Adaptive visual tracking via multiple appearance models and multiple linear searches

Tuan Nguyen, M.Sc.

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

July 2015

Abstract

This research is concerned with adaptive, probabilistic single target tracking algorithms. Though visual tracking methods have seen significant improvement, sustained ability to capture appearance changes and precisely locate the target during complex and unexpected motion remains an open problem. Three novel tracking mechanisms are proposed to address these challenges.

The first is a Particle Filter based Markov Chain Monte Carlo method with sampled appearances (MCMC-SA). This adapts to changes in target appearance by combining two popular generative models: templates and histograms, maintaining multiple instances of each in an appearance pool. The proposed tracker automatically switches between models in response to variations in target appearance, exploiting the strengths of each model component. New models are added, automatically, as necessary.

The second is a Particle Filter based Markov Chain Monte Carlo method with motion direction sampling, from which are derived two variations: motion sampling using a fixed direction of the centroid of all features detected (FMCMC-C) and motion sampling using kernel density estimation of direction (FMCMC-S). This utilises sparse estimates of motion direction derived from local features detected from the target. The tracker captures complex target motions efficiently using only simple components.

The third tracking algorithm considered here combines these above methods to improve target localisation. This tracker comprises multiple motion and appearance models (FMCMC-MM) and automatically selects an appropriate motion and appearance model for tracking. The effectiveness of all three tracking algorithms is demonstrated using a variety of challenging video sequences. Results show that these methods considerably improve tracking performance when compared with state of the art appearance-based tracking frameworks.

Published work

The following research papers have been produced from this thesis:

1. Tuan Nguyen, Tony Pridmore, Adaptive Tracking via Multiple Appearance Models and Multiple Linear Searches, the 10th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP) 2015, Berlin, Germany. (Accepted)

2. Tuan Nguyen, Tony Pridmore, Adaptive Tracking via Multiple Appearance Models and Multiple Linear Searches, British Machine Vision Conference (BMVC) Workshops 2014, Nottingham, United Kingdom. (Presentation)

3. Tuan Nguyen, Tony Pridmore, Tracking Using Multiple Linear Searches and Motion Direction Sampling, 22nd International Conference on Pattern Recognition (ICPR), Stockholm, Sweden, 2014.

Acknowledgements

It is difficult to overstate my gratitude to my supervisor, Professor Tony P. Pridmore. With his enthusiasm, inspiration, and great efforts to explain things clearly and simply, he helped me to understand this difficult topic and suggested some useful material for my thesis. I thank him for the wealth of technical knowledge he has imparted and his kindness in sparing time as well as his humour and perspective, to discuss my submitted work and give valuable comments.

I want to say "thank you" to all of my friends, especially Stefan Mairhofer, Michael Pound and Muhammad Haris Khan, who encouraged me to overcome problems when I was working with this project.

Last but not least, a very heartfelt gratitude goes to my entire family for their constant love, support and encouragement.

Contents

1 Introduction **1**
1.1 Motivation and Approach 1
1.2 Contributions 4
1.3 Thesis Structure 4

2 Background **7**
2.1 Introduction 7
2.2 Tracking Algorithms 8
2.2.1 Deterministic Tracking Algorithms 9
2.2.2 Probabilistic Tracking Algorithms 12
2.2.3 Fused Trackers 22
2.3 Appearance Models 23
2.3.1 Generative Models 26
2.3.2 Discriminative Models 30
2.3.3 Combination Models 34
2.4 Motion Models 36
2.5 Summary 37

3 Tracking with Multiple Generative Models **41**
3.1 Introduction 41
3.2 Proposed Tracking Algorithm 43
3.2.1 Motion Model 44
3.2.2 Appearance Model 45
3.2.3 Sampling Appearance Models 47
3.2.4 Updating a Model 50
3.2.5 Handling Occlusion & Re-detecting the Target 51
3.2.6 Algorithm 52

3.3 Experiments and Results . . . 53
3.3.1 Data . . . 53
3.3.2 Experimental Settings . . . 55
3.3.3 Result . . . 57
3.4 Discussion . . . 66
3.4.1 Appearance change handling . . . 66
3.4.2 Target location improvement . . . 69
3.4.3 Occlusion handling . . . 72
3.4.4 Motion variation handling . . . 72
3.5 Summary . . . 73

4 Tracking with Multiple Linear Searches **75**
4.1 Introduction . . . 75
4.2 Proposed Tracking Algorithm . . . 77
4.2.1 Appearance Model . . . 78
4.2.2 Motion Model . . . 78
4.2.3 Algorithm . . . 83
4.3 Experiments and Results . . . 88
4.3.1 Data . . . 88
4.3.2 Experimental Settings . . . 92
4.3.3 Result . . . 93
4.4 Discussion . . . 102
4.4.1 Smooth Motion Handling . . . 102
4.4.2 Unexpected Motion Handling . . . 103
4.4.3 Distractor Handling . . . 105
4.4.4 Occlusion Handling . . . 108
4.4.5 Appearance Change Handling . . . 109
4.5 Summary . . . 109

5 An Unified Tracking Algorithm **113**
5.1 Introduction . . . 113
5.2 Proposed Tracking Algorithm . . . 114
5.2.1 Appearance Model . . . 114
5.2.2 Motion Model . . . 115
5.2.3 Sampling Appearance & Motion Models . . . 116
5.2.4 Updating Appearance Model . . . 117

5.2.5 Handling Occlusion & Re-detecting the Target . . . 117
5.2.6 Updating Motion Model . . . 117
5.2.7 Algorithm . . . 117
5.3 Experiments and Results . . . 120
5.3.1 Data . . . 120
5.3.2 Experimental Settings . . . 121
5.3.3 Result . . . 122
5.4 Discussion . . . 138
5.4.1 Handling Changes in Appearance . . . 138
5.4.2 Target Location Improvement . . . 141
5.4.3 Motion Variation Handling . . . 146
5.4.4 Distractor Handling . . . 147
5.4.5 Occlusion Handling . . . 148
5.4.6 Scale Change Handling . . . 149
5.5 Summary . . . 149

6 Conclusion and Future Work **151**
6.1 Contributions . . . 151
6.2 Research Outcomes . . . 152
6.3 Future Work . . . 153

A Algorithms **154**
A.1 Kernel Mean-shift tracking . . . 155
A.2 Kalman filter . . . 156

B Tracking Results for Chapter 3 **157**

C Tracking Results for Chapter 4 **171**

D Tracking Results for Chapter 5 **185**

List of Figures

1.1 Overview of the proposed approach. 3

2.1 Particle filtering based methods and their problem. 15
2.2 First order Markov Chain. 16
2.3 Object representation (Yilmaz et al. [2006]). 24
2.4 Multi part representation (Maggio and Cavallaro [2005a]). 25
2.5 Part based representation (Adam et al. [2006]). The target is divided into multiple parts and each part associates with one histogram and votes for the centre location of the target. 25
2.6 Patched based Appearance Model (Kwon and Lee [2013]). 26
2.7 (Enlarged) Histogram & Template. 27
2.8 Dense Optical Flow (using Farnebäck [2003]). 28
2.9 Feature generation and selection process 31
2.10 Online Boosting using feature selection 32
2.11 Tracking with Online AdaBoost 33

3.1 Overview MCMC-SA framework. 44
3.2 Building an appearance model. 46
3.3 Errors at each frame and accumulated errors over time of trackers for the Rolling Ball sequence. 60
3.4 Errors at each frame and accumulated errors over time of trackers for the David2 sequence. 60
3.5 Errors at each frame and accumulated errors over time of trackers for the Doll sequence. 61
3.6 Errors at each frame and accumulated errors over time of trackers for the Girl sequence. 62

3.7 Errors at each frame and accumulated errors over time of trackers for the Boy sequence. . . . 62
3.8 Errors at each frame and accumulated errors over time of trackers for the Animal sequence. . . . 63
3.9 Errors at each frame and accumulated errors over time of trackers for the Jogging sequence. . . . 63
3.10 Errors at each frame and accumulated errors over time of trackers for the Cup sequence. . . . 64
3.11 Errors at each frame and accumulated errors over time of trackers for the Bird2 sequence. . . . 64
3.12 Errors at each frame and accumulated errors over time of trackers for the Jumping sequence. . . . 65
3.13 Tracking results in selected frames of the Rolling Ball sequence. . . . 66
3.14 Tracking results in selected frames of the David2 sequence. . . . 67
3.15 Tracking results of the Animal sequence. . . . 68
3.17 (Enlarged) Girl templates detected during tracking. . . . 68
3.16 Tracking results in selected frames of the Doll sequence. . . . 69
3.18 Tracking results in selected frames of the Girl sequence. . . . 70
3.20 (Enlarged) Boy templates detected during tracking. . . . 70
3.19 Tracking results in selected frames of the Boy sequence. . . . 71
3.21 Tracking results in selected frames of the Bird2 sequence. . . . 71
3.22 (Enlarged) Bird templates detected during tracking of MCMC-SA. . . . 71
3.23 Tracking results in selected frames of the Jogging sequence. . . . 72
3.24 Tracking results in selected frames of the Jumping sequence. . . . 73

4.1 Overview of the FMCMC framework. . . . 78
4.3 Optical flow at Frame #2 of Football sequence. . . . 80
4.4 Features detected at Frame #1 of Football sequence. . . . 81
4.5 Local motion estimates obtained via feature matching. The arrows show the movement of features detected in two consecutive frames. . . . 82
4.6 Kernel Density Estimation of the motion direction distribution at Frame #14 of the PETS2009 sequence. . . . 84
4.7 A fixed motion direction. Lines are parallel. . . . 86
4.8 Sampling motion directions from KDE. . . . 88
4.9 Errors at each frame and accumulated errors over time of trackers for the Data11 sequence. . . . 96

