(\mathbf{x}_{k-1}^{j}|\mathbf{Z}_{k-1}), \widetilde{\eta}_{ij}), \\ &= \sum_{j=1}^{M} \Lambda(\sum_{r=1}^{N_r} \pi_k^{j,r} \delta(\mathbf{x}_k^j - \mathbf{x}_k^{j,r}), \widetilde{\eta}_{ij}), \end{aligned} \tag{5.9}$$

where M and N_r are the number of trackers and particles, respectively,

$$\widetilde{\eta}_{ij} = \frac{\eta_{ij}}{\sum_{j=1}^{M} \eta_{ij}}.$$

η_{ij} is a weight that determines the level of prior correction for T^i from the prior of T^j and $\widetilde{\eta}_{ij}$ is its normalised weight. $p(\mathbf{x}_{k-1}^{j}|\mathbf{Z}_{k-1})$ represents the prior obtained from each tracker before the fusion. $\Lambda(\cdot,\cdot)$ is a sampling function.

The function $\Lambda(\cdot,\cdot)$ samples the number of particles proportional to $\widetilde{\eta}_{ij}$ from $p(\mathbf{x}_{t-1}^{j}|\mathbf{Z}_{t-1})$ according to the particle weights. For a poorly performing tracker the particles with low weights are replaced with particles with high weights from the well-performing tracker (see Fig. 5.3). During prior correction, T^i only keeps $\widetilde{\eta}_{ii} \cdot N$ particles with good weights, while the remaining



(a) (b)

Figure 5.4: Weight assignment between trackers T^i and T^j according to their performance measure levels. $0 < \eta^1 < \eta^2 < \eta^3 < 1$. (a) η_{ij}: prior state passing from tracker j to tracker i and (b) η_{ji}: prior state passing from tracker i to tracker j.

particles $(1-\widetilde{\eta}_{ii}) \cdot N$ are taken from the other trackers. For trackers with different representations of the state vector $\mathbf{x}_k$ (e.g. rectangular and elliptic), the calculation of the corrected prior (Eq. 5.9) requires a transformation between the states.

Figure 5.4 shows the estimation of η_{ij} between trackers for level of interaction in their prior. $\eta_{ij} \in \{0, \eta^1, \eta^2, \eta^3, 1\}$ is dependent on the trackers respective performance levels p^i and p^j, which in turn depends on u_i and u_j. η^1, η^2 and η^3 are constants such that $0 < \eta^1 < \eta^2 < \eta^3 < 1$. Considering T^i and T^j with $\mathrm{p}^i = \mathrm{p}_3$ and $\mathrm{p}^j = \mathrm{p}_1$, respectively, we get $\eta_{ij} = 1$ and $\eta_{ji} = 0$. This case results T^i to leap (complete re-initialisation) to the state of T^j (Fig. 5.4). Similarly, considering the case T^i with $\mathrm{p}^i = \mathrm{p}_2$ and T^j with $\mathrm{p}^j = \mathrm{p}_3$, we get $\eta_{ij} = \eta^1$ and $\eta_{ji} = \eta^3$. In this case since $\eta^1 < \eta^3$, T^i will take more prior of T^j and T^j takes only small part of the prior from T^i as collaboration. Although prior correction might not be necessary when all the available trackers perform well ($\mathrm{p}^i = \mathrm{p}^i = \mathrm{p}_1$), we prefer trackers to interact in order to avoid possible tracking failures such as local minimum state estimation due to clutter or similar background to the target. As a particular case, prior mixing with good performance conditions is prominent for trackers using MAP estimate strategies.

Note that a poorly performing tracker can degrade the performance of a well-performing tracker during prior interaction. In our approach the constraint $0 < \eta^1 < \eta^2 < \eta^3 < 1$ for the weights minimises the effect of a poorly performing tracker on a well-performing tracker and maximises the performance improvement of the poorly performing tracker. Although the weight

of $\eta^1 \approx 0$ could avoid this effect, this solution would be completely dependent on the accuracy of the online performance measure. We discuss the values of the weights in Section 5.7.4.

The correction of a tracker in the form of prior can be compared to a resampling technique in a particle filter. In the case of a single particle filter, the resampling is constrained to its own particles only. In the proposed fusion framework the prior correction of a tracker can be considered as a similar strategy of resampling particles between trackers thus achieving robust tracking in an assistive manner.

5.3.2 Estimated-state fusion

The final target state $\widehat{\mathbf{x}}_k^f$ from the fusion framework is obtained as a weighted sum of the states estimated by each tracker. The weights for the fusion are estimated similarly to the prior state correction. $\widehat{\mathbf{x}}_k^f$ is calculated as

$$\widehat{\mathbf{x}}_k^f = \sum_{i=1}^{M} \widetilde{\eta}_{ii} \widehat{\mathbf{x}}_k^i, \tag{5.10}$$

where $\widehat{\mathbf{x}}_k^i$ is the state estimate from each tracker in the fusion framework and the weight $\widetilde{\eta}_{ii} = \left(1 - \sum_{j=1, i\neq j}^{M} \widetilde{\eta}_{ij}\right)$ measures the tracking quality level of each tracker. Similarly to the idea presented in Eq. 5.9, $\widetilde{\eta}_{ii}$ determines the number of particles a tracker keeps while correcting its prior state. Similar to the prior state correction, in the case of differences in state representation between trackers, the transformation of the states to a similar representation is required in order to apply the weighted-sum rule (Eq. 5.10).

5.3.3 Computational complexity

The fusion framework with M trackers and N particles for each trackers has an upper-bound computational complexity of $\mathrm{O}(M^2N)$. Specifically, the complexity of the fusion framework can be divided into the complexity of the performance measure, prior correction and fusing output operations for the individual trackers. The performance measure (Eq. 5.3-5.7) involves summation of weighted states of particles and has a complexity of $\mathrm{O}(N)$. The prior correction for a tracker involves state exchange in the form of particles with other trackers based on minimum and maximum weights selection (Eq. 5.9). This operation has an upper-bound complexity of $\mathrm{O}(MN)$. For M trackers, the performance measure and prior correction have a complexity of $\mathrm{O}(MN)$ and

$\mathrm{O}(M^2N)$, respectively. The fusion operation (Eq. 5.10) multiplies the estimated state from each tracker and has a complexity of $\mathrm{O}(M)$.

5.4 Visual trackers

As example of application for the proposed framework, we consider two particle filters with complementary appearance and motion models. The first tracker is based on colour histograms and a constant velocity motion model [76], whereas the second tracker is based on sparse features with a Gaussian random walk motion model [107]. The details of the trackers are given below.

5.4.1 Colour-Histogram-based Particle Filter (CHPF)

Colour features encoded in a histogram are robust to rotations, deformations and non-rigidity of the target. Let the set $\left\{\mathbf{x}_k^{1,r}, \pi_k^{1,r}\right\}_{r=1}^{N_r^1}$ be the states of the N^1 particles and their weights for $CHPF$. The state of each particle $\mathbf{x}^1$ (indexes k and s are removed for simplicity) is represented as an ellipse with parameters $\mathbf{x}^1 = [x, \dot{x}, y, \dot{y}, h_x, h_y, \theta]$, where (x, y) represents the centre, $(\dot{x}, \dot{y})$ is the velocity of the centre, (h_x, h_y) represent the half-length of the major and minor axes, respectively, and θ is the orientation for the ellipse.

The probability distribution for the state transition $p(\mathbf{x}_k^1|\mathbf{x}_{k-1}^1)$ is obtained from a constant velocity motion model with Gaussian random noise. The state transition is formulated as

$$\mathbf{x}_k^1 = A\mathbf{x}_{k-1}^1 + W_k, \tag{5.11}$$

where A is a matrix that defines the constant velocity model from the state components $\dot{x}$ and $\dot{y}$ [128] and W_k is the Gaussian noise for the state transition model.

The weight π^1 is estimated through the likelihood function defined from the observed colour histogram at $\mathbf{x}^1$ and the target's reference colour histogram. For measuring the similarity between colour histograms, we use the Bhattacharyya distance $d_B = \sqrt{1 - \sum_{u=1}^{m} \sqrt{p_u q_u}}$, where p_u and q_u are colour histograms with $m-$bin for the candidate sample and the reference model, respectively. The weights of each particle are calculated from d_B as

$$\pi^1 \propto exp^{-(\frac{d_B}{\varepsilon})^2}, \tag{5.12}$$

where ε is a constant.

In *CHPF* the Minimum Mean Square (MMS) formulation is used to estimate the final output for representing the target state $\widehat{\mathbf{x}}_k^1$ from $p(\mathbf{x}_k^1|\mathbf{Z}_k)$. $\widehat{\mathbf{x}}_k^1$ is estimated as

$$\widehat{\mathbf{x}}_k^1 = \sum_{r=1}^{N_r^1} \mathbf{x}_k^{1,r} \widetilde{\pi}_k^{1,r}, \tag{5.13}$$

where $\widetilde{\pi}_k^{1,r}$ is the normalised particle weight (Eq. 5.3).

5.4.2 Least Soft-thresold Squares (LSS) tracker

LSS uses sparse features obtained from intensity values of a target template. The template is resized to $T_s \times T_s$ pixels for extracting the features. Representations based on sparse features have achieved success in tracking [111, 113, 58], classification and recognition [122, 114]. Let the set $\left\{\mathbf{x}_k^{2,r}, \pi_k^{2,r}\right\}_{r=1}^{N_r^2}$ includes N_r^2 particles and their weights for *LSS*. The state $\mathbf{x}^2$ of each particle is represented by a rectangle with affine parameters $\mathbf{x}^2 = [x, y, s_x, \theta, r_x, \delta]$, where (x, y) is the centre, s_x is the scale, θ is the rotation, r_x is the aspect ratio and δ is the skew angle of the rectangle.

The state transition $p(\mathbf{x}_k^2|\mathbf{x}_{k-1}^2)$ for the particles is modelled as a random walk

$$\mathbf{x}_k^2 = \mathbf{x}_{k-1}^2 + \Omega, \tag{5.14}$$

where Ω is a matrix representing the Gaussian random walk composed of the standard deviations of each affine model parameter.