4.10 Errors at each frame and accumulated errors over time of trackers for the Data12 sequence. . . . 96
4.11 Errors at each frame and accumulated errors over time of trackers for the Bouncing1 sequence. . . . 97
4.12 Errors at each frame and accumulated errors over time of trackers for the Bouncing2 sequence. . . . 98
4.13 Errors at each frame and accumulated errors over time of trackers for the Tennis Match sequence. . . . 98
4.14 Errors at each frame and accumulated errors over time of trackers for the Emilio sequence. . . . 99
4.15 Errors at each frame and accumulated errors over time of trackers for the Animal sequence. . . . 99
4.16 Errors at each frame and accumulated errors over time of trackers for the Table Tennis sequence. . . . 100
4.17 Errors at each frame and accumulated errors over time of trackers for the Football sequence. . . . 100
4.18 Errors at each frame and accumulated errors over time of trackers for the PETS09 sequence. . . . 101
4.19 Errors at each frame and accumulated errors over time of trackers for the Girl sequence. . . . 101
4.20 Tracking results in a selected frame of the Data11 sequence. . . . 102
4.21 Local motion directions at selected frames of the Data12 sequence. . . . 103
4.22 Tracking results in selected frames of the Tennis match sequence. . . . 103
4.23 Local motion directions at a selected frame of the Bouncing1 sequence. . . . 104
4.24 Tracking results in selected frames of the Bouncing2 sequence. . . . 104
4.25 Tracking results in a selected frame of the Data12 sequence. . . . 105
4.26 KDE at Frame #22 of the Football sequence. Angles are calculated in radian. . . . 106
4.27 Tracking results in selected frames of the Football sequence. . . . 107
4.28 Tracking results in selected frames of the PETS09 sequence. . . . 108
4.29 Local motions at Frame #56 of the PETS2009 sequence. . . . 108
4.30 Tracking results in selected frames of the Girl sequence. . . . 110

5.1 Overview of the proposed approach. . . . 115
5.2 Errors at each frame and accumulated errors over time of trackers for the Data11 sequence. . . . 126

5.3 Errors at each frame and accumulated errors over time of trackers for the Data12 sequence. . . . 126
5.4 Errors at each frame and accumulated errors over time of trackers for the Bouncing1 sequence. . . . 127
5.5 Errors at each frame and accumulated errors over time of trackers for the Table Tennis sequence. . . . 127
5.6 Errors at each frame and accumulated errors over time of trackers for the Emilio sequence. . . . 128
5.7 Errors at each frame and accumulated errors over time of trackers for the Tennis Match sequence. . . . 128
5.8 Errors at each frame and accumulated errors over time of trackers for the Animal sequence. . . . 129
5.9 Errors at each frame and accumulated errors over time of trackers for the Football sequence. . . . 130
5.10 Errors at each frame and accumulated errors over time of trackers for the PETS09 sequence. . . . 130
5.11 Errors at each frame and accumulated errors over time of trackers for the Bouncing2 sequence. . . . 131
5.12 Errors at each frame and accumulated errors over time of trackers for the Rolling Ball sequence. . . . 131
5.13 Errors at each frame and accumulated errors over time of trackers for the Doll sequence. . . . 132
5.14 Errors at each frame and accumulated errors over time of trackers for the David2 sequence. . . . 132
5.15 Errors at each frame and accumulated errors over time of trackers for the Boy sequence. . . . 133
5.16 Errors at each frame and accumulated errors over time of trackers for the Jogging sequence. . . . 133
5.17 Errors at each frame and accumulated errors over time of trackers for the Jumping sequence. . . . 134
5.18 Errors at each frame and accumulated errors over time of trackers for the Girl sequence. . . . 134
5.19 Errors at each frame and accumulated errors over time of trackers for the Bird2 sequence. . . . 135
5.20 Errors at each frame and accumulated errors over time of trackers for the Cup sequence. . . . 135

5.21 Errors at each frame and accumulated errors over time of trackers for the Hand sequence. 136
5.22 Errors at each frame and accumulated errors over time of trackers for the Tiger1 sequence. 136
5.23 Errors at each frame and accumulated errors over time of trackers for the Freeman1 sequence. 137
5.24 Tracking results in selected frames of the Boy sequence. 138
5.25 Tracking results in selected frames of the Girl sequence. 139
5.26 Tracking results in selected frames of the Bird2 sequence. 139
5.27 Tracking results in the selected frame of the Rolling Ball sequence. 140
5.28 Tracking results in the selected frame of the Bouncing2 sequence. 140
5.29 (Enlarged) Feature movement at the Frame #31 the Jumping sequence. . 142
5.30 Tracking results in selected frames of the Hand sequence. 143
5.31 Feature motion at the Frame #135 of the Hand sequence. 144
5.32 Tracking results in selected frames of the Tiger1 sequence. 145
5.33 (Enlarged) Bird templates detected during tracking of FMCMC-MM. . . . 145
5.34 Tracking results in selected frames of the Doll sequence(Part 1). 146
5.35 Tracking results in the selected frame of the Animal sequence. 146
5.36 Tracking results in the selected frame of the Emilio sequence(Part 1). . . . 147
5.37 Tracking results in the selected frame of the Football sequence(Part 1). . . 147
5.38 Tracking results in selected frames of the David2 sequence. 148
5.39 Tracking results in selected frames of the Jogging sequence. 148

B.1 Tracking results of the Rolling Ball sequence. 158
B.2 Tracking results of the David2 sequence. 159
B.3 Tracking results of the Doll sequence (Part 1). 160
B.4 Tracking results of the Doll sequence (Part 2). 161
B.5 Tracking results of the Girl sequence. 162
B.6 Tracking results of the Boy sequence (Part 1). 163
B.7 Tracking results of the Boy sequence (Part 2). 164
B.8 Tracking results of the Animal sequence. 165
B.9 Tracking results of the Jogging sequence. 166
B.10 Tracking results of the Cup sequence. 167
B.11 Tracking results of the Bird2 sequence. 168
B.12 Tracking results of the Jumping sequence. 169

C.1 Tracking results of the Data11 sequence. 171
C.2 Tracking results of the Data12 sequence. 172
C.3 Tracking results of the Bouncing1 sequence (Part 1). 173
C.4 Tracking results of the Bouncing1 sequence (Part 2). 174
C.5 Tracking results of the Bouncing2 sequence. 175
C.6 Tracking results of the Tennis match sequence. 176
C.7 Tracking results of the Emilio sequence (Part 1). 177
C.8 Tracking results of the Emilio sequence (Part 2). 178
C.9 Tracking results of the Animal sequence. 179
C.10 Tracking results of the Table tennis sequence. 180
C.11 Tracking results of the Football sequence (Part 1). 181
C.12 Tracking results of the Football sequence (Part 2). 182
C.13 Tracking results of the PETS09 sequence. 183
C.14 Tracking results of the Girl sequence. 184

D.1 Tracking results of the Data11 sequence. 185
D.2 Tracking results of the Data12 sequence. 186
D.3 Tracking results of the Bouncing1 sequence (Part 1). 187
D.4 Tracking results of the Bouncing1 sequence (Part 2). 188
D.5 Tracking results of the Table tennis sequence. 189
D.6 Tracking results of the Emilio sequence (Part 1). 190
D.7 Tracking results of the Emilio sequence (Part 2). 191
D.8 Tracking results of the Tennis match sequence. 192
D.9 Tracking results of the Animal sequence. 193
D.10 Tracking results of the Football sequence (Part 1). 194
D.11 Tracking results of the Football sequence (Part 2). 195
D.12 Tracking results of the PETS09 sequence. 196
D.13 Tracking results of the Bouncing2 sequence. 197
D.14 Tracking results of the Rolling Ball sequence. 198
D.15 Tracking results of the Doll sequence (Part 1). 199
D.16 Tracking results of the Doll sequence (Part 2). 200
D.17 Tracking results of the David2 sequence. 201
D.18 Tracking results of the Boy sequence (Part 1). 202
D.19 Tracking results of the Boy sequence (Part 2). 203
D.20 Tracking results of the Jogging sequence. 204
D.21 Tracking results of the Jumping sequence. 205

D.22 Tracking results of the Girl sequence. 206
D.23 Tracking results of the Bird2 sequence. 207
D.24 Tracking results of the Cup sequence. 208
D.25 Tracking results of the Hand sequence (Part 1). 209
D.26 Tracking results of the Hand sequence (Part 2). 210
D.27 Tracking results of the Hand sequence (Part 3). 211
D.28 Tracking results of the Tiger1 sequence. 212
D.29 Tracking results of the Freeman1 sequence. 213

List of Tables

1 Abbreviations have been used in this thesis. xxi

2.1 Common similarity functions. 10

3.1 Testing video sequences and their challenges. 55
3.2 The Correlation Coefficient 56
3.3 The centre location error (in pixels) averaged over all frames of each sequence. All data were presented in corresponding graphs listed below. . . 58
3.4 The overlap ratio between the predicted bounding box and the ground truth bounding box for each testing video sequence. 59

4.1 Testing video sequences and their challenges. 92
4.2 The centre location error (in pixels) averaged over all frames of each sequence. All data were presented in corresponding graphs listed below. . . 94
4.3 The overlap ratio between the predicted bounding box and the ground truth bounding box for each testing video sequence. 94

5.1 Testing video sequences and their challenges. 120
5.2 The centre location error (in pixels) averaged over all frames of each sequence. All data were presented in corresponding graphs listed below. . . 124
5.3 The overlap ratio between the predicted bounding box and the ground truth bounding box for each testing video sequence. 125

A.1 Kalman filter algorithm. 156

Abbreviations

The following table describes the significance of various abbreviations and acronyms used throughout the thesis.

Abbreviation	**Meaning**
MCMC	Markov Chain Monte Carlo
PF	Particle Filter
MS	Mean-shift
SSD	Sum of Squared Differences
NCC	Correlation Coefficience
NCC	Normalised Correlation Coefficience
CCORR	Cross Correlation
NCCORR	Normalised Cross Correlation
MCMC-SA	Markov Chain Monte Carlo with Sampled Appearances
FMCMC-C	Feature based Markov Chain Monte Carlo using a fixed direction of the Centre position
FMCMC-S	Feature based Markov Chain Monte Carlo using Sampled directions
FMCMC-MM	Feature based Markov Chain Monte Carlo using Multiple Models
TT	Template-based tracking
CDF	Cumulative Distribution Function

Table 1: Abbreviations have been used in this thesis.

Chapter 1

Introduction

1.1 Motivation and Approach

Visual tracking is an important computer vision task that has received much attention. It is involved in a wide range of applications from air traffic control (e.g. T. VomÃącka [2010]), surveillance (e.g. people tracking) (e.g. Smith et al. [2005]; Rowe et al. [2010]; Kuo et al. [2010]), wildlife tracking (e.g. Ramanan and Forsyth [2003]; Walther et al. [2004]), motion capture (e.g. Horn and Schunck [1981]; Liu et al. [2013]), military applications (e.g. Berleant and Anderson [2007]), human computer interaction (e.g. Sears and Jacko [2007]; Cipolla and Pentland [1998]) and biological and medical imaging (e.g. Robb [2000]; Dhawan [2011]). Tracking is a time dependent problem. Its aim is to model target appearance and use that model to estimate the state of a moving target, retrieving its trajectory and maintaining its identity through an image sequence. The tracking problem can be formulated as searching for the region with the highest probability of being generated from the appearance model. Key components of a tracker are therefore the search method and the appearance model matching approach used. The search method might be a sliding window (e.g. Grabner and Bischof [2006]) or sampling approach (e.g. Kwon and Lee [2011]) or could use target motion modelling to hypothesise where the target might be (e.g. Isard and Blake [1998]; Grabner et al. [2010]). The target appearance model is typically constructed by extracting features from the first frame. These are then compared to measurements recovered from incoming frames at candidate target positions to estimate the most likely target state. In real world scenarios, targets' appearance can, however, vary over time as a result of illumination changes, pose variations, target and/or camera movement, full or partial occlusions by other targets or by objects in the background, target deformation and complex background clutter. Also, targets'

appearance might be similar, or even identical to, objects in the local background, which may attract the tracker.

To achieve long-term, robust tracking, many researchers have tried to develop richer appearance models. This leads to the use of high-dimensional features to represent the object, increasing computational cost and making the correctness of the model hard to verify. These appearance models are also adapted during the tracking process to learn appearance changes. A fixed appearance model cannot handle target appearance changes well enough to support reliable visual tracking.

Two types of model are used to capture target appearance: generative and discriminative. Generative models try to learn the target's likely appearance, while discriminative models try to include features that separate the target from its local surroundings. They normally either maintain an appearance model or train an online classifier by extracting positive and negative samples around the current target position. This can be considered a self-learning method. A wide variety of appearance models have been proposed and are discussed in Chapter 2. Regardless of approach, adaptive appearance-based trackers face a key problem: the model drift that occurs when background information contaminates the model. The risk and degree of drift increases quickly if the tracked target is not well-located. Several methods have been proposed to deal with the drift problem (discussed in Chapter 2). Despite some success in alleviating drift, these struggle to react quickly enough to large appearance variations. Building an efficient tracking able to cope with these issues is an important and challenging open task.

Trackers focusing on search, on the other hand, aim to enhance target prediction and reduce search space. Rather than performing target detection and data association on each and every frame, these trackers typically model target motion. Many methods have been proposed, and are discussed in Chapter 2. Although they can improve target localisation, these methods typically assume target appearance to be (approximately) constant. It remains difficult to model complex and unexpected target motion.

The key to the model drift problem is to locate the target precisely and carefully control any updates made to the appearance model. Updates should not lose information already learnt and must avoid reflecting abnormal appearance changes. To this end, we develop an online tracker capable of adapting to appearance changes without being too prone to drifting, and able to recover from drift and partial or full occlusion. A number of questions should be considered when constructing a tracking method:

- What appearance model(s) should be used?
- When should additional appearances be learnt?

- How can complex target movement be recovered precisely?

In visual tracking, the best match at time t to appearance observed at time $t-1$ may not be the target, because of changes in visual properties. Thus, to reduce the risk of adaptation drift, additional constraints or supervision of the appearance model are needed. Figure 1.1 gives an overview of the proposed approach. The tracker contains two crucial components: the first learns target appearance changes during tracking and the second utilises features to enhance target prediction via multiple linear searches.

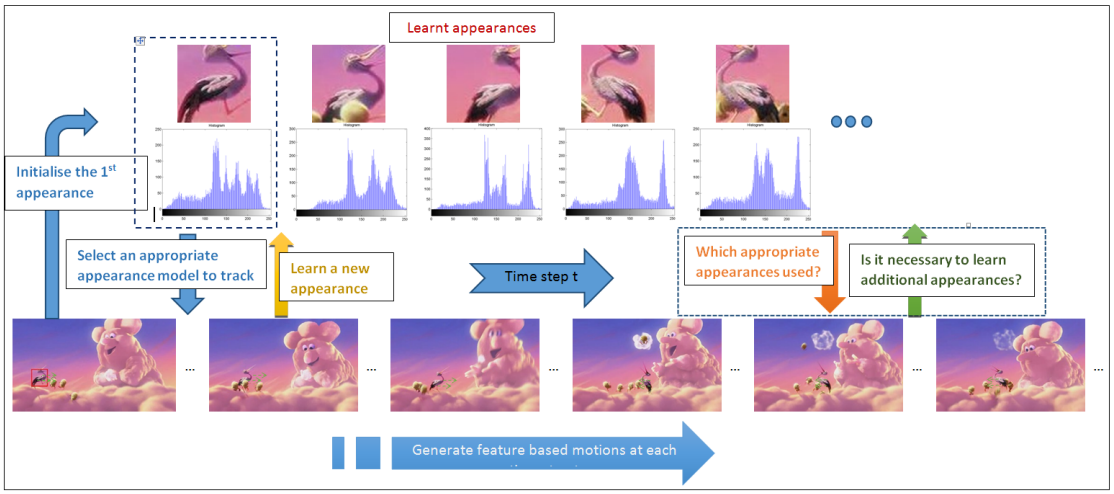


Figure 1.1: Overview of the proposed approach.

This simple yet effective method models appearance using a combination of two popular generative models: templates and histograms. Each new template-histogram pair reflects a new appearance change and is maintained in a pool of appearance models built over different time periods. The tracker automatically switches among models to select the most appropriate model for the current image data. During drifting or occlusion, the tracker can detect the target's presence utilising the appearance model pool and re-initialise the tracking process by selecting a suitable model.

The second component of the tracker aims to handle an unexpected and abrupt target movement. A distribution of likely motion directions is constructed, providing an implicit representation of complex target movements which are difficult to model explicitly. The motion model is constructed by motion directions of local features stored in a feature pool. Each motion direction is formed from low-level target features detected and matched between consecutive frames. The method can enhance target location without using a complex motion model or models, and select an appropriate model with which to search.

Each component is built, individually, into the Particle Filter based Monte Carlo Markov Chain (MCMC) algorithm and thoroughly tested before combining them into a single, unified tracking mechanism.

1.2 Contributions

This thesis introduces a single target tracking algorithm using multiple models (FMCMC-MM), containing two important components: an appearance model and search mechanism, capable of adapting to changes in target appearance and handling motion variations. Given only an initial template representation of the target, the proposed tracker can learn appearance changes in a supervised manner and generate appropriate target motions using the target's local features without knowing the target movement in advance. During tracking, it automatically switches between models in response to variations in target appearance, exploiting the strengths of each model component. New models are added, automatically, as necessary. The effectiveness of the approach is demonstrated using a variety of challenging video sequences.

1.3 Thesis Structure

The remainder of the thesis is organised as follows.

Chapter 2: Background

This chapter presents an introduction to tracking and overview of existing tracking algorithms. It also discusses each component, appearance model and motion model of a tracking framework and describes methods used to model target appearance and construct target motion.

Chapter 3: Tracking with Multiple Generative Models

The first tracking mechanism using sampled appearances (MCMC-SA) is presented and explained. An online appearance model learning mechanism is proposed which utilises matched pairs of generative models: templates and histograms. These models are learnt during tracking and maintained in an appearance pool. The dynamic selection of appearance models for use in tracking is discussed.

Chapter 4: Tracking with Multiple Linear Searches

The second tracking mechanism focusses on capturing target motion. Two variations, motion sampling using a fixed direction of the centroid position of the target features (FMCMC-C) and motion sampling using kernel density estimation of direction (FMCMC-S), are introduced and tested.

Chapter 5: An Unified Tracking Algorithm

The final, unified tracking algorithm with multiple models (FMCMC-MM) combines the appearance model sampling presented in Chapter 3 and motion direction sampling introduced in Chapter 4. Evaluation is conducted using a new data set and all the data sets used during development of the earlier algorithms.

Chapter 6: Conclusion and Future Work

This final chapter reviews contributions made throughout this research, and proposes improvements and directions for future research.

Chapter 2

Background

2.1 Introduction

Visual tracking is a long-standing and challenging problem in computer vision with various practical applications such as surveillance, robotics and human-computer interfaces. It involves several tasks: modelling target appearance, detecting specified targets, applying data association to match the detected target to previously tracked target, recovering the target movement, etc.

Over the past few years, numerous methods addressing a wide range of issues in data association and predictive filtering have been devised for object tracking in image sequences. These vary from simple methods such as frame differencing to complex methods such as fused trackers incorporating target motion models. This chapter presents a general background review of some common tracking algorithms and their variations (Section 2.2).