The weights π^2 are assigned using the Least Soft-thresold Square distance, d_{LSS}, which is obtained by minimising the error with a linear regression between the template model and the observation $\mathbf{z}_k$ from I_k. d_{LSS} is obtained by assuming a Laplacian distribution for the error term in order to reduce the effect of outliers. A linear regression for observation $\mathbf{z}$ (the index k is removed for simplicity) with two independent noise components, Gaussian and Laplacian, is given as

$$\mathbf{z} = \grave{\mathbf{D}}\mathbf{y} + \omega_G + \omega_L, \tag{5.15}$$

where $\mathbf{y}$ is the vector of coefficients to be estimated from the regression; $\grave{\mathbf{D}}$ is the known dictionary or basic matrix of the target; ω_G and ω_L are the Gaussian and Laplacian noise vectors. d_{LSS} is defined as the distance between $\mathbf{z}$ and $\grave{\mathbf{D}}$ as

$$d_{LSS}(\mathbf{z}, \grave{\mathbf{D}}) = \min_{\{\mathbf{y}, \omega_L\}} \frac{1}{2}||\mathbf{z} - \grave{\mathbf{D}}\mathbf{y} - \omega_L||_2^2 + \lambda||\omega_L||_1, \tag{5.16}$$

where λ is a regularisation constant. The weight π^2 for the particle is assigned according to d_{LSS} as

$$\pi^2 \propto exp^{(-d_{LSS})}. \tag{5.17}$$

The Maximum a Posteriori (MAP) formulation is used to estimate the final state of the target $\widehat{\mathbf{x}}_k^2$ from $p(\mathbf{x}_k|\mathbf{Z}_k)$. $\widehat{\mathbf{x}}_k^2$ is assigned to the state of the particle with maximum π^2 as

$$\widehat{\mathbf{x}}_k^2 = arg \max_{\pi^{2,r}} [\mathbf{x}_k^{2,r}, \pi_k^{2,r}]_{r=1}^{N_r^2}. \tag{5.18}$$

Note that the two selected trackers have limitations related to using a single motion model in the case of variation of motion dynamics. In *CHPF*, the constant velocity motion model may result in drifting in the cases of a sudden motion change. The colour histogram used in *CHPF* is sensitive to clutter. *LSS* uses a Gaussian motion model and maximum a posteriori formulations, which cannot explicitly cope with occlusions. In addition to this, *LSS* calculates intensity features in the gray image only: this could be improved using colour information. In the proposed fusion framework, effective collaborative tracking is enabled and the limitations from each tracker are minimised.

5.5 Appearance model update

Updating the target appearance model using each output $\widehat{\mathbf{x}}_k$ may lead to drifting. Either an assumption of the estimated state $\widehat{\mathbf{x}}_k$ for the tracker [86, 44] or a separate external labeler to select $\widehat{\mathbf{x}}_k$ [46, 55] are usually used for updating the appearance model [88]. We propose a selective update for the appearance model by observing the tracking performance level. The current state $\widehat{\mathbf{x}}_k$ is considered for updating the current model only when a tracker is at level p_1 and p_2, because when the performance of a tracker is at level p_3 the tracker may be tracking the wrong target and will result in an incorrect model update and subsequent tracking failure.

For *CHPF* the update is performed according to

$$q(k) = \alpha q_0 + (1 - \alpha) p_{MMS}(k), \tag{5.19}$$

	$[p^i, p^j]$								
	$[p_1, p_1]$	$[p_1, p_2]$	$[p_1, p_3]$	$[p_2, p_1]$	$[p_2, p_2]$	$[p_2, p_3]$	$[p_3, p_1]$	$[p_3, p_2]$	$[p_3, p_3]$
c^i	2	1	0	5	4	3	8	7	6
c^j	2	5	8	1	4	7	0	3	6

Figure 5.5: Mapping function f_c^i (Table 3.1) to obtain the values of $\mathbf{c}_k^i$ for tracker i from the value of its $\mathbf{p}_k^i$ and the value of $\mathbf{p}_k^j$ from tracker j.

where $p_{MMS}(k)$ is the target representation obtained from the track estimate $\widehat{\mathbf{x}}_k$ and α measures the weight contribution of p_{MMS} as the target model together with the original model, q_0. The authors of *LSS* [107] have proposed an update based on the state estimates of the tracker every 5 frames. In our fusion framework, a similar update strategy is followed, but only using tracks with performance levels of p_1 and p_2.

5.6 Discrete variables of the DBN model

The performance levels p_1, p_2 and p_3 represent the discrete values $\mathbf{p}_k$. For the two Bayesian trackers are considered, 9 different correction variable values $\mathbf{c}_k^i = \{0, 1, \cdots, 8\}$ exist. The values of $\mathbf{c}_k^i$ are obtained from a combination of $\mathbf{e}_k^i$ and $\mathbf{e}_k^j$ values for trackers using the mapping function f_c^i in Eq. 3.19 (Fig. 5.5). The discrete value $\mathbf{c}_k^i = 0$ represents no prior correction on the tracker, while the other values are for different levels of prior correction according to the weight η_{ij} (Fig. 5.4).

Performance of two trackers in the tracker-level fusion is compared to the individual use of trackers in Fig. 5.6. The quality of tracks are measured using overlap score (Sec. 5.7.3). The result shows that trackers in the fusion framework has improved tracking performances compared to the independent use of trackers. In the figure the different levels of prior correction are shown through the sequence. $c = 2$ represents when both trackers are in good performance and with equal amount of prior exchange, $c = 8$ represents complete re-initialisation of a tracker from the state of the other tracker. The case $c = 0$ represents when no prior correction is done on the tracker. In most of the sequence the $c = 2$ indicating both trackers are performing good, however at $k = 125, 126, 127$, *CHPF* is completely re-initialised from the prior of *LSS* ($c = 8$). The re-initialisation allow *CHPF* to resume good tracking performance. At the re-initialisation of *CHPF*, *LSS* takes no prior from the other tracker, i.e. $c = 0$. *LSS* fails at $k = 80$, while in the fusion framework the tracker is able to avoid the failure.

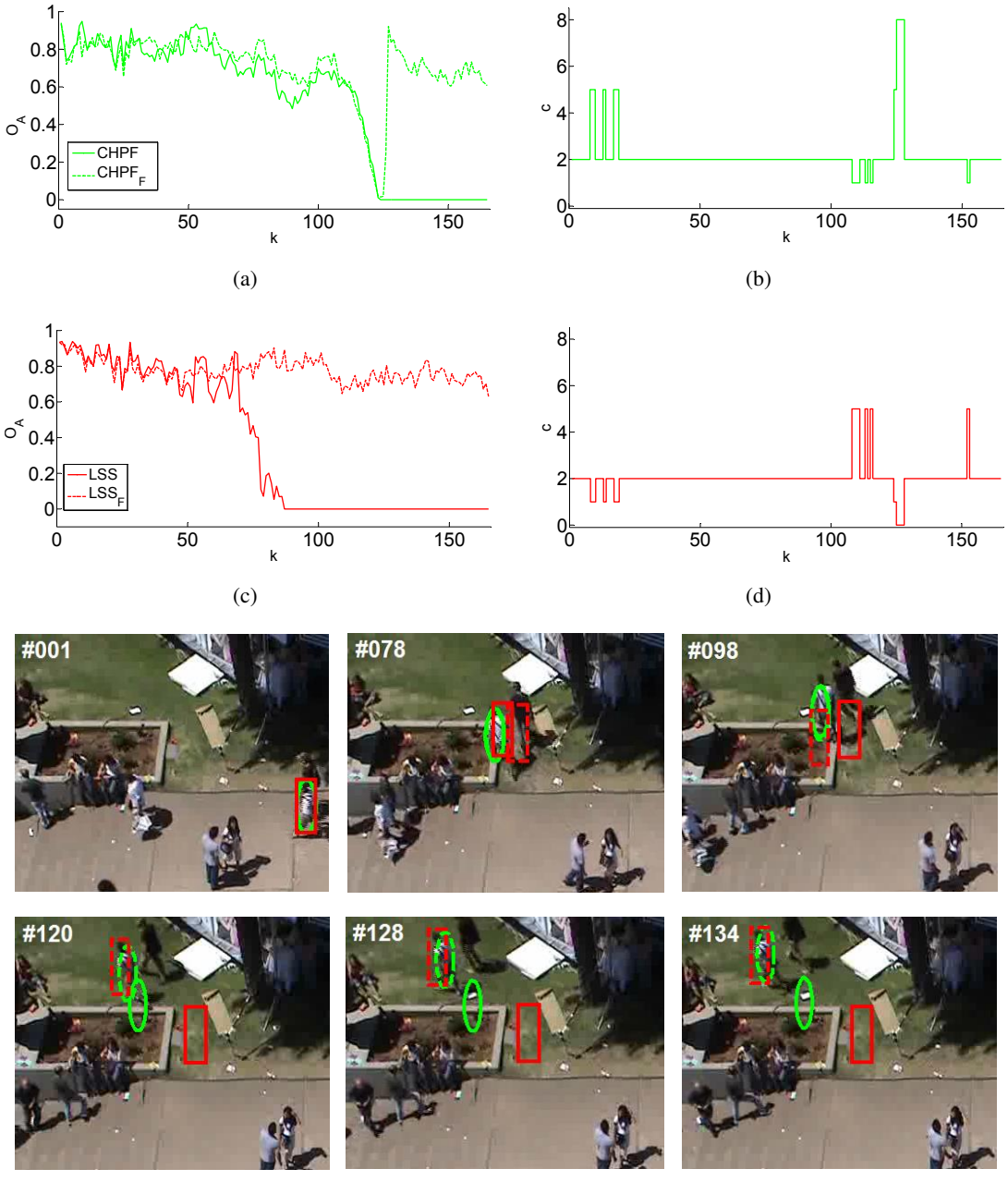


Figure 5.6: Comparison of trackers working independently and collaboratively in the fusion framework. (a) and (b) *CHPF*, and (c) and (d) *LSS*. (a) and (c) Area overlap score, O_A, between track estimate and ground-truth data (Sec. 5.7.3), and (b) and (d) values of c with $p(\mathbf{c}_k = c) = 1$. (e) Sample tracking results for —: *CHPF*, —: *LSS*, - - -: $CHPF_F$ (*CHPF* collaborating with *LSS* in the fusion framework) and - - -: LSS_F (*LSS* collaborating with *CHPF* in the fusion framework.

5.7 Experimental results and analysis

5.7.1 Experimental setup

For testing the proposed approach, we consider the STUDENTS[2] [56], CAVIAR[3], MEN100m[4,5], VISOR[6], PETS2006[7], SINGER1, SINGER2, SKATING2, BASKETBALL[8], DAVIDOUTDOOR [107], PANDA[9], MOTOR1[10], MOTOR2[11] and CHASING[12] datasets. The targets in these datasets undergo motion blur, occlusion, different motion dynamics, illumination and background changes. The datasets have 19 targets and a total of 8482 frames in which the targets exist. Table 5.1 summarises the properties of the datasets used in the experiments. Figure 5.7 shows the target initialisations.

We compare the output of the fusion framework F (Eq. 5.10) with that of individual trackers $CHPF$ and LSS, and 7 state-of-the-art trackers. CHPF and LSS, and their fusion F are implemented (non-optimised code) using MATLAB2013b, which run 45, 6 and 5 frames per second, respectively, on an intel(R) core(TM) i5-3570k, 3.4GHz CPU with 8GB RAM on windows 7. We comparison with the VTS tracker [52] [53], which uses tracker-level fusion. VTS uses 2 motion models and 4 features as appearance models to construct 8 separate trackers. From the separate trackers, a fused state is estimated according to their likelihood values. We also compare with AFT [27], which uses an adaptive cue integration for part-based tracking. Additionally, we consider five state-of-the-art trackers $L1T$ [68], MTT [131], FCT [130], CT [129] and $3DDCT$ [57] for comparison. The performance of $CHPF$ and LSS change due to their collaboration when running in the fusion framework. In order to analyse the change, we also compare the performance of $CHPF_F$ and LSS_F, where $CHPF_F$ and LSS_F are output of $CHPF$ and LSS, respectively, while collaborating in the fusion framework.