This thesis focuses on the commonly adopted region tracking approach. A region tracker defines an image region that contains the target of interest, with the boundary often being a bounding box or a simple polygon. Then, in the next image of a sequence, it looks for a corresponding region of the image using a similarity measure to decide on the best matching region. The regions can be defined manually (e.g. annotated with bounding boxes or ellipses by a human) or automatically (e.g. specified by object detectors such as human (Dalal and Triggs [2005]) or face detectors (Viola and Jones [2001])).

In general, there are two important components for a tracker: the appearance model and the motion model. In order to track a target, its appearance should be well modelled, since if the assumptions made by the appearance model are incorrect or inaccurate, the

tracker may fail. Image data provides many features which can be exploited to define target appearance models e.g. colour, contours, texture, corners or combination of these features. The use of features varies between different tracking approaches. A single tracker can use an appearance model which is defined by one feature or many. In other cases, multiple trackers can be employed, each using a different feature to track one target, with these trackers interacting at a later stage to produce a final estimation.

Tracking performance depends not only on the appearance model, but also on the environment surrounding the target of interest. During tracking, the target might change its pose or produce unexpected and fast movements. Moreover, illumination changes, the presence of clutter, i.e. unrelated objects similar to the target, partial or full occlusion may play a major role in tracker failure. Ideally, trackers should be able to adapt their models to deal with these problems. A common approach to the construction of an adaptive tracker is to update the target appearance model. A key problem while updating the appearance model is model drift: the background information contaminates the appearance model. This problem occurs when the tracker estimates the target location incorrectly and tries to update the appearance model. Update methods and mechanisms proposed to deal with the drifting problem are discussed in Section 2.3.

Some tracking approaches emphasise motion modelling, incorporating one or more motion models to improve the estimation process. Without motion models, to constrain their search, trackers must examine a larger area of the image. One approach is to use a sliding window to search either the whole image or an area around the target's previous location. Modelling target movements, however, is not easy, as targets can exhibit complex and unpredictable target motion. Section 2.4 discusses common motion models used in visual tracking.

The literature on visual tracking is large, and expanding quickly. A complete and detailed review is impossible. This chapter highlights the more common tracking approaches, and appearance models most closely related to the work reported here.

2.2 Tracking Algorithms

Visual tracking algorithms can be roughly classified into two categories: deterministic methods and stochastic methods.

2.2.1 Deterministic Tracking Algorithms

Deterministic methods typically track the target by performing an iterative search for the local maximum (or minimum) of a similarity (or cost) function. There are two popular approaches: template based tracking and kernel mean-shift tracking.

Template based tracking

Template-based tracking estimates target location by using similarity functions to search the incoming image for the patch best matched to a fixed template image T, usually extracted from the first frame of the sequence, and describing the target appearance. The patch with the highest similarity score is treated as the new target location. The simplest but least efficient search strategy is exhaustive search or a sliding window technique. Basic steps that might be performed during template-based tracking algorithms are described in Algorithm 1.

Algorithm 1 Basic steps in template based tracking algorithms (Adapted from Cannons [2008]).

1. Initialise the template on the target region in the first frame.
2. Predict where the target will appear in the subsequent frame (optional).
3. Load the next frame
4. Match the template to image regions centred on the predicted target position and within a surrounding neighbourhood search region by methods described in the Table 2.1.
5. Select the location that provides the highest matching score as the current target centre.
6. Update the target template (optional).
7. Repeat until the end of the image sequences.

To compare a template with an image patch, most common similarity functions have been used in the literature such as Sum of Squared Difference (SSD) or Normalised SSD (Kanade et al. [1995]; Hager and Belhumeur [1998]; Nickels and Hutchinson [2002]; Baker and Matthews [2004]; Okuma et al. [2004]); Correlation Coefficience (CCOEFF) or Normalised CCOEFF (Derpanis et al. [2006]); Cross Correlation (CCORR) or Normalised CCORR (Brown et al. [2003]), etc. described in the Table 2.1 where I is the given image, x, y are pixel locations in the image, T is the testing template and x', y' are pixel locations

in the testing template. In practice, to handle intensity changes, normalised matching methods have been widely used.

Method	**Definition**
Sum of Squared Differences (SSD)	$R(x,y) = \sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2$ (2.1)
Normalised SSD	$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}}$ (2.2)
Correlation Coefficience (CCOEFF)	$R(x,y) = \sum_{x',y'}(T'(x',y').I'(x+x',y+y'))$ (2.3) $T'(x',y') = T(x',y') - \frac{1}{w.h}.\sum_{x'',y''} T(x'',y'')$ (2.4) $I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{w.h}.\sum_{x'',y''} I(x'',y'')$ (2.5)
Normalised CCOEFF	$R(x,y) = \frac{\sum_{x',y'}(T'(x',y').I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 . \sum_{x',y'} I'(x+x',y+y')^2}}$ (2.6)
Cross Correlation (CCORR)	$R(x,y) = \sum_{x',y'}(T(x',y').I(x+x',y+y'))$ (2.7)
Normalised CCORR	$R(x,y) = \frac{\sum_{x',y'}(T(x',y').I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}}$ (2.8)

Table 2.1: Common similarity functions.

Template-based tracking algorithms using similarity functions tend to lose the target of interest when the target appearance changes or the target is occluded, though they are less effected by illumination changes if using normalised similarity measures. The advantage of the often large search area considered is that they can often recover the target after occlusion if its appearance remains roughly constant.

A more advanced approach is to use gradient descent techniques. Given the template T, take all pixels x from the template and warp them using the function $W(x;p)$ parameterised in terms of parameters p, a motion parameter vector, to the input image. Assign the pixel value of the input image at the warped location to the template image. The Lucas-Kanade algorithm is summarised in Algorithm 2.

Algorithm 2 The Lucas-Kanade algorithm (adapted from Baker and Matthews [2004]).
Repeat

1. Warp I with a warp function $W(x;p)$ to compute $I(W(x;p))$.
2. Compute the error image $T(x) - I(W(x;p))$
3. Warp the gradient $\bigtriangledown I$ with W(x;p)
4. Evaluate the Jacobian $\frac{\partial W}{\partial p}$ at $(x;p)$.
5. Compute the steepest descent image $\bigtriangledown I \frac{\partial W}{\partial p}$.
6. Compute the Hessian matrix $H = \sum_x [\bigtriangledown I \frac{\partial W}{\partial p}]^T [\bigtriangledown I \frac{\partial W}{\partial p}]$.
7. Compute $\triangle p = H^{-1} \sum_x [\bigtriangledown I \frac{\partial W}{\partial p}]^T [T(x) - I(W(x;p))]$.
8. Update the parameters $p \leftarrow p + \triangle p$

Until $\triangle p < \epsilon$ (i.e. ϵ is a small value to stop the iteration.)

The Lucas-Kanade algorithm assumes that:

1. only the object to be tracked appears in the template image.
2. the entire template is visible in the input image, i.e. there is no occlusion.
3. the image intensity of the object is always the same (brightness constancy).

These assumptions are not always true in real world video sequences.

Kernel Mean-shift tracking

Not all deterministic methods employ templates; colour histograms are a popular choice and have been used to good effect in kernel mean-shift tracking (Comaniciu et al. [2003]; Collins [2003]). The idea of a basic colour histogram is to consider each colour in turn and count the number of pixels across the target that are of this colour. Each bin within a colour histogram can be constructed using:

$$p_y^{(u)} = C \sum_{i=1}^{n} \delta \left(b(x_i) - u \right). \tag{2.9}$$

where x_i is a single target pixel, C is a constant to ensure that the histogram bins sum to one, u is a particular histogram bin, n is the number of pixels in the region, $b(x_i)$ is

a function to map the colour at x_i to the histogram bin u, and δ is the Kronecker delta function.

In mean-shift, however, the colour histogram is modified to include a kernel weighting function. The rationale behind the kernel function is to weight pixels depending on their spatial location within the tracking window. Pixels at the centre of the window are more likely to belong to the target and so are weighted highly. On the other hand, pixels near the border of the tracking window have a higher chance of being part of the background. They, therefore, receive lower weights. Moreover, the tracking window does not perfectly adhere to the outline of the target, so that pixels near the border of the tracking window do not greatly affect the histogram representing the candidate target. The Epanechnikov kernel $k(r)$ is widely used in mean-shift tracking (Comaniciu et al. [2003]).

The colour histogram $p_y = \{p_y^{(u)}\}_{u=1\ldots m}$ at location y is

$$p_y^{(u)} = C \sum_{i=1}^{n} k\left(\left\|\frac{y - x_i}{h}\right\|^2\right) \delta\left(b(x_i) - u\right). \tag{2.10}$$

where h is the bandwidth of the kernel and the normalisation factor

$$C = \frac{1}{\sum_1^n k\left(\|\frac{y-x_i}{h}\|^2\right)}. \tag{2.11}$$

Similar to simple gradient ascent, the mean-shift algorithm seeks the modes of a distribution using an estimate of the function's gradient. The modes of a distribution are located by iteratively computing the mean-shift vector and translating the centre of the kernel to the specified location. Starting from the target's position in the previous frame, these steps are repeated until convergence has been reached or a fixed number of iterations have been executed. Algorithm 15 presents a summary of the kernel mean-shift tracking algorithm of Comaniciu et al. [2003].

Kernel mean-shift tracking can easily be distracted from its target by background clutter, causing the search to proceed in the wrong direction. Also, it is a hill climbing search, i.e. it will stop the search when it finds a peak, and so can be trapped by local maxima. It is not easy for the tracker to recover from this error.

2.2.2 Probabilistic Tracking Algorithms

Stochastic methods have gained much attention because they can account for uncertainty and ambiguity in a principled way. They use a state space to model the underlying dynamics of the tracking process, and transform the visual tracking task to a Bayesian infer-

ence problem into which a number of hypotheses are generated to estimate and propagate the posterior distribution of the state. Compared with their deterministic counterparts, stochastic methods usually perform more robustly, but suffer a heavy computational load due to the large number of hypotheses involved, especially in high-dimensional state spaces.

In a linear-Gaussian model with linear measurement, there is always only one mode in the posterior probability density function (pdf), the Kalman filter (Kalman [1960]) propagates and updates the mean and covariance of the distribution. For nonlinear or non-Gaussian problems, it is impossible to evaluate the distributions analytically and many algorithms have been proposed to approximate them. One route to a solution is the sequential importance sampling (SIS) algorithm, a Monte Carlo method commonly known as bootstrap filtering (Gordon et al. [1993]), the Condensation algorithm (MacCormick and Blake [1999]), or particle filtering (Carpenter et al. [1999]).

Bayesian tracking is commonly defined in terms of a process model f and a measurement model h:

$$x_t = f_t(x_{t-1}, u_{t-1}, w_{t-1}). \tag{2.12}$$

$$z_t = h_t(x_t, v_t). \tag{2.13}$$

The symbol x_t denotes the system state at time t and z_t denotes the observation made at time t. Both models are in general non-linear and time-dependent. The random variables w_{t-1}, v_t represent the process and measurement noise and u_{t-1} is the control input.

Kalman filter

The Kalman filter (Kalman [1960], Welch and Bishop [1995]) provides a means of optimally estimating the hidden state of a system by analysing observable measurements. The Kalman filter is only optimal when a certain set of assumptions hold true. Once these assumptions are violated, the estimates provided by the Kalman filter may no longer be optimal.

The Kalman filter assumes that the posterior density at every time step is Gaussian and can be characterised by a mean and covariance. Let x_t be the state of the system at time t, and z_t be the measurements. They are presented as transition and measurement model equations below:

$$x_t = Ax_{t-1} + Bu_{t-1} + w_{t-1}. \tag{2.14}$$

$$z_t = Hx_t + v_t. \tag{2.15}$$

The random variables w_t, v_t represent the process and measurement noise. They are assumed to be independent each other and drawn from normal probability distributions. Matrix A specifies the relation between target state at times $t-1$ and t, B relates the optional control input to the state x_t, and H relates the state to the measurement z_t. In visual tracking, the control input u is usually omitted.

The Kalman filter is a process of estimation using a form of feedback control (Welch and Bishop [1995]): the filter estimates the process state and then obtains feedback in the form of (noisy) measurements. Two stages are defined: time update and measurement update. During time update, the current state and error covariance estimates are projected forward to obtain a priori estimates for the next time step. In the measurement update, a new measurement is incorporated into a priori estimate to obtain an improved a posteriori estimate. A complete Kalman filter is presented in the Table A.1.

Kalman filtering does not work well given non-linear equations. The Extended Kalman filter (EKF) (e.g. Bianchi and I.Tinnirello [2003]) allows the prediction and correction models to be non-linear.

Particle filtering

In visual tracking, assuming the target distribution is a unimodal Gaussian, as required by the Kalman filter, is not feasible. Tracking in cluttered or complex environments often makes the distribution multi-modal. Particle filtering, another state prediction method, is a technique for implementing recursive Bayesian filters by Monte Carlo sampling. A recursive filtering approach means that received data can be processed sequentially rather than as a batch, so that it is not necessary to store the complete data set nor to reprocess existing data if a new measurement becomes available (Arulampalam et al. [2002]). The key idea is to use a set of random particles with associated weights to represent the posterior density (pdf). Particle filtering has become a tremendously popular tool with which to perform visual tracking incorporating nonlinearity, which is the major restriction of the Kalman filter. Figure 2.1 shows some common Particle filtering based methods and their drawbacks. Each method is briefly discussed below.

From a Bayesian perspective, given a series of observations, the aim of tracking is to find the most likely state of a target at each time point. The state at time t is given by $x_t = \{x, y\}$ where (x, y) is the target location, and the observations up to time t, $z_{1:t}$. The posterior pdf $p(x_t|z_{1:t})$ estimated in this approach consists of two essential stages: prediction and update. The update operation uses the latest measurement to modify the

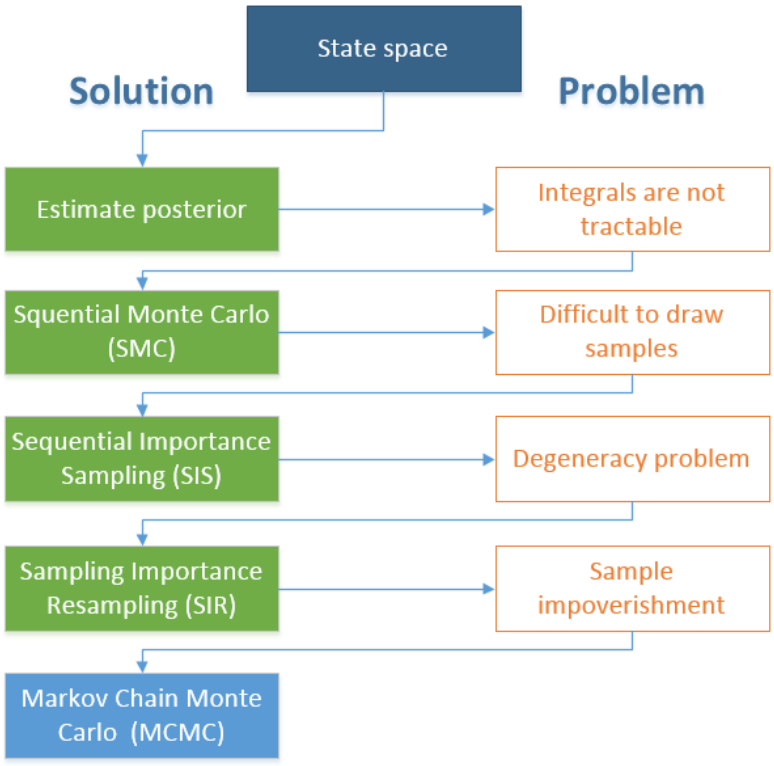


Figure 2.1: Particle filtering based methods and their problem.

prediction pdf. The assumption is that the initial pdf $p(x_0|z_0) \equiv p(x_0)$ of the state.

The prediction stage uses the system model Equation 2.12 to project the state pdf forward from one measurement time to the next via the Chapman-Kolmogorov equation 2.16. Since the state is usually subject to unknown disturbances (modelled as random noise), prediction generally translates, deforms, and spreads the state pdf.

$$p(x_t|x_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1}. \tag{2.16}$$

The current state can be estimated, given that the previous state and all previous observations are known, using the prediction equation. If assuming a Markov process of order one, it allows us to consider the conditional density of the novel state as an integral over its conditional density given the previous state.

A Markov process of order one (as in Figure 2.2) is used in Equation 2.16, the current state of the target depends on the immediately previous state, i.e. $p(x_t|x_{t-1}, z_{1:t-1}) = p(x_t|x_{t-1})$. The probabilistic model of the state evolution $p(x_t|x_{t-1})$ is defined by the system equation 2.12 and the known statistics of w_{t-1}, a Gaussian noise.

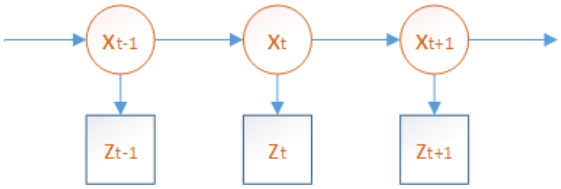


Figure 2.2: First order Markov Chain.

At time t, when a measurement z_t is available, the prior is updated via Bayes' rule:

$$p(x_t|z_{1:t}) = \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})}. \tag{2.17}$$

where the normalising constant $p(z_t|z_{1:t-1}) = \int p(z_t|x_t)p(x_t|z_{1:t-1})dx_t$ depends on the likelihood function $p(z_t|x_t)$ specified by the measurement model (Equation 2.13). In this case, the measurements depend only on the current state. In the update step (Equation 2.17), the measurement z_t is used to modify the prior density to obtain the required posterior density of the current state.

The recursive relations Equation 2.16 and Equation 2.17 form the basis for the optimal Bayesian solution. This recursive propagation of the posterior density is only a conceptual solution. It cannot be determined analytically.

Sequential Monte Carlo (SMC):
SMC methods are very popular, e.g., Gustafsson et al. [2002], Hue et al. [2000], and Thrun [2002], and used to make approximations in many fields of research. This Monte Carlo method is known as bootstrap filtering (Gordon et al. [1993]), the Condensation algorithm (MacCormick and Blake [1999]), or particle filtering (Carpenter et al. [1999]). As the number of samples becomes very large, this MC characterization becomes an equivalent representation to the usual functional description of the posterior pdf, and approaches the optimal Bayesian estimate. Their main advantage is their ability to approximate complex high dimensional densities, i.e they can be used to approximate states of non-linear dynamical models and non-Gaussian noise.

Let $\{x^i_{0:t}, w^i_t\}$ denote a random measure that characterises the posterior pdf $p(x_{0:t}|z_{1:t})$, where $\{x^i_{0:t}\}^{N_s}_{i=0}$ is a set of support points with associated weights $\{w^i_t\}^{N_s}_{i=0}$ and $x_{0:t} = \{x_j\}^t_{j=0}$ is the set of all states up to time t. The weights are normalised such that $\sum^{N_s}_{i=0} w^i_t = 1$. Then, the posterior density at t can be approximated as

$$p(x_{0:t}|z_{1:t}) \approx \sum_{i=1}^{N_s} w^i_t \delta(x_{0:t} - x^i_{0:t}) \tag{2.18}$$

We therefore have a discrete weighted approximation to the true posterior, $p(x_{0:t}|z_{1:t})$.