Table 5.1: Summary of the datasets used in the experiments. key- $S_1, S_2, S_3, R_1, R_2, B_1$ and B_2: particular targets in the datasets, K: number of frames, O: occlusion, BC: background clutter, MV: motion variation, IC: illumination changes, SC: scale changes, AC: strong articulated structure changes, MIN: minimum and MAX: maximum

Dataset		K	**Image size**	**Target size**		**Characteristics**
				MIN	MAX	
STUDENTS	S_1	250	720 × 576	26 × 58	26 × 58	O, BC, MV, IC, AC
	S_2	250		23 × 65	23 × 65	
	S_3	165		24 × 64	24 × 64	
CAVIAR	R_1	500	384 × 288	24 × 62	52 × 136	SC, O, BC, IC
	R_2	500		24 × 62	52 × 136	
MEN100M	B_1	500	640 × 360	22 × 61	26 × 74	MV, BC, AC
	B_2	293	480 × 270	33 × 50	35 × 75	
VISOR		300	288 × 352	11 × 18	18 × 37	O, MV, SC
PETS2006		247	720 × 576	34 × 105	49× 158	IC, SC
SINGER1		321	624 × 352	26 × 79	70 × 270	IC, BC, SC
SINGER2		366	624 × 352	64 × 142	84 × 264	IC, BC, MV, AC
SINGER2^{face}		366	624 × 352	25 × 28	34 × 40	IC, BC, MV
SKATING2		707	640 × 352	40 × 130	70 × 200	O, BC, MV
DAVIDOUTDOOR		252	640 × 480	34 × 140	40 × 154	O, MV
BASKETBALL		725	576 × 432	33 × 80	33 ×80	O, BC, MV
PANDA		970	312 × 233	22 × 14	40 × 30	BC, IC, MV, SC
MOTOR1		800	1280 × 720	39 × 62	83 × 166	MV, IC, SC, BC
MOTOR2		370	1280 × 720	52 × 132	85 × 172	MV, SC, BC
CHASING		600	320 × 240	20 × 22	58 × 47	MV, IC, SC, BC

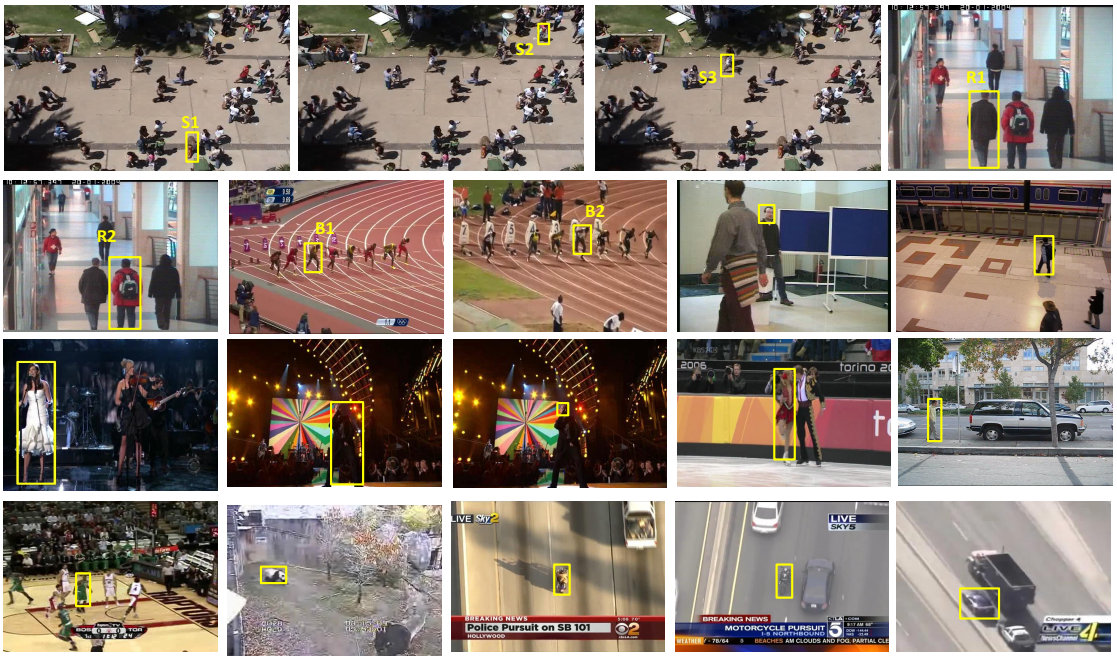


Figure 5.7: Targets selected and initialisation in each dataset. The order from top-left to bottom-right along the rows is according to the order of the rows in Table 5.1.

Table 5.2: Summary of parameters used for the individual trackers and the fusion framework.

Parameter name	Symbol	Value	Reference
Number of particles	N_r	≤ 600	Eq. 5.1
Smoothing coefficient for u_k	ρ	0.5	Eq. 5.7
Spatial uncertainty thresholds	u_{Th_1}	1	Eq. 5.8
	u_{Th_2}	2	
Appearance model update coefficient	α	0.7	Eq. 5.19
Interaction weights	η^1	0.25	Fig. 5.4
	η^2	0.5	
	η^3	0.75	
LSS tracker template size	T_s	32	Sec. 5.4.2
LSS tracker regularisation constant	λ	0.1	Eq. 5.16

5.7.2 Parameters

The thresholds for performance measure levels are set to 1 and 2 for u_{Th_1} and u_{Th_2}, respectively, for all datasets, except for DAVIDOUTDOOR. In this sequence, due to clutter with similar colour distribution, so 0.5 and 1 are used for u_{Th_1} and u_{Th_2}, respectively. The interaction weights between trackers for prior mixing η^1, η^2 and η^3 are selected as 0.25, 0.5 and 0.75, respectively. The parameter values for the fusion are selected based on a parameter sensitivity analysis (Section 5.7.4). For the thresholds of performance measure levels, a similar value of threshold u_{Th_2} is used in [89] for tracker quality level segmentation. The smoothing coefficient for u_k is $\rho = 0.5$. For *CHPF* the colour histogram is constructed using $8 \times 8 \times 8$ bins in RGB colour space and its model update coefficient is $\alpha = 0.7$. For the *LSS* tracker, we use the original implementation from the authors with $T_s = 32$, $\lambda = 0.1$ and 16 for the eigenvector representation[13]. For all trackers the same number of particles is used in a particular dataset. The number of particles used in a particular dataset is a maximum of 600 or less. In our implementation, we choose the state representation of $\widehat{\mathbf{x}}_k^f$ to be similar to one used in the *LSS* tracker. For the *VTS* and 3*DDCT*,

[2]http://graphics.cs.ucy.ac.cy/research/downloads/crowd-data
[3]http://groups.inf.ed.ac.uk/vision/ CAVIAR/CAVIARDATA1/
[4]https://www.youtube.com/watch?v=2O7K-8G2nwU
[5]http://www4.comp.polyu.edu.hk/∼cslzhang/CT/CT.htm
[6]http://imagelab.ing.unimore.it/visor/
[7]http://www.cvg.rdg.ac.uk/PETS2006/data.html
[8]http://cv.snu.ac.kr/research/vtdvts/
[9]http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/index.html
[10]http://www.youtube.com/watch?v=3PoMeL1mCak
[11]http://www.youtube.com/watch?v=cYq1esN_yJ4
[12]http://www4.comp.polyu.edu.hk/~cslzhang/FCT/FCT.htm
[13]code:http://faculty.ucmerced.edu/mhyang/pubs.html

the common parameters such as number of particles and motion parameters for the targets in the datasets, are set to the same value as for *LSS* and *CHPF*. The other parameters of *VTS*, such as the bin size of the colour and edge histograms, and the number of frames used to generate the target model, are set to their default values as provided in the authors' code[14]. Similar strategy is followed for *L1T*[15], *MTT*[16], *FCT*[17], *3DDCT*[18] and *AFT*[19]. A similar tracking parameters with the best performing colour component of RGB images are used for *CT*[20]. The parameters we use for the trackers are optimal for each dataset (i.e the best results for the trackers).

5.7.3 Evaluation measures

In order to compare the performance of the trackers, we use the overlap score O_A and the area under the lost-track-ratio curve AUC_κ [72], [64]. O_A measures the area of overlap between the bounding boxes generated by the tracker estimate and the ground truth. O_A is calculated as

$$O_A = \frac{2|A_o \cap A_g|}{|A_o| + |A_g|}, \tag{5.20}$$

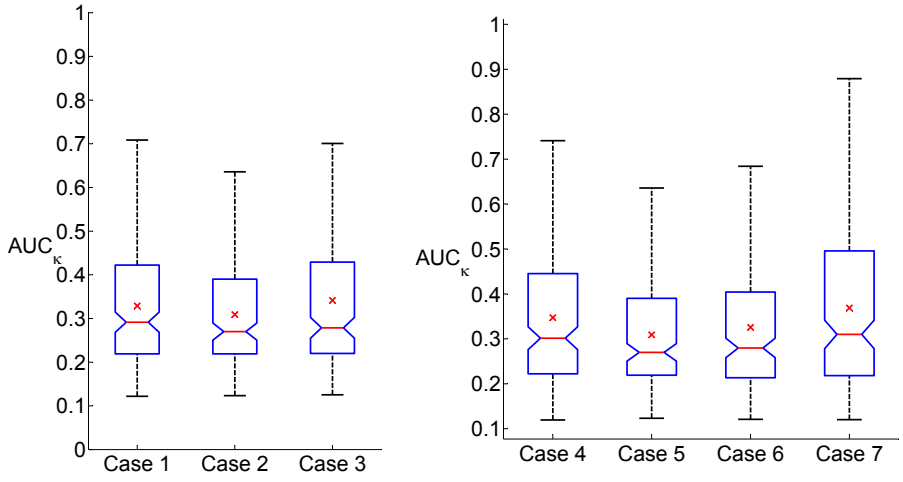
where A_o and A_g are the bounding boxes for the tracker estimate and ground truth, respectively; and $\cap$ and $|\cdot|$ denote the intersection and area estimation operators on the bound boxes, respectively. $O_A \in [0,1]$ and the larger its value, the better the tracking result. For *CHPF*, the ellipse output is converted into a rectangular one for calculating its O_A with the rectangular ground truth. In order to quantify a track failure, the lost-track-ratio κ is estimated as [64]:

$$\kappa = \frac{N_k}{N_f}, \tag{5.21}$$

where N_f is the total number frames where the target exists and N_κ is the number of frames with $O_A < \tau$, $\tau \in [0,1]$. In order to remove dependency on τ, the AUC_κ is estimated as

$$AUC_\kappa = \Delta\tau \sum_{\tau=0}^{1} \kappa(\tau), \tag{5.22}$$

[14]code: http://cv.snu.ac.kr/research/∼vts/
[15]code: http://www.dabi.temple.edu/∼hbling/code_data.htm
[16]code: https://sites.google.com/site/zhangtianzhu2012/
[17]code: http://www4.comp.polyu.edu.hk/∼cslzhang/FCT/FCT.htm
[18]code: https://code.google.com/p/boosting/downloads/list
[19]code: http://web.cs.hacettepe.edu.tr/∼erkut/publications.html
[20]code:http://www4.comp.polyu.edu.hk/∼cslzhang/CT/CT.htm