Sequential Importance Sampling (SIS):
Suppose $p(x) \propto \pi(x)$ is a probability density from which it is difficult to draw samples but for which $\pi(x)$ can be evaluated. In addition, let $x^i \sim q(x), i = 1, .., N_s$ be samples that are easily generated from a proposal $q(.)$ called an importance density. Then, a weighted approximation to the density $p(.)$ is given by

$$p(x) \approx \sum_{i=1}^{N_s} w^i \delta(x - x^i). \tag{2.19}$$

where

$$w^i = \frac{\pi(x^i)}{q(x^i)}. \tag{2.20}$$

is the normalised weight of the ith particle. Therefore, if the sample $x^i_{0:t}$ were drawn from an importance density $q(x_{0:t}|z_{1:t})$ then the weights in Equation 2.18 are defined by Equation 2.20 to be

$$w^i \propto \frac{p(x^i_{0:t}|z_{1:t})}{q(x^i_{0:t}|z_{1:t})}. \tag{2.21}$$

If the importance density is chosen to factorise such that

$$q(x_{0:t}|z_{1:t}) = q(x_t|x_{0:t}, z_{1:t})q(x_{0:t}|z_{1:t-1}). \tag{2.22}$$

then samples $x^i_{0:t} \sim q(x_{0:t}|z_{1:t})$ can be obtained by augmenting each of the existing samples $x^i_{0:t-1} \sim q(x_{0:t-1}|z_{1:t-1})$ with the new state $x^i_t \sim q(x_t|x_{0:t-1}, z_{1:t})$ and $p(x_{0:t}|z_{1:t})$ can be derived as

$$p(x_{0:t}|z_{1:t}) = \frac{p(z_t|x_t)p(x_t|x_{t-1})}{p(z_t|z_{1:t-1})}p(x_{0:t-1}|z_{1:t-1}) \tag{2.23}$$

$$\propto p(z_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|z_{1:t-1}). \tag{2.24}$$

The weight update can be shown to be

$$w^i_t \propto \frac{p(z_t|x^i_t)p(x^i_t|x^i_{t-1})p(x^i_{0:t-1}|z_{1:t-1})}{q(x^i_t|x^i_{0:t-1}, z_{1:t})q(x^i_{0:t-1}|z_{1:t-1})} \tag{2.25}$$

$$= w^i_{t-1}\frac{p(z_t|x^i_t)p(x^i_t|x^i_{t-1})}{q(x^i_t|x^i_{0:t-1}, z_{1:t})}. \tag{2.26}$$

Furthermore, if $q(x_t|x_{0:t-1}, z_{1:t}) = q(x_t|x_{t-1}, z_t)$, then the importance density becomes only dependent on x_{t-1} and z_t. This is particularly useful in the common case when only

a filtered estimate of $p(x_t|z_{1:t})$ is required at each time step. Then, only x_t^i needed to be stored and $x_{0:t-1}^i$ and observations $z_{1:t-1}$ can be discarded. The modified weight is then

$$w_t^i = w_{t-1}^i \frac{p(z_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, z_t)}. \tag{2.27}$$

and the posterior filtered density $p(x_t|z_t)$ can be approximated as

$$p(x_t|z_t) \approx \sum_{i=1}^{N_s} w_t^i \delta(x_t - x_t^i). \tag{2.28}$$

Algorithm 3 The Sequential Important Sampling algorithm (adapted from Arulampalam et al. [2002]).

Given the $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^{N_s}$

1. For $i = 1{:}N_s$
   (a) Draw $x_t^i \sim q(x_t|x_{t-1}^i, z_t)$.
   (b) Assign the particle a weight w_t^i according to Equation 2.27.
2. End For

A common problem with the SIS particle filter is the degeneracy problem in which, after a few iterations, a few particles will have negligible weight, and the variance of the importance weights can only increase over time (showed in Doucet et al. [2000]). One method to reduce the effects of degeneracy is to use resampling.

Sampling Importance Resampling (SIR)

The basic idea of resampling is to eliminate particles that have small weights and concentrate on particles with large weights. The resampling step involves generating a new set $\{x_t^{i*}\}_{i=1}^{N_s}$ by resampling (with replacement) N_s times from an approximate discrete representation of $p(x_t|z_{1:t})$ given by

$$p(x_t|z_{1:t}) \approx \sum_{i=1}^{N_s} w_t^i \delta(x_t - x_t^i). \tag{2.29}$$

so that $p(x_t^{i*} = x_t^j) = w_t^j$. The resulting sample is in fact an i.i.d. sample from the discrete density Equation 2.29; therefore, the weights are now reset to $w_t^i = \frac{1}{N_s}$. Systematic resampling (Kitagawa [1996]) is preferred and described in Algorithm 4, where $U[a, b]$ is the uniform distribution on the interval $[a, b]$. The index of the parent of each resampled

particle x_t^* is stored and denoted as i^j.

Algorithm 4 The Sampling Importance Resampling algorithm (adapted from Arulampalam et al. [2002]).

Given the $\{x_t^i, w_t^i\}_{i=1}^{N_s}$

1. Initialise Cumulative Distribution Function (CDF): $c_1 = 0$
2. For $i = 2{:}N_s$
   - Construct CDF: $c_i = c_{i-1} + w_t^i$.
3. End For
4. Start at the bottom of the CDF: $i = 1$.
5. Draw a starting point $u_1 \sim U[0, \frac{1}{N_s}]$.
6. For $j = 1 : N_s$
   (a) Move along the CDF: $u_j = u_1 + N_s^{-1}(j-1)$.
   (b) $i = 1$
   (c) While $u_j > c_i$
      - $i = i + 1$
   (d) End While
   (e) Assign sample: $x_t^{j*} = x_t^i$.
   (f) Assign weight: $w_t^j = \frac{1}{N_s}$.
   (g) Assign parent: $i^j = i$.
7. End For

The SIR algorithm can be easily derived from the SIS algorithm by an appropriate choice of

1. The importance density $q(x_t|x_{t-1}^i, z_{1:t})$, which is chosen to be the prior density $p(x_t|x_{t-1}^i)$
2. The resampling step to be applied at every time step.

With the above choices made, the particle's weight is given by

$$w_t^i \sim w_{t-1}^i p(z_t|x_t^i). \tag{2.30}$$

where the likelihood $p(z_t|x_t^i)$ is available. Because the resampling is applied at every time step, we have $w_{t-1}^i = \frac{1}{N_s}, \forall i$, and the weight becomes

$$w_t^i \approx p(z_t|x_t^i). \tag{2.31}$$

The weights given by the proportionality in Equation 2.31 are normalised before the resampling stage. The SIR algorithm is given in Algorithm 5.

Algorithm 5 The Sequential Important Resampling algorithm (adapted from Arulampalam et al. [2002]).

Given the $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^{N_s}$

1. For $i = 1{:}N_s$
   (a) Draw $x_t^i \sim p(x_t|x_{t-1})$
   (b) Calculate weight $w_t^i \sim p(z_t|x_t^i)$.
2. End For
3. Calculate total weight: $t = \sum_{i=1}^{N_s} w_t^i$
4. For $i = 1{:}N_s$
   (a) Normalise: $w_t^i = \frac{w_t^i}{t}$.
5. End For
6. Resample using Algorithm 4 to obtain new $\{x_t^i, w_t^i\}_{i=1}^{N_s}$

As Arulampalam et al. [2002] state, though the resampling step reduces the effects of the degeneracy problem occurring in the SIS, it introduces other practical problems. First, it limits the opportunity to parallelise, since all the particles must be combined. Second, particles with high weights may be selected many times. This leads to a loss of diversity among the particles as the resultant sample will contain many repeated points. This problem, which is known as sample impoverishment, is severe in the case of small process noise. A Markov Chain Monte Carlo (MCMC) approach is introduced to solve this problem.

Markov Chain Monte Carlo (MCMC)

The Markov Chain Monte Carlo based particle filter defines a Markov Chain over the state space X, such that the stationary distribution $\pi(x)$ of the chain is equal to the sought pos-

terior $p(X_t|Z_{1:t})$. One way to simulate the MCMC chain is via the Metropolis-Hastings (MH) algorithm (Hastings [1970]) constructing a set of unweighted samples. Due to the limitations of importance sampling in high dimensional state spaces, especially relevant when seeking multiple targets, the Markov Chain Monte Carlo method is commonly applied in visual tracking (Khan et al. [2005]). Though MCMC has been designed to deal with multiple objects, it can be used to track a single target because MCMC considers only one target at each iteration of the MH algorithm. To simplify, the interaction model which prevents trackers from different objects converging onto a single object is ignored.

A candidate particle X'_t, sampled from the current sample X_t using a proposal $Q(X'_t; X_t)$ is accepted if the acceptance ratio (in Equation 2.32) exceeds 1.

$$a = \frac{P(X'_t|Z_t)Q(X_t; X'_t)}{P(X_t|Z_t)Q(X'_t; X_t)}. \tag{2.32}$$

The proposal density $Q(X'_t; X_t)$ for one target is typically designed as a zero-mean normal distribution and the observation likelihood is defined individually for each target. In addition, the acceptance ratio is applied for one target at one time. Therefore, the acceptance ratio in Equation 2.32 is simplified to a ratio of observation likelihood of the proposed state X'_t and the previous state X_t as in Equation 2.33. See (Khan et al. [2005]) for more details.

$$a = \frac{P(Z_t|X'_t)}{P(Z_t|X_t)}. \tag{2.33}$$

A maximum a posterior (MAP) has typically been used to find a particle most likely the target over N samples at each time t (Khan et al. [2005]). Algorithm 6 summarises the MCMC-based Particle filter. Note that when tracking one target, the state X_t contains only a configuration for that target at time t.