(a) (b)

Figure 5.8: Sensitivity analysis of the fusion parameters. (a) Performance with different magnitudes of interaction weights, Case 1: $\eta^1 = 0.10, \eta^2 = 0.50, \eta^3 = 0.90$; Case 2: $\eta^1 = 0.25, \eta^2 = 0.50, \eta^3 = 0.75$; Case 3: $\eta^1 = 0.40, \eta^2 = 0.50, \eta^3 = 0.60$. u_{Th} is selected as of Case 5 mentioned below. In the box plots the mean AUC_κ values, indicated by mark '**x**', for Cases 1, 2 and 3 are 0.33, 0.31 and 0.34, respectively. (b) Performance with different threshold levels of the spatial uncertainty, Case 4: $u_{Th_1} = 0.5, u_{Th_2} = 1$; Case 5: $u_{Th_1} = 1, u_{Th_2} = 2$; Case 6: $u_{Th_1} = 3, u_{Th_2} = 4$; Case 7: $u_{Th_1} = u_{Th_2} = \infty$. For η_{ij} we use Case 2. In the box plots the mean AUC_κ values, indicated by mark '**x**', for Cases 4, 5, 6 and 7 are 0.35, 0.31, 0.33 and 0.37, respectively.

where $\Delta\tau$ is a small incremental value used to count the possible variations in τ. In in our experiment we set $\Delta\tau = 0.01$. $AUC_\kappa \in [0,1]$ and the lower its value, the better the tracking result.

5.7.4 Discussion

Figure 5.8 shows the results of the fusion framework to parameter variations. The results by varying the weight η_{ij} (assigned from η^1, η^2 and η^3) with $u_{Th_1} = 1$ and $u_{Th_2} = 2$ are shown in Fig. 5.8(a). In the box plots, the AUC_κ results are obtained by averaging the results from all the datasets. Since η^2 represents tracking conditions with similar performance levels, we keep its value constant as $\eta^2 = 0.5$, while three different values of η^1 and η^3 are considered, with the assumption that $\eta^1 < \eta^2 < \eta^3$. From the box plots we observe similar performance between the three cases. However *Case 2* with $\eta^1 = 0.25, \eta^2 = 0.5$ and $\eta^3 = 0.75$ outperforms *Case 1* with $\eta^1 = 0.1, \eta^2 = 0.5$ and $\eta^3 = 0.9$, and *Case 3* with $\eta^1 = 0.4, \eta^2 = 0.5$ and $\eta^3 = 0.6$. In the case of a tracker with a large number of particles, the effect of variations in weight is barely noticeable in the fusion result. This is due to the fact that the interaction weights are proportional to the percentage of particles state exchange between the trackers. As an additional parameter-sensitivity analysis, the results for different sets of u_{Th_1} and u_{Th_2} are shown in Fig. 5.8(b). In order

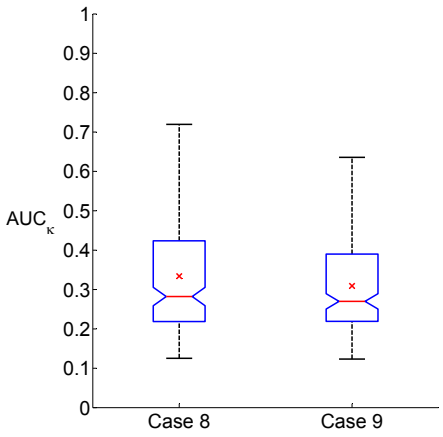


Figure 5.9: Comparison of performance of the fusion framework between normal appearance model update, Case 8: update for all p, and proposed update strategy, Case 9: update for only $p \in \{p_1, p_2\}$. The mean AUC_κ, indicated by mark '**x**', in the box plots for Cases 8 and 9 are 0.34 and 0.31, respectively.

to generate the results, we use *Case 2* of the weight. The best fused output is obtained in *Case 5* with $u_{Th_1} = 1$ and $u_{Th_2} = 2$. *Case 7* with $u_{Th_1} = u_{Th_2} = \infty$ is equivalent to the assumption that the trackers perform well, independently of the values given by the online performance measure. As expected, the result from *Case 7* is the worst of all the other cases. However, *Case 7* still involves prior exchange between the trackers at every frame with $\eta^2 = 0.5$, resulting in a better performance compared to the independent use of the trackers, as discussed below.

Figure 5.9 shows the performance of the fusion framework with the proposed selective appearance model update compared to a normal model update technique applied independently of the quality measure. Averaged over all the datasets, the proposed model update (*Case 9*), defined in reference to the performance measure levels for $p \in \{p_1, p_2\}$, results in 3% improvement compared to normal model update at every track output (*Case 8*). In the cases of CAVIAR R_1, STUDENTS S_3 and SINGER1 the proposed model update gives the best improvements of 11%, 7% and 5%, respectively.

Figure 5.10 shows sample comparisons of trackers performance using O_A values. A poorly performing tracker is characterised by small O_A values. STUDENTS S_1 undergoes occlusions, changes of motion dynamics and blur in a crowded scene, causing *LSS* to fail around frame 75 (Fig. 5.10(a)). STUDENTS-S_3 undergoes strong background changes resulting in failure of *CHPF* and *LSS* around frames 87 and 122, respectively (Fig. 5.10(b)). Using the fusion framework, we are able to use the best performing tracker and obtain robust tracking results for both STUDENTS-S_1 and STUDENTS-S_3. The target in CAVIAR-$R1$ dataset undergoes occlusion,

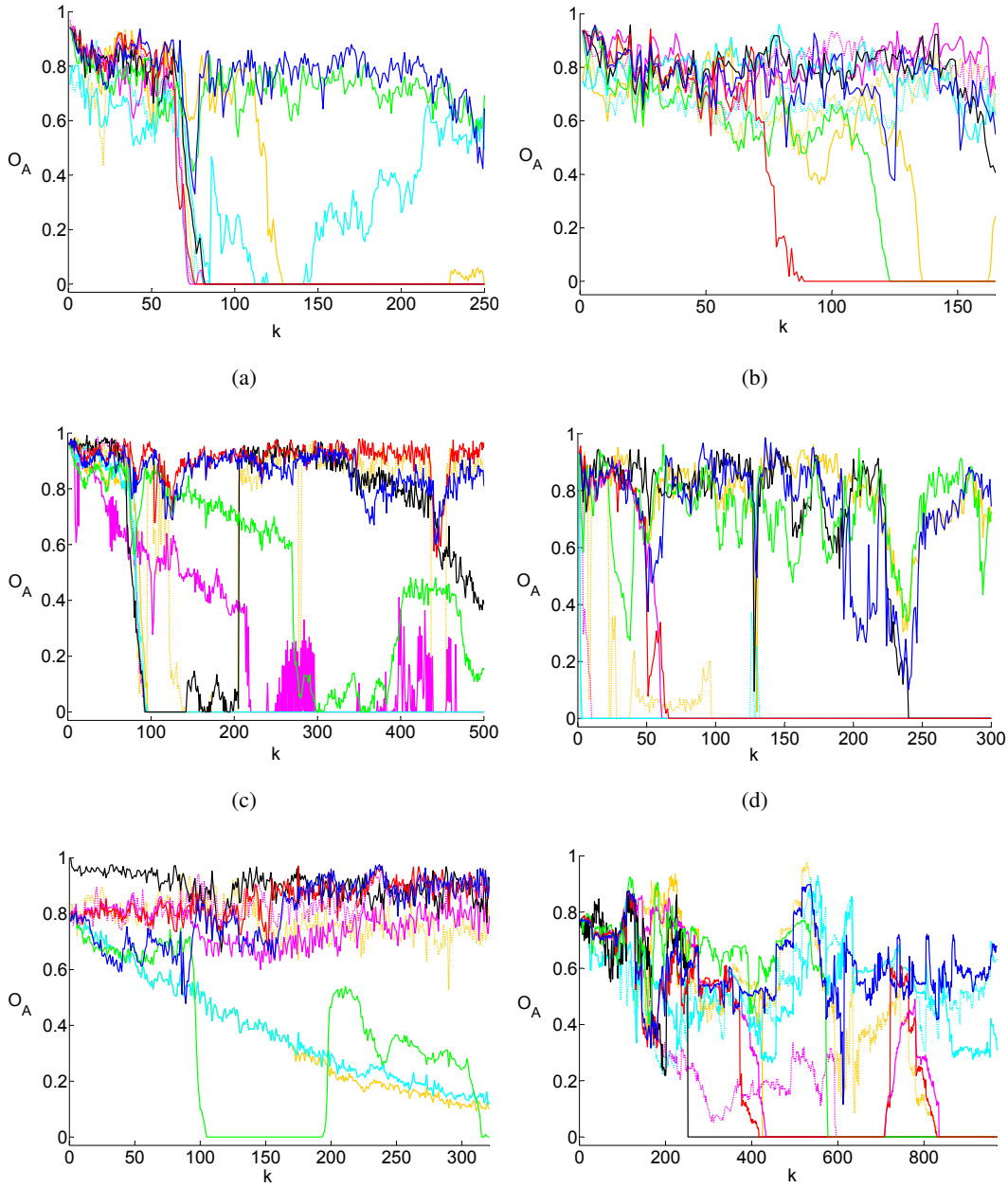


Figure 5.10: O_A scores for the trackers under analysis for selected sequences. A decrease in O_A value indicates a tracking performance degradation and in particular a zero value indicates a complete failure. (a) STUDENTS-S_1, (b) STUDENTS-S_3, (c) CAVIAR-R_1, (d) VISOR, (e) SINGER1 and (f) PANDA (see Table 5.1). —:F, —:*LSS*, —: *CHPF*,—: *VTS*, —: *CT*,- - -: *FCT*, —: 3*DDCT*, - - -: *MTT*, —: *AFT*, - - -: *L1T*.

abrupt motion and large scale changes. The performance of *CHPF* degrades through the sequence and fails around frame 300. The *LSS* tracker overcomes the challenges and achieves good tracking performance (Fig. 5.10(c)). Using *F* the target in CAVIAR-R_1 is successfully tracked. The accuracy of *F* is slightly smaller than that of *LSS* due to the poor performance of

CHPF.

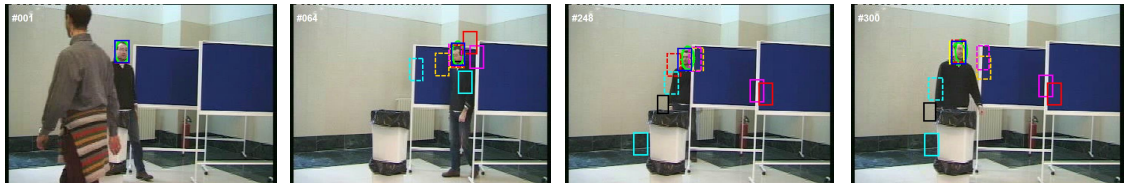
In VISOR, *CHPF* performs well due to the colour difference between face and background even when the face undergoes sudden change in motion and partial occlusions. However, due to the similarity in intensity between the face and background in the gray-scale image, and abrupt change in its state, *LSS* fails around frame 70 (Fig. 5.10(d)). The result from F shows that the target is tracked successfully. The small values of O_A from F at frame numbers such as 200 and 240 are due to the small size of the target, which causes a sudden drop in the lost-track value when the target undergoes sudden motion changes and due to the *MMS* formulation of the *CHPF* (*CHPF* performs well in the fusion). In SINGER1 *CHPF* fails around frame 100 (Fig. 5.10(e)) due to scale and illumination changes, and in PANDA both *LSS* and *CHPF* fail around frames 400 and 580 (Fig. 5.10(f)), respectively, due to out-of-plane rotation, scale changes and cluttered background. Using F the targets are tracked successfully even though one or both of the tracking components failed. In the PANDA sequence, small values of O_A are due to the scale changes from the out-of-plane rotations that the trackers can not cope with. Note that when all tracker components fail at the same time, it is highly probable that F fails too.

In Fig. 5.10, we observe the complementary nature in performance of *CHPF* and *LSS* for some of the targets. In STUDENTS-S_1, *CHPF* performs well, while the *LSS* fails around frame 75; whereas in CAVIAR-R_1 *LSS* performs well, while *CHPF* fails around frame 300. However, in both targets robust tracking is obtained using F by exploiting the best performing tracker in the mutually supportive fusion framework. The results of other trackers considered for our comparison are also shown in Fig. 5.10. *VTS* fails for STUDENT-S_1, CAVIAR-R_1 and PANDA around frames 80, 100 and 150, respectively, and the results can be interpreted likewise for other trackers.

Tracking results for some of the datasets are shown in Fig. 5.11. The videos of the tracking results for all datasets can be found in `http://www.eecs.qmul.ac.uk/~andrea/tlf.html`. The video results in the website allow one to compare qualitatively the performance of our proposed framework with the competing methods.

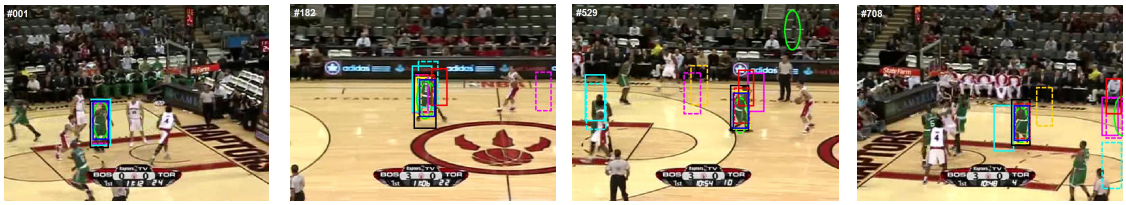
The advantages of the fusion framework are improvement in tracking performance compared to the performance of individual trackers in the framework, handling challenges that cannot be addressed by a single tracker and effectiveness for tracking an arbitrary target where variety of tracking challenges might exist. Tables 5.3 summarises the quantitative evaluation using O_A. The

(a)

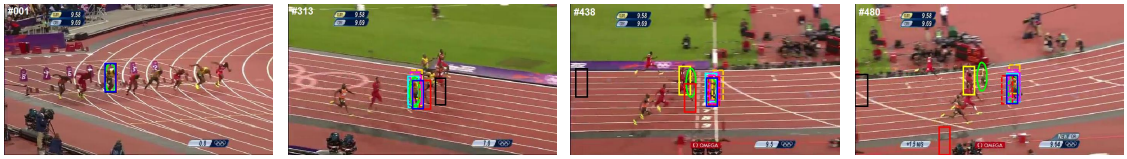


(b)

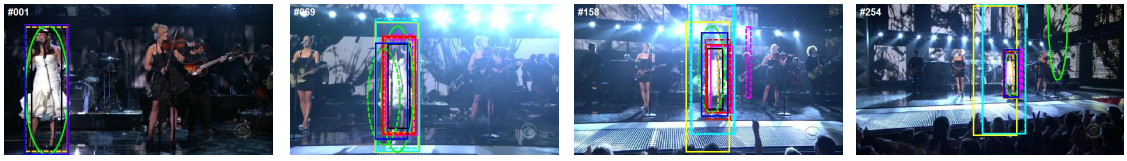
(c)



(d)



(e)



(f)

Figure 5.11: Sample tracking results. (a) STUDENTS-S_1, (b) VISOR, (c) PANDA, (d) BASKETBALL, (e) MEN100M-B_1 and (f) SINGER1 (see Table 5.1). —:F, - - -:LSS_F, - - -: $CHPF_F$, —:LSS, —: $CHPF$,—: VTS, —: CT, - - -: FCT, —: $3DDCT$, - - -: MTT, —: AFT, - - -: $L1T$. For the videos with the tracking results, please see:`http://www.eecs.qmul.ac.uk/~andrea/tlf.html`

average value μ over the video is shown. Since the trackers under comparison are probabilistic, the results reported are obtained by averaging 10 independent runs for each datasets. The stan-

dard deviation σ of each evaluation measure is indicated in the same cell next to μ. In order to show the variations in the results over the datasets, the standard deviation values in the last row of the table (**average** tag) are estimated for the average values of each dataset result. By comparing the columns *CHPF* and *LSS* with columns $CHPF_F$ and LSS_F it is possible to observe that a failure in one tracker is corrected by the other tracker in the proposed fusion framework. For instance, for target STUDENTS-S_1 the performance of *LSS* has improved by 52% in O_A as in LSS_F and for target CAVIAR-R_1 the performance of *CHPF* has improved by 28% in O_A as in $CHPF_F$. In most of the targets, the output accuracy of F is slightly higher than the performance of $CHPF_F$ and LSS_F.

In some of the targets such as CAVIAR-R_1 and DAVIDOUTDOOR the results from F are slightly lower than that of LSS_F. However, the O_A values do not show large differences, i.e. 0.05 and 0.09, respectively. For the targets in STUDENTS-S_2, CAVIAR-R_1 and DAVIDOUTDOOR, the trackers operating in the fusion framework ($CHPF_F$ or LSS_F) generate less accurate results than that of the best performing individual trackers (*CHPF* or *LSS*). This occurs due to the lack of perfection in the spatial uncertainty analyses as performance measure. The spatial uncertainty analyses on the particles state depend on the noises in the motion model and observation models; and the result of the performance measure may not be absolute all the time. Averaged overall the datasets, on average the decrease in accuracy of the well-performing tracker by the poorly performing tracker is only 2.7% in O_A compared to 23.3% improvement in O_A achieved by the poorly performing tracker in the framework.

A tracker that fails in the middle of the sequence is associated to a large value of σ (e.g. *CHPF* in targets STUDENTS-S_2, CAVIAR-R_1 and DAVIDOUTDOOR; *LSS* in STUDENTS-S_1, VISOR and SINGER2; *CT* in SKATING2 and BASKETBALL; *VTS* in CAVIAR-R_1 and MEN100m). In addition to this, the value of σ indicates the variation of results over different runs of the probabilistic trackers and the fusion framework, such as the performance of F in STUDENTS-S_2 and CAVIAR-R_2. For multiple runs on STUDENTS-S_2, CAVIAR-R_2, SINGER2 and DAVIDOUTDOOR, we noticed that F fails for some of the runs, and as a result F has relatively larger values of σ for those targets compared to the values of σ for other targets. In SINGER2 both *CHPF* and *LSS* perform poorly, and the fusion framework does not cope with the poor performance of both trackers for a long duration, unlike MEN100M.

The performance of other state-of-the-art trackers is shown in Table 5.3. *CT* and *AFT* can-

Table 5.3: Comparison of trackers performance using overlap area score, O_A. The larger O_A value, the better tracking result. Key - μ: average O_A, σ: O_A standard deviation, $L1T$: ℓ_1 minimisation based tracker, MTT: multi-task learning based tracker, AFT: adaptive fragment based tracker, $3DDCT$: incremental 3d discrete cosine transform representation based tracker, FCT: fast compressive tracker, CT: real-time compressive tracker, VTS: tracking by sampling tracker, $CHPF$: colour histogram particle filter, LSS least soft-thresold squares tracker, $CHPF_F$: $CHPF$ running under the fusion framework, LSS_F: LSS tracker under the fusion framework, F: proposed fusion framework. *: the authors start at a different frame number in [57].