A burn-in period B is typically used to discard any bias introduced by the starting position. During and after the burn-in period, the algorithm continues by performing a search of the state space, by changing a single target's parameters at a time. These changes are compared with the previous state, to ensure that the newer joint state represents an improvement over the previous. Improved states are likely to be accepted (after a thinning period M which is used to select one particle out of M particles generated), and added to the sets of states in the Markov chain. This search approach allows the algorithm to search a complex multi-dimensional joint state space, while remaining computationally efficient. A thorough description of the original MCMC tracker can be found in (Khan et al. [2005]), along with experimental results detailing its accuracy and

Algorithm 6 Particle filter based MCMC (adopted from Khan et al. [2005]).

1. Initialise the MCMC sampler: randomly pick a sample $X_{t-1}^{(r)}$ and move the target using a motion model. The result is the initial state of the X_t Markov chain.
2. MCMC sampling step (Metropolis-Hasting): Repeat $(B + MN)$ times, where B is the length of the burn-in period and M is the length of the thinning interval:
   (a) Sample from the proposal density: propose a new state X'_t for the target, by sampling from the proposal density $Q(X'_t; X_t)$.
   (b) Compute the acceptance ratio a.
   (c) If $a \geq 1$, then accept X'_t: set the target in X_t to X'_t and update the cached likelihood. Otherwise, accept with probability a. If rejected, leave X_t unchanged.
3. As an approximation for the current posterior $P(X_t|Z_{1:t})$, we return the new sample set $\{X_t^{(i)}\}_{i=1}^{N}$, obtained by storing every M_{th} sample after the initial B burn-in iterations above.

robustness.

2.2.3 Fused Trackers

A fused tracker is formed when two or more existing tracking algorithms are combined to achieve a hopefully superior tracking algorithm. Combing two or more in an efficient manner may exploit their strengths while reducing their drawbacks. For instance, mean-shift maintains only one hypothesis, and usually fails when the distance the target object moves is greater than the allowed bandwidth. It, however, hill climbs efficiently. Particle filters, on the other hand, need many particles to cover a given space thoroughly as the particles are scattered randomly (according to its motion model(s)).

Maggio and Cavallaro [2005a] and Shan et al. [2004] combined Particle Filter and mean-shift tracking algorithms, allowing each particle generated by the Particle Filter to be clustered towards the local maxima by mean-shift. Multiple hypotheses are maintained by projecting a number of particles randomly around the prior position, and then these particles hill climb towards the best target centre. This hybrid tracker requires a smaller number of particles to carry out tracking successfully. The hybrid tracker shows performance advantages over both Particle filter and Mean Shift tracking. However, as particles are randomly projected, we still need a good number to cover a given search space. Running N mean-shift trackers, where N is the number of particles in the system,

also makes the system computationally expensive. Furthermore, many of the particles coalesce during the mean-shift phase, moving to the same hypothesis and making the representation redundant. If Condensation tends towards an incorrect local maximum, mean-shift will accelerate the process.

Recently, Kwon and Lee (Kwon and Lee [2010]; K. and Lee [2013]) proposed a tracker sampling method to handle appearance and motion changes by generating multiple trackers and using the best state among these trackers to estimate new target locations. Their framework performed appearance sampling by utilising image data from the last five frames. Features of the target in these frames are extracted and the appearance model is constructed using Sparse principal component analysis (SPCA). The idea is to select the features that best separate the target from the local background. A similar idea was used in Collins et al. [2005] but this work used a fixed set of (49) features. Santner et al. [2010] proposed a tracker (PROST) which combines three simple trackers: Template Matching (using NCC), Meanshift Optical flow (FLOW) (Werlberger et al. [2009]) and Online Random Forest (ORF) (Saffari et al. [2009]). FLOW is fast and accurately adapts to appearance changes, so it is overruled by ORF if it is not overlapping and ORF has a confidence above a given threshold. ORF is updated only if it overlaps with NCC or FLOW. The challenge raised by these works is how to ensure agreement among trackers.

2.3 Appearance Models

Before the tracking process is invoked, the likely appearance of the target of interest must be modelled. One of the key factors in tracking is how to choose an appropriate method to represent the object. A good representation should be robust to object rotation, scale variation, partial occlusion, etc. Object representation methods should satisfy two properties: discriminability and computational efficiency. Several ways to represent the target are reported in the literature. Representations are usually chosen to suit a specific application domain. Yilmaz et al. [2006] provide the following synthesis:

- Points: The object can be represented by a centre point (Figure 2.3(a)) (Veenman et al. [2001]) or by a set of points (Figure 2.3(b)) (Serby et al. [2004]). In general, the point representation is suitable for tracking objects that occupy small regions of the image.

- Primitive geometric shapes: A rectangle or ellipse is used to represent an object shape (Figure 2.3(c), (d)) (Comaniciu et al. [2003]). This type can be used for simple rigid and nonrigid objects.

- Object silhouette and contour: Contour representation defines the boundary of an object (Figure 2.3(g), (h)). The region inside the contour is called the silhouette of the object (see Figure 2.3(i)). Silhouette and contour representations are suitable for tracking complex nonrigid shapes (Yilmaz et al. [2004]).

- Articulated shape models: Articulated objects (e.g. the human body) are composed of body parts that are held together with joints. The relationship between the parts are governed by kinematic motion models, for example, joint angle, etc. In order to represent an articulated object, one can model the constituent parts using cylinders or ellipses as shown in Figure 2.3(e).

- Skeletal models: This model is commonly used as a shape representation for recognising objects (Ali and Aggarwal [2001]). Skeleton representation can be used to model both articulated and rigid objects (see Figure 2.3(f)).

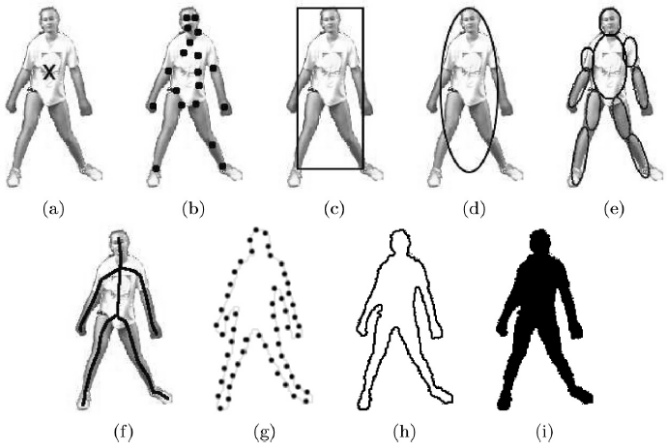


Figure 2.3: Object representation (Yilmaz et al. [2006]).

Part- or patch-based methods have been used in several works (e.g Maggio and Cavallaro [2005b]; Adam et al. [2006]; Kwon and Lee [2013])). In Maggio and Cavallaro [2005b], seven parts (Figure 2.4) are used to represent an object: a whole (2.4(a)), four parts (2.4(b)) designed to help recognise rotations, and a size-sensitive division into two further parts (2.4(c)). The complete representation is shown in (2.4(d)). In Adam et al. [2006], the target is represented by multiple parts as shown in Figure 2.5. To handle partial occlusions, each part votes for the target location. Kwon and Lee [2013] generate and select patches (Figure 2.6) by calculating scores of the Hessian Matrix for each pixel. Patches do not overlap and their sizes are random. This method is designed to handle

drastic geometric appearance changes. Target location is determined by combining the vote maps of all patches.

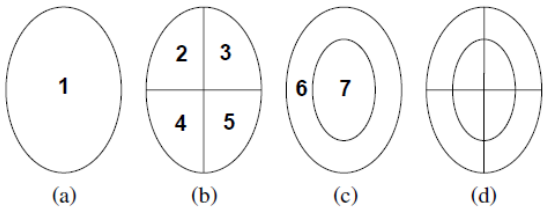


Figure 2.4: Multi part representation (Maggio and Cavallaro [2005a]).

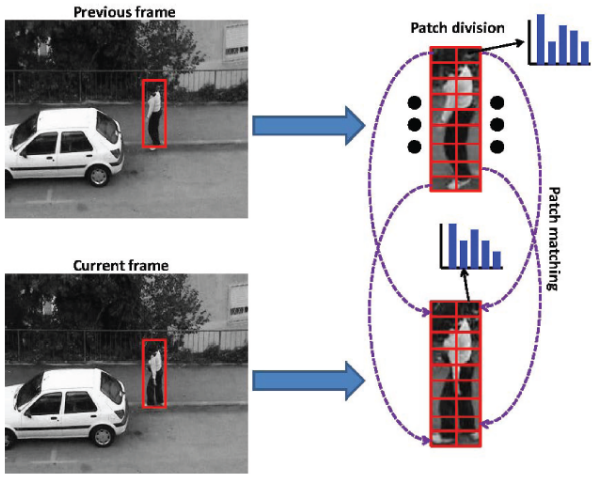
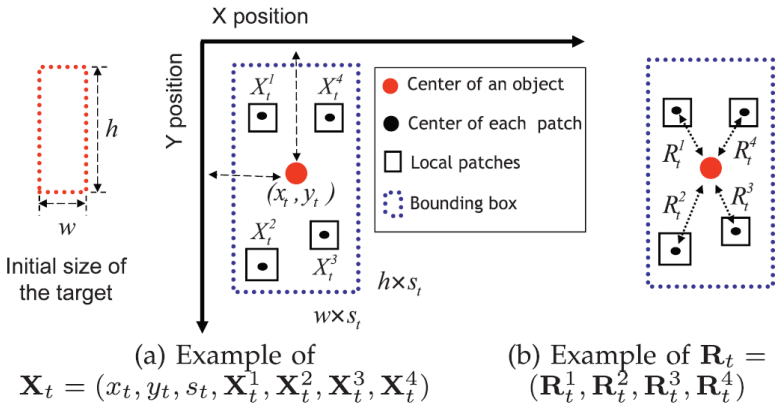


Figure 2.5: Part based representation (Adam et al. [2006]). The target is divided into multiple parts and each part associates with one histogram and votes for the centre location of the target.