Dataset		**L1T** μ/σ	**MTT** μ/σ	**AFT** μ/σ	**3D DCT** μ/σ	**FCT** μ/σ	**CT** μ/σ	**VTS** μ/σ	**CHPF** μ/σ	**LSS** μ/σ	**CHPF$_F$** μ/σ	**LSS$_F$** μ/σ	**F** μ/σ
STUDENTS	S_1	**.22/** .35	**.20/** .33	**.52/** .35	**.21/** .34	**.19/** .31	**.39/** .24	**.23/** .36	**.76/** .09	**.23/** .37	**.74/** .10	**.75/** .12	**.77/** .10
	S_2	**.62/** .27	**.84** .06	**.82/** .10	**.85/** .06	**.63/** .15	**.77/** .06	**.83/** .06	**.40/** .36	**.76/** .09	**.61/** .10	**.67/** .23	**.71/** .24
	S_3	**.71/** .10	**.66/** .25	**.51/** .29	**.82/** .07	**.67/** .06	**.78/** .08	**.55/** .09	**.55/** .27	**.36/** .38	**.68/** .22	**.67/** .23	**.72/** .21
CAVIAR	R_1	**.54/** .42	**.15/** .34	**.15/** .32	*__.28/__ .30	**.15/** .33	**.15/** .32	**.62/** .36	**.47/** .32	**.92/** .06	**.75/** .10	**.91/** .05	**.86/** .06
	R_2	**.69** /.29	**.21/** .32	**.37/** .30	**.16/** .31	**.26/** .25	**.27/** .24	**.74/** .12	**.66/** .10	**.80/** .09	**.75/** .22	**.74/** .22	**.80/** .24
MEN100m	B_1	**.10/** .25	**.31/** .37	**.56/** .35	**.80/** .06	**.65/** .13	**.64/** .13	**.41/** .40	**.51/** .27	**.40/** .34	**.61/** .16	**.61/** .23	**.64/** .17
	B_2	**.27/**. 31	**.47/** .31	**.57/** .32	**.73/** .06	**.69/** .10	**.73/** .10	**.43/** .36	**.46/** .33	**.19/** .31	**.57/** .27	**.36/** .30	**.53/** .28
VISOR		**.08/** .22	**.24/** .35	**.77/** .15	**.15/** .31	**.01/** .06	**.01/** .06	**.62/** .35	**.71/** .12	**.14/** .30	**.74/** .12	**.69/** .28	**.75/** .17
PETS2006		**.50/** .39	**.40/** .40	**.77/** .06	**.70/** .17	**.60/** .19	**.72/** .10	**.75/** .06	**.76/** .07	**.77/** .20	**.81/** .07	**.76/** .13	**.83/** .07
SINGER1		**.75/** .12	**.46/** .40	**.38/** .20	**.76/** .06	**.39/** .19	**.39/** .19	**.87/** .11	**.33/** .30	**.85/** .06	**.66/** .25	**.79/** .20	**.74/** .20
SINGER2		**.31/** .36	**.31/** .33	**.57/** .23	**.54/** .30	**.49/** .27	**.43/** .30	**.75/** .20	**.20/** .32	**.56/** .28	**.41/** .33	**.41/** .36	**.41/** .34
SINGER2^{face}		**.02/** .12	**.69/** .11	**.71/** .12	**.35/** .35	**.02/** .11	**.02/** .06	**.72/** .12	**.57/** .22	**.46/** .29	**.68/** .12	**.60/** .18	**.73/** .13
SKATING2		**.05/** .17	**.04/** .15	**.43/** .32	**.15/** .23	**.26/** .26	**.48/** .27	**.75/** .20	**.69/** .14	**.12/** .24	**.69/** .16	**.68/** .18	**.69/** .16
DAVIDOUTDOOR		**.17/** .33	**.25/** .37	**.47/** .39	**.31/** .33	**.29/** .36	**.51/** .31	**.52/** .34	**.42/** .36	**.69/** .16	**.60/** .32	**.56/** .33	**.61/** .36
BASKETBALL		**.22/** .30	**.07/** .22	**.80/** .10	**.22/** .32	**.24/** .30	**.23/** .33	**.77/** .10	**.63/** .30	**.47/** .41	**.70/** .25	**.75/** .24	**.75/** .25
PANDA		**.42/** .30	**.24/** .26	**.30/** .34	**.33/** .33	**.55/** .14	**.53/** .14	**.34/** .34	**.48/** .30	**.28/** .30	**.61/** .17	**.54/** .20	**.57/** .18
MOTOR1		**.33/** .42	**.64/** .35	**.61/** .36	**.63/** .32	**.23/** .33	**.23/** .32	**.67/** .33	**.57/** .28	**.40/** .43	**.65/** .16	**.63/** .23	**.67/** .19
MOTOR2		**.71/** .08	**.72/** .07	**.74/** .06	**.75/** .06	**.69/** .06	**.68/** .10	**.71/** .09	**.55/** .18	**.75/** .05	**.60/** .17	**.68/** .22	**.66/** .20
CHASING		**.69/** .33	**.82/** .12	**.74/** .19	**.75/** .15	**.72/** .11	**.04/** .01	**.56/** .37	**.75/** .13	**.78/** .17	**.72/** .15	**.78/** .16	**.78/** .14
average		**.39/** .27	**.41/** .23	**.57/** .24	**.50/** .22	**.41/** .19	**.41/** .18	**.64/** .22	**.55/** .24	**.53/** .24	**.66/** .21	**.66/** .20	**.70/** .20

not deal with scale changes, thus resulting in a poor performance in CAVIAR-R_1 and R_2, and SINGER2. For small targets (faces) with background clutter in VISOR and SINGER2$^{\text{face}}$, CT performs the worst. FCT has a similar implementation to CT, however the use of the best performing colour channel of RGB in STUDENTS, SKATING2 and DAVIDOUTDOOR datasets for CT results in a better performance over the use of gray-scale image for FCT. Although $3DDCT$ performs the best on some targets, the results are not satisfactory for the targets where the colour distribution plays an important role in target discrimination such as in VISOR, SKATING2, BASKETBALL and CAVIAR-R_2. $L1T$ performs poorly on strong articulated changes of the target such as in MEN100m, SINGER2 and SKATING1, and MTT fails due to occlusions in STUDENTS2, BASKATEBALL and CAVIAR1. VTS performs relatively well on most of the datasets. However it fails to track the whole sequence for STUDENTS-S_1, MEN100m and PANDA due to occlusion and background clutter.

F gives 15% and 17% improvement in O_A over the individual trackers $CHPF$ and LSS, respectively, proving that the fusion framework achieves a more robust tracking. VTS tracker outperforms $CHPF$ and LSS, F has a 6% improvement in O_A over VTS. The explicit performance measure computed for each tracker and the exchange of prior between trackers as correction give better tracking results compared to directly using the likelihood values as done in VTS. Compared to the other trackers $L1T$, MTT, AFT, $3DDCT$, FCT and CT, the fusion F results in a performance improvement of 31%, 29%, 13%, 20%, 29% and 29% in O_A values, respectively.

5.8 Summary

We presented a fusion framework that allows us to obtain self-correcting target tracking. The framework measures online the performance of individual trackers for guiding the fusion decision. Trackers in the framework assist each other based on an appropriate prior state exchange, referred to as prior state correction. The individual trackers and their fusion are able to avoid and recover from failures due to the online performance based prior state correction. In addition to this, we exploited the result of the performance measure for selecting appropriate states for the appearance model update in order to minimise drifting.

The framework is demonstrated using two Bayesian trackers [76, 107] and challenging datasets containing 19 targets and a total of 8482 frames. The proposed selective model update results in an average of 3% improvement in overlap score compared to the normal (every frame) model

update. Moreover, the framework results in averages of 15% and 17% improvement of overlap score over the individual trackers considered. Finally, the framework is shown to have improved performance compared to other state-of-the-art trackers [52, 27, 68, 131, 130, 129, 57].

Chapter 6

Conclusions

6.1 Summary of achievements

In this thesis, we addressed the problem of how to achieve self-correcting tracking that can avoid and recover from tracking failures. Targets undergo challenges such as deformation, variable motion dynamics changes, illumination changes and be similar to background clutter that lead to performance degradation for trackers. Except [48, 87, 120, 25, 24, C1], most existing state-of-the-art tracking methods do not explicitly detect poor performance and failures, and do a correction in the state estimation. To address this, we proposed a generalised Track-Evaluate-Correct framework, where a tracker is assisted by online evaluation and correction blocks in order to obtain the self-correcting tracking. The evaluation block allows the tracker to gather information about its track quality, and the correction block helps the tracker to avoid and recover from poor performance and failures. We discussed various types of performance measures and, similarly, we discussed different means of correcting a tracker. The main concern for designing self-correcting tracking is identifying appropriate strategies to be employed in the evaluation and correction blocks. The thesis mainly considered tracking in a Bayesian framework. In particular, we addressed the problem of representation and formalisation of self-correcting tracking by using DBN and the problem of obtaining the self-correcting tracking by embedding appropriate evaluation and correction blocks to the state-of-the art trackers. Details of the particular achievements are presented below.

We utilised DBN model for representation of self-correcting tracking. In the DBN model

representation, we used additional hidden variables for the evaluation and correction together with variables for the target state and measurement. The inclusion of evaluation and correction variables results in deep-hierarchical DBN model when compared with the DBN model of a baseline tracker with only target state and measurement variables. From the inference between variables, we showed the filtering equations of the target state estimation that depends on the evaluation and correction variables. In particular we showed the dependence of models, such as motion and appearance, on the evaluation and correction variables in self-correcting tracking. Both the evaluation and correction variables in the DBN are discrete. In the case of evaluation, the discrete values correspond to classes that characterise the quality of tracks, while in the case of correction, the different discrete values primarily allow the self-correcting framework to decide whether correction is needed or not. Additionally, the different values for correction variables can indicate different correction schema on the tracker, such as correction on the motion model or correction in the appearance model. Moreover, for a tracking scenario with multiple models, the different correction variable values help to select the appropriate model or decide their optimal combinations. By using the Track-Evaluate-Correct framework, we provided filtering equations for the proposed DBN model of the self-correcting tracking.

We used the Track-Evaluate-Correct framework for improving probabilistic tracking of an extended object with a set of model points [J1]. Self-correcting tracking requires correction information about the true state of the target in cases of failures. In the tracking of the extended object, we exploited the correlation information between model points tracks for correction. Tracking involves estimating the state and identify for each of the model points. However, challenges from data association, misdetection and clutter lead to tracking failures. We modelled sources of failure and covariance characteristics of the tracker in order to measure the quality of tracks. Based on the measure, we classify model point trackers into strong and weak trackers. Weak trackers are those having poor performance or uncertain about their true state. The correction of weak trackers is achieved in the form of re-initialisation based on an online learnt correlation model. Inferring the corrected state of the weak tracker from that of strong trackers is done by Partial Least Square regression using trackers short-windowed trajectory. For an accurate estimation of a weak tracker state, the predictions from the list of available strong trackers are weighted according to the observed correlation level. The correlation level between trackers is observed during the learning step of the regression model. We showed the improvement in performance

of our proposed method compared to other state-of-the-art methods on tracking of data obtained from optical motion capture systems.

We have proposed a tracker-level fusion framework as a means of achieving self-correcting tracking [J2]. There is a need for correction information in case of poor performance and failures of trackers. The individual trackers are used as the source of correction information for each other in the tracker-level fusion. In the framework we incorporated an online performance measure to identify the track quality level of individual trackers. We classified the track quality in to three classes: well, medium and poorly performing. The quality levels are used for guiding the fusion and collaboration for the trackers. We defined the collaboration strategy using prior states of trackers. The collaboration strategy is a form prior correction for poorly performing trackers. For recovering poor performance of the tracker, its prior is partially or fully replaced with the prior state of a well performing tracker. In addition to this, corrections on trackers in the fusion framework are done in the form of appearance model update. The model update considers only tracks with medium and good performance classes aimed at minimising the drifting due to wrong updates. We demonstrate the performance of the proposed fusion framework by using two particle filter based Bayesian trackers on challenging datasets. Trackers are selected taking into account variation in trackers' components and complementary performance to tracking challenges. We compared with state-of-the-art methods and show the improvement in performance by our method in the experimental analysis and validation. Moreover, we did sensitivity analysis of the fusion framework for its parameters as part of the experiment.

In summary, we presented a framework that takes into consideration the capabilities to avoid and recover from failures in target tracking. The framework achieves the capabilities by the explicit use of online track quality measurement and correction blocks for trackers. We used DBN as a tool for representing the design of self-correcting tracking. Moreover, we employed the framework for tracking an extended object with model points and fusion of multiple trackers.

6.2 Future research directions

This section summarises possible future research directions of the thesis as follow.

1. In this thesis, step-wise integration of the tracker with the evaluation and correction blocks is used in the Track-Evaluate-Correct framework. Other frameworks to obtain self-correcting tracking need to be addressed for maximising self-correcting capabilities. The new frame-

work could be explored from techniques to obtain an approximate inference of the proposed DBN model for self-correcting tracking.

2. For recovering trackers from failures, the main challenge lays in obtaining information about the true state and updated properties of the target. Context information in the scene could be exploited as a means of getting the information for correcting the trackers.

3. The proposed correlation-based self-correcting tracking in Chapter 4 can be extended to other multi-target tracking scenarios. The correlation information between trackers, which has been used for correction, can be exploited to assist hard decision in data association strategies. Moreover, improved techniques for correlation modelling could be considered.

4. Threshold-free performance measures need to be investigated for better design of self-correcting tracking. Additionally, the concept of performance measure needs to be extended also for detection of tracking challenges. The detected tracking challenge can be accompanied with appropriate tracking strategies and corrections on the tracker to overcome the challenge. This approach minimises failures and also partially overcomes the problem of obtaining correction information in self-correcting tracking.

5. We showed that fusion of trackers helps us to obtain self-correcting tracking (Chapter 5). The fusion presented uses evaluation and collaboration strategy based on particle filter. The framework can be extended to contain non-Bayesian trackers and a collaborative method independent of the tracking strategies needs to be addressed. Additionally, in the fusion framework, a mechanism needs to be designed for minimising the effect of a poorly performing tracker on a well-performing tracker.

6. In tracker-level fusion complementary performance characteristics of trackers play an important role in avoiding failures. Methods to identify the complementary performance of trackers need to be studied. In order to have target reacquisition after long-term occlusions, a tracker with such capability should be considered in the fusion framework.

Bibliography

[1] Carnegie Mellon University motion capture database. http://mocap.cs.cmu.edu. Last accessed: March 2014.

[2] Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. http://vision.cs.brown.edu/humaneva/index.html. Last accessed: March 2014.

[3] S. Avidan. Support vector tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(8):1064–1072, August 2004.

[4] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(6):562–575, June 1995.

[5] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *IEEE Control Syst. Mag.*, 29(6):82–100, December 2009.

[6] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2001.

[7] D. Barber and A.T. Cemgil. Graphical models for time-series. *IEEE Signal Process. Mag.*, 27(6):18–28, November 2010.

[8] A. Bargi, R. Y. D. Xu, and M. Piccardi. An online HDP-HMM for joint action segmentation and classification in motion capture data. In *Proc. of Conf. on Computer Vision and Pattern Recognition Workshops*, pages 1–7, June 2012.

[9] C. Bracegirdle and D. Barber. Switch-Reset models: Exact and approximate inference. In *Proc. Int. Conf. on Artificial Intelligence and Statistics*, volume 15, pages 190–198, April 2011.

[10] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. In *Proc. of SPIE: Optical Pattern Recognition*, volume 4387, pages 95–102, March 2001.

[11] M. B. Caglar and N. da Vitoria Lobo. Self correcting tracking for articulated objects. In *Proc. of Int. Conf. on Automatic Face and Gesture Recognition (FGR)*, pages 609–616, April 2006.

[12] L. Cehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(4):941–953, April 2013.

[13] A. B. Chan and N. Vasconcelos. Counting people with low-level features and Bayesian regression. *IEEE Trans. Image Process.*, 21(4):2160–2177, April 2012.

[14] H. J. Chang, M. S. Park, H. Jeong, and J. Y. Choi. Tracking failure detection by imitating human visual perception. In *Proc. of IEEE Int. Conf. on Image Processing*, pages 3293–3296, September 2011.

[15] D. P. Chau, J. Badie, F. Bremond, and M. Thonnat. Online tracking parameter adaptation based on evaluation. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance*, pages 189–194, August 2013.

[16] P. Chawah, A. Sourice, Guy Plantier, and J. Chery. Real time and adaptive Kalman filter for joint nanometric displacement estimation, parameters tracking and drift correction of EFFPI sensor systems. In *IEEE Sensors*, pages 882–885, October 2011.

[17] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, May 2003.

[18] Student Dave. Multi bug (object) tracking. http://studentdavestutorials.weebly.com/multi-bugobject-tracking.html. Last accessed: March 2014.

[19] C. Dicle, O. I. Camps, and M. Sznaier. The way they move: Tracking multiple targets with similar appearance. In *Proc. of IEEE Int. Conf. on Computer Vision*, pages 1–8, December 2013.

[20] P. M. Djuric, Jayesh H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez. Particle filtering. *IEEE Signal Process. Mag.*, 20(5):19–38, 2003.

[21] A Dore, M. Soto, and C.S. Regazzoni. Bayesian tracking for video analytics. *IEEE Signal Process. Mag.*, 27(5):46–55, September 2010.

[22] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, July 2000.

[23] Uri Dubin. Probabilistic Data Association Filters (pdaf) - a tracking demo. http://www.mathworks.co.uk/matlabcentral/fileexchange/34146. Last accessed: March 2014.

[24] S. Duffner and J. Odobez. Exploiting long-term observations for track creation and deletion in online multi-face tracking. In *Proc. of IEEE Int. Conf. on Automatic Face Gesture Recognition and Workshops*, pages 525–530, March 2011.

[25] S. Duffner and J. Odobez. Track creation and deletion framework for long-term online multiface tracking. *IEEE Trans. Image Process.*, 22(1):272–285, January 2013.

[26] C. E. Erdem, B. Sankur, and A. M. Tekalp. Performance measures for video object segmentation and tracking. *IEEE Trans. Image Process.*, 13(7):937–951, July 2004.

[27] E. Erdem, S. Dubuisson, and I. Bloch. Fragments based tracking with adaptive cue integration. *Computer Vision and Image Understanding*, 116(7):827–841, July 2012.

[28] P. Fearnhead. Exact and efficient Bayesian inference for multiple changepoint problems. *Statistics and Computing*, 16(2), June 2006.

[29] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE J. Ocean. Eng.*, 8(3):173–184, July 1983.

[30] S. Fruhwirth-Schnatter. *Finite Mixture and Markov Switching Models: Modeling and Applications to Random Processes*. Springer series in statistics. Springer, 2006.

[31] P. Gabriel, J.-B. Hayet, J. Piater, and J. Verly. Object tracking using color interest points. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance*, pages 159–164, September 2005.

[32] Y. Gao, R. Ji, L. Zhang, and A. Hauptmann. Symbiotic tracker ensemble towards a unified tracking framework. *IEEE Trans. Circuits Syst. Video Technol.*, 24(7):1122–1131, July 2014.

[33] X. Gu, H. Wang, L. Wang, and C. Pan. Adaptive multi-cue fusion for visual target tracking based on uncertainly measure. In *Proc. of Int. Conf. of Image and Vision Computing New Zealand (IVCNZ)*, pages 1–8, November 2010.

[34] G. B. Guerra-filho. Optical motion capture: Theory and implementation. *Journal of Theoretical and Applied Informatics*, 12(2):61–89, 2005.

[35] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Trans. Signal Process.*, 50(2):425–437, February 2002.

[36] D. Hall. Automatic parameter regulation of perceptual systems. *Image Vision Computing*, 24(8):870–881, August 2006.

[37] A. Hampapur, R. Bobbitt, L. Brown, M. Desimone, R. Feris, R. Kjeldsen, M. Lu, C. Mercier, C. Milite, S. Russo, Chiao fe Shu, and Yun Zhai. Video analytics in urban environments. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance*, pages 128–133, September 2009.

[38] J. Hartikainen and S. Särkkä. *RBMCDAbox-Matlab Toolbox of Rao-Blackwellized Data Association Particle Filters*. Department of Biomedical Engineering and Computational Science, Helsinki University of Technology, 2008.

[39] F. He, Z. Tian, X. Liu, and X. Duan. A fast edge tracking algorithm for image segmentation using a simple Markov Random Field model. In *Proc. of Int. Conf. on Computer Science and Electronics Engineering (ICCSEE)*, pages 633–636, March 2012.

[40] W. He, T. Yamashita, H. Lu, and S. Lao. SURF tracking. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 1586–1592, September 2009.

[41] W. Hu, X. Zhou, W. Li, W. Luo, X. Zhang, and S. Maybank. Active contour-based visual tracking by integrating colors, shapes, and motions. *IEEE Trans. Image Process.*, 22(5):1778–1792, May 2013.

[42] C. Hua, H. Wu, Q. Chen, and T. Wada. K-means tracker: A general algorithm for tracking people. *Journal of Multimedia*, 1(4):46–53, July 2006.

[43] M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *Int. Journal of Computer Vision*, 29:5–28, August 1998.

[44] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1296–1311, October 2003.

[45] Z. Jiang, D. Q. Huynh, W. Moran, and S. Challa. Tracking pedestrians using smoothed colour histograms in an interacting multiple model framework. In *Proc. of IEEE Int. Conf. on Image Processing*, pages 2313–2316, September 2011.

[46] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *Proc. of IEEE Int. Conf. on Computer Vision Workshops*, pages 1417–1424, September 2009.

[47] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Proc. of Int. Conf. on Pattern Recognition (ICPR)*, pages 2756–2759, August 2010.

[48] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, July 2012.

[49] C.-J. Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60(1-2):1–22, January 1994.

[50] S. Kwak, W. Nam, B. Han, and J. H. Han. Learning occlusion with likelihoods for visual tracking. In *Proc. of IEEE Int. Conf. on Computer Vision*, pages 1551–1558, November 2011.

[51] J. Kwon and K.-M. Lee. Visual tracking decomposition. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 1269–1276, June 2010.

[52] J. Kwon and K. M. Lee. Tracking by sampling trackers. In *Proc. of IEEE Int. Conf. on Computer Vision*, pages 1195–1202, November 2011.

[53] J. Kwon and K.-M Lee. Tracking by sampling and integrating multiple trackers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1428–1441, July 2014.

[54] I. Leichter, M. Lindenbaum, and E. Rivlin. A probabilistic framework for combining tracking algorithms. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 445–451, June 2004.

[55] C. Leistner, H. Grabner, and H. Bischof. Semi-supervised boosting using visual similarity learning. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[56] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):655–664, October 2007.

[57] X. Li, A. Dick, C. Shen, A. Van Den Hengel, and H. Wang. Incremental learning of 3D-DCT compact representations for robust visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(4):863–881, April 2013.

[58] B. Liu, J. Huang, L. Yang, and C. Kulikowsk. Robust tracking using local sparse appearance model and K-selection. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 1313–1320, June 2011.

[59] H. Loeliger. An introduction to factor graphs. *IEEE Signal Process. Mag.*, 21(1):28–41, January 2004.

[60] L. Lu, X.-T. Dai, and G. Hager. A particle filter without dynamics for robust 3D face tracking. In *Proc. of Conf. on Computer Vision and Pattern Recognition Workshops*, pages 70–70, June 2004.

[61] Y. Lu, C. Guo, and T. Ikenaga. Estimation-correction scheme based articulated object tracking using SIFT features and mean shift algorithm. In *Proc. of Int. Conf. on New Trends in Information Science and Service Science*, pages 275–280, May 2010.

[62] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of Int. Joint conf. on Artificial Intelligence*, volume 2, pages 674–679, 1981.

[63] E. Maggio and A. Cavallaro. Hybrid particle filter and mean shift tracker with adaptive transition model. In *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 221–224, March 2005.

[64] E. Maggio and A. Cavallaro. *Video Tracking: Theory and Practice*. Wiley, 2011.

[65] E. Maggio, F. Smerladi, and A. Cavallaro. Adaptive multifeature tracking in a particle filtering framework. *IEEE Trans. Circuits Syst. Video Technol.*, 17(10):1348–1359, October 2007.

[66] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking: a survey. *IEEE Trans. Aerosp. Electron. Syst.*, 34(1):103–123, January 1998.

[67] B. McCane, B. Galvin, and K. Novins. Algorithmic fusion for more robust feature tracking. *Int. Journal of Computer Vision*, 49(1):79–89, September 2002.

[68] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(11):2259–2272, November 2011.

[69] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.

[70] A. Mittal and A. A. Kassim. *Bayesian Network Technologies: Applications and Graphical Models*. IGI publishing, 2007.

[71] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, July 2002.

[72] T. Nawaz and A. Cavallaro. PFT: A protocol for evaluating video trackers. In *Proc. of IEEE Int. Conf. on Image Processing*, pages 2325–2328, September 2011.

[73] T. Nawaz and A. Cavallaro. A protocol for evaluating video trackers under real-world conditions. *IEEE Trans. Image Process.*, 22(4):1354–1361, April 2013.

[74] K. Nickels and S. Hutchinson. Estimating uncertainty in SSD-based feature tracking. *Image and Vision Computing*, 20(1):47–58, January 2002.

[75] U. Nodelman, C. R. Shelton, and D. Koller. Learning continuous time Bayesian networks. In *Proc. of Int. Conf. on Uncertainty in Artificial Intelligence*, pages 451–458, 2003.

[76] K. Nummiaro, E. Koller-Meier, and L. Van Gool. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, January 2003.

[77] J. Odobez, D. Gatica-Perez, and S. O. Ba. Embedding motion in model-based stochastic tracking. *IEEE Trans. Image Process.*, 15(11):3514–3530, November 2006.

[78] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *Int. Journal of Computer Vision*, 77:103–124, May 2008.

[79] T. Okuma, T. Kurata, and K. Sakaue. A natural feature-based 3D object tracking method for wearable augmented reality. In *Proc. of IEEE Int. Workshop on Advanced Motion Control*, pages 451–456, March 2004.

[80] V. Pavlovic, J. M. Rehg, T. j. Cham, and K. P. Murphy. A Dynamic Bayesian Network approach to figure tracking using learned dynamic models. In *Proc. of Int. Conf. on Computer Vision*, volume 1, pages 94–101, September 1999.

[81] C. Piciarelli, G. L. Foresti, and L. Snidaro. Trajectory clustering and its applications for video surveillance. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance*, pages 40–45, September 2005.

[82] Simon J. D. Prince. *Computer vision: models, learning and inference*. Cambridge University Press, UK, 2011.

[83] Z. Qian, D. N. Metaxas, and L. Axel. Boosting and nonparametric based tracking of tagged MRI cardiac boundaries. In *Proc. of Int. Conf. on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 4190, pages 636–644, 2006.

[84] T. Rehrl, N. Theissing, A. Bannat, J. Gast, D. Arsic, F. Wallhoff, and G. Rigoll. Tracking using Bayesian inference with a two-layer Graphical Model. In *Proc. of IEEE Int. Conf. on Image Processing*, pages 3961–3964, September 2010.

[85] R. Rosipal and N. Kramer. Overview and recent advances in partial least squares. In *Lecture Notes in Computer Science in Subspace, Latent Structure and Feature Selection Techniques*, pages 34–51. Springer, 2006.

[86] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *Int. Journal of Computer Vision*, 77(1-3):125–141, May 2008.

[87] M. Sabet, R.A. Zoroofi, K.S. Niiat, and M. Sabbaghian. Automated face tracking with self correction capability. In *Proc. of Soft Computing and Pattern Recognition (SoCPaR)*, pages 280–284, October 2011.

[88] S. Salti, A. Cavallaro, and L. Di Stefano. Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE Trans. Image Process.*, 21(10):4334–4348, 2012.

[89] J. C. SanMiguel, A. Cavallaro, and J. M. Martinez. Adaptive online performance evaluation of video trackers. *IEEE Trans. Image Process.*, 21(5):2812–2823, May 2012.

[90] J. C. SanMiguel, A. Cavallaro, and J. M. Martinez. Standalone evaluation of deterministic video tracking. In *Proc. of IEEE Int. Conf. on Image Processing*, pages 1353–1356, September 2012.

[91] R. Schubert, H. Klöden, G. Wanielik, and S. Kälberer. Performance evaluation of multiple target tracking in the absence of reference data. In *Proc. of Int. Conf. on Information Fusion (FUSION)*, pages 1–7, July 2010.

[92] D. Schuhmacher, Ba-Tuong Vo, and Ba-Ngu Vo. A consistent metric for performance evaluation of multi-object filters. *IEEE Trans. Signal Process.*, 56(8):3447–3457, August 2008.

[93] T. Senst, V. Eiselein, and T. Sikora. Robust local optical flow for feature tracking. *IEEE Trans. Circuits Syst. Video Technol.*, 22(9):1377–1387, September 2012.

[94] C. Setchell and E. L. Dagless. Vision-based road-traffic monitoring sensor. *IEE Proc.-Vision, Image and Signal Processing*, 148(1):78–84, February 2002.

[95] J. Shao, G. Xie, J. Yu, and L. Wang. A tracking controller for motion coordination of

multiple mobile robots. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 783–788, August 2005.

[96] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1442–1468, July 2014.

[97] M. Soto and C. S. Regazzoni. A general Bayesian algorithm for visual object tracking based on sparse features. In *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 1181–1184, May 2011.

[98] V.B. Subburaman, A. Descamps, and C. Carincotte. Counting people in the crowd using a generic head detector. In *Proc. of IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance*, pages 470–475, September 2012.

[99] L. Sun and G. Liu. Visual object tracking based on combination of local description and global representation. *IEEE Trans. Circuits Syst. Video Technol.*, 21(4):408–420, April 2011.

[100] X. Sun, M. Chen, and A. Hauptmann. Action recognition via local descriptors and holistic features. In *Proc. of Conf. on Computer Vision and Pattern Recognition Workshops*, pages 58–65, June 2009.

[101] J. Supancic and D. Ramanan. Self-paced learning for long-term tracking. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 2379–2386, June 2013.

[102] F. Tang, S. Brennan, O. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *Proc. of IEEE Int. Conf. on Computer Vision*, pages 1–8, October 2007.

[103] F. Tang and H. Tao. Probabilistic object tracking with dynamic attributed relational feature graph. *IEEE Trans. Circuits Syst. Video Technol.*, 18(8):1064–1074, August 2008.

[104] J. Tinevez. Simple tracker. http://www.mathworks.co.uk/matlabcentral/fileexchange/34040. Last accessed: March 2014.

[105] N. Vaswani. Additive change detection in nonlinear systems with unknown change parameters. *IEEE Trans. Signal Process.*, 55(3):859–872, March 2007.

[106] J. Vermaak, N. Ikoma, and S.J. Godsill. Sequential Monte Carlo framework for extended object tracking. *IEE Proc. Radar, Sonar and Navigation*, 152(5):353–363, October 2005.

[107] D. Wang, H. Lu, and M.-H. Yang. Least soft-threshold squares tracking. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 2371–2378, June 2013.

[108] J. Wang, D. Zhao, W. Gao, and S. Shan. Interacting multiple model particle filter to adaptive visual tracking. In *IEEE Symp. on Multi-Agent Security and Survivability*, pages 568–571, 2004.

[109] P. Wang and H. Qiao. Online appearance model learning and generation for adaptive visual tracking. *IEEE Trans. Circuits Syst. Video Technol.*, 21(2):156 –169, February 2011.

[110] Q. Wang, F. Chen, W. Xu, and M. Yang. An experimental comparison of online object tracking algorithms. In *Proc. of SPIE: Image and Signal Processing*, pages 1–11, August 2011.

[111] Q. Wang, F. Chen, W. Xu, and M.-H. Yang. Online discriminative object tracking with local sparse representation. In *Proc. of IEEE Workshop on the Applications of Computer Vision*, pages 425–432, January 2012.

[112] S. Wang, H. Lu, and G. Yang. Complementary visual tracking. In *Proc. of IEEE Int. Conf. on Image Processing*, pages 477–480, September 2011.

[113] Z. Wei, L. Huchuan, and Y. Ming-Hsuan. Robust object tracking via sparsity-based collaborative model. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 1838–1845, June 2012.

[114] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):210–227, February 2009.

[115] H. Wu, A. C. Sankaranarayanan, and R. Chellappa. Online empirical evaluation of tracking algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1443–1458, August 2010.

[116] H. Wu and Q. Zeng. Self-evaluation for video tracking system. In *Proc. Army Science Conf.*, November 2004.

[117] J. Wu, S. Hu, and Y. Wang. Adaptive multifeature visual tracking in a probability-hypothesis-density filtering framework. *Signal Processing*, 93:2915–2926, November 2013.

[118] Y. Wu, G. Hua, and T. Yu. Switching observation models for contour tracking in clutter. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 295–302, June 2003.

[119] F. Xiong, O. I. Camps, and M. Sznaier. Dynamic context for tracking behind occlusions. In *Proc. of European Conf. on Computer Vision*, volume 7576, pages 580–593, 2012.

[120] X. Xu and B. Li. Exploiting motion correlations in 3-D articulated human motion tracking. *IEEE Trans. Image Process.*, 18(6):1292–1303, June 2009.

[121] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823–3831, November 2011.

[122] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Trans. Image Process.*, 19(11):2861–2873, November 2010.

[123] M. Yang. *Context-aware and Attentional Visual Object Tracking*. PhD thesis, Northwestern University, 2008.

[124] M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(7):1195–1209, July 2009.

[125] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Association for Computing Machinery*, 38(4):1–45, December 2006.

[126] J. H. Yoon, D. Y. Kim, and K.-J. Yoon. Visual tracking via adaptive tracker selection with multiple features. In *Proc. of European Conf. on Computer Vision*, volume 7575, pages 28–41, October 2012.

[127] Q. Yu, T. B. Dinh, and G. G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. In *Proc. of European Conf. on Computer Vision*, volume 5303, pages 678–691, October 2008.

[128] Y. Zhai, M. B. Yeary, S. Cheng, and N. Kehtarnavaz. An object-tracking algorithm based on multiple-model particle filtering with state partitioning. *IEEE Trans. Instrum. Meas.*, 58(5):1797–1809, May 2009.

[129] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *Proc. of European Conf. on Computer Vision*, pages 864–877, October 2012.

[130] K. Zhang, L. Zhang, and M.-H. Yang. Fast compressive tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, Doi: 10.1109/TPAMI.2014.2315808.

[131] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 2042–2049, June 2012.

[132] N. Zheng and J. Xue. *Statistical Learning and Pattern Analysis for Image and Video Processing*. Springer Verlag, 2009.

[133] B. Zhong, H. Yao, S. Chen, R. Ji, X. Yuan, S. Liu, and W. Gao. Visual tracking via weakly supervised learning from multiple imperfect oracles. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, pages 1323–1330, June 2010.

[134] S. k. Zhou, R. Chellappa, and B. Moghaddam. Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Trans. Image Process.*, 13(11):1434–1456, November 2004.

[135] Z. Zhu and Q. Ji. Eye gaze tracking under natural head movements. In *Proc. of Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 918–923, June 2005.