(a) Example of $\mathbf{X}_t = (x_t, y_t, s_t, \mathbf{X}_t^1, \mathbf{X}_t^2, \mathbf{X}_t^3, \mathbf{X}_t^4)$ (b) Example of $\mathbf{R}_t = (\mathbf{R}_t^1, \mathbf{R}_t^2, \mathbf{R}_t^3, \mathbf{R}_t^4)$

Figure 2.6: Patched based Appearance Model (Kwon and Lee [2013]).

Each object representation normally comes with an appropriate appearance model. A number of excellent reviews of appearance models exist (e.g. Yilmaz et al. [2006]; Cannons [2008]; Li et al. [2013]). Recent years have seen an increased focus on appearance modelling, often including an element of machine learning. Appearance models can be broadly categorised into two types: generative and discriminative. Generative models try to learn the appearance of an object, while discriminative models try to build and train a classifier to distinguish the object from the background.

2.3.1 Generative Models

Many features are available for use in visual tracking: colour, corners or points, gradient orientation, motion, contour, texture, etc. Selecting the right features plays a critical role in tracking performance (Yilmaz et al. [2006]). A number of questions should be considered while selecting features for tracking (Collins et al. [2005]):

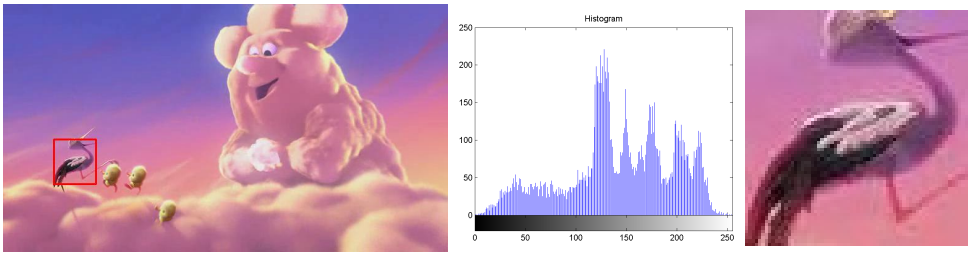
- How many features will be selected;
- What type(s) of features are used;
- What are the feature selection mechanisms;
- When during tracking should feature selection be made.

Colour features are widely used in tracking. They are typically represented in the form of colour distributions or colour histograms (e.g. Comaniciu et al. [2000]; Perez et al. [2002]; Nummiaro et al. [2003]; Czyz et al. [2005]; Adam et al. [2006]; Kwon and Lee [2013]). Colour histograms (e.g. Figure 2.7b) are constructed by splitting the range

of colours into equal-sized bins. Then for each bin, the number of colour pixels from the image data that fall into each bin are counted. They allow for significant data reduction, and can be computed efficiently; Moreover, discarding colour spatial distribution, colour histograms are robust to noise, small object deformation, scaling and rotation, and partial occlusions. Conversely, without spatial or shape information, similar objects of different colour may be indistinguishable based solely on colour histogram comparisons. By choosing different colour spaces such as RGB, HSV or rgb (normalised) colour spaces could make the colour histogram different. It, however, depends on the nature of the video sequences and applications. Also, selecting which colour components in which colour spaces has been addressed by several works such as feature selections. The basic idea is that making the target appearance differential to the background appearance.

To reduce effect of local background such as (partial) occlusion or clutter, by assigning smaller weights to pixels farther from the target centre, Comaniciu et al. [2000] incorporated a kernel (e.g. Epanechnikov kernel) into the calculation of colour histograms. To preserve spatial information, Maggio and Cavallaro [2005a]; Adam et al. [2006]; Shahed Nejhum et al. [2008]; Kwon and Lee [2013] used colour histogram computed over local patches to represent the target. These methods need a mechanism to combine all votes for the target (centre) location.

Another approach is to use a template image (e.g Figure 2.7c) of an object as an appearance model. An advantage of a template is that it carries both spatial and appearance information. Templates, however, only encode the object appearance generated from a single view. Thus, they are only suitable for tracking objects whose pose does not vary considerably during the course of tracking Yilmaz et al. [2006]. Some works (e.g. McIntyre et al. [2009]) tried to integrate geometric transformations of templates into tracking. These methods could help to reduce the number of templates used in tracking. They, however, need to specify how the template is transformed and are specific applications.



(a) Image (b) Histogram (c) Template

Figure 2.7: (Enlarged) Histogram & Template.

Colour correlograms (Huang et al. [1998]) have also been used in tracking (e.g. Zhao and Tao [2005, 2007]). These express how the spatial correlation of pairs of colours changes with distance. Informally, a correlogram of an image is a table indexed by colour pairs, where the kth entry row (i, j) specifies the probability of finding a pixel of colour j at a distance k from a pixel of colour i in this image. In other words, a colour correlogram contains not only colour statistics, but also its spatial distribution. It, however, is more suitable for use in content-based image retrieval because of its computational complexity and memory consumption.

The Gaussian Mixture Model (GMM), a weighted sum of m component Gaussian densities, has been applied in several works (e.g. McKenna et al. [1997]; Kim et al. [2014]). Expectation-maximisation (EM) (Dempster et al. [1977]) is typically used to estimate GMM parameters. The resulting mixture model will depend on the number of components m; and the initial choice of parameters for these components.

Motion (e.g. optical flow) is another feature used in visual tracking. Optical flow produces a dense field of displacement vectors which defines the translation of each pixel in a region (Horn and Schunck [1981]; Lucas and Kanade [1981]; Farnebäck [2003]). It is computed using the brightness constraint, which assumes brightness constancy of corresponding pixels in consecutive frames, and assumes neighbouring pixels have similar motion. Optical flow is commonly used as a feature in motion-based segmentation and tracking applications. Kristan et al. [2009] incorporated the motion into the target appearance model to handle the problems arising when the target is close or occluded by a visually similar object. The method assumed that the target does not significantly change its motion during the occlusion. Figure 2.8 shows an example of a dense optical flow.

(a) #3 (b) #4 (c) Optical flow

Figure 2.8: Dense Optical Flow (using Farnebäck [2003]).

Local features (e.g. Kim [2008]) used corner features (Förstner and Gülch [1987]), Zhou et al. [2009] used SIFT (Lowe [1999]), He et al. [2009] used SURF (Bay et al.

[2006]) are also used to represent the target appearance. These features are detected inside the target boundary and tracked in consecutive frames. The target position can be estimated according to local feature locations. Local features are reliable and robust to illumination and appearance changes. The performance of the tracker, however, degrades if too few features are detected or mis-tracked. Moreover, some outlier features can give an incorrect target location estimation.

In contrast with methods using a single feature, multiple feature fusion approaches try to integrate two or more features into single tracking algorithms or multiple tracker fusion methods. No single visual feature is robust and general enough to deal with changes of environment (Spengler and Schiele [2001]). Combining multiple features to describe the object may make the tracker more robust. In single tracking algorithms, Wu and Huang [2004] integrated colour and shape to form a richer target representation, while Maggio et al. [2007]; Han et al. [2011] combined colour and orientation features; Wang and Suter [2006]; Shen et al. [2003] combined colour and edges; Triesch and Malsburg [2001] integrated five cues (motion, colour, position, shape and contrast).

Taking a multiple tracker fusion approach, each cue in Wei and Justus [2008] is tracked individually and modelled by a Hidden Markov Model (HMM). All HMMs are presented in a Linked Hidden Markov Model (lHMM) to show the interaction between pairs of HMMs. In Noguer [2005] the output of one cue (colour) is used for propagation of other cues (contours).

In general, multiple feature fusion work has demonstrated that the approach makes tracking more robust. The challenge, however, raised by these works is how to estimate the contribution, i.e. the relative weight, of each feature when estimating target state. Adapting to the reliability of each feature is important since different features respond in different ways affects to changes in an object's appearance, such as motion blur, illumination change, etc.

In most applications and for long time periods, it, however, is crucial to update the target representation or model to account for appearance variations. Without adaptation, tracking is reliable only over short periods of time when the appearance does not change significantly. While much progress has been made, it is still difficult to get an adaptive appearance model to avoid drift.

Several methods have been proposed to deal with the drift problem (e.g. Matthews et al. [2004]). Though it allows fast reaction to appearance changes, naive update is rarely used since it can easily harm the model. A simple linear update of the reference model was introduced in Nummiaro et al. [2003]. They used a fixed adaptation speed, making it suitable in some situations. Collins et al. [2005] proposed to anchor the developing model

on the original one, but the method could not react quickly enough to large variations, such as the appearance of a hidden part.

2.3.2 Discriminative Models

In this approach, background information is incorporated to select the best features to distinguish the target from its local background. This method is also known as tracking by detection. A feature selection method is commonly used to discard irrelevant or redundant features in pattern classification, where an optimal subset of features is chosen from a feature set (i.e. feature pool) according to a certain criterion. The discriminative model maintains these discriminative features of the objects and updates them during tracking since the prominent feature set can differ from frame to frame due to the changes of the local background or of the target. To search for the target, a deterministic approach or sliding window is typically used.

Collins et al. [2005] maintains 49 colour features in a feature pool. These features are linear combination of R, G, B components and specified by $\mathcal{F} \equiv \{w_1R + w_2G + w_3B | w \in [-2, -1, 0, 1, 2]\}$. For each feature f, normalised histograms H^f_{obj}, H^f_{bg} with n bins for the target and the local background respectively are calculated. Then m best features are selected according to how well the object is separated from the local background using Equation 2.34.

$$VR(L; H_{obj}, H_{bg}) \equiv \frac{var(L; (H_{obj} + H_{bg})/2)}{[var(L; H_{obj}) + var(L, H_{bg})]}. \tag{2.34}$$

where

$$var(L; H_{obj}) = E[L^2] - (E[L])^2 = \sum H_{obj} L^2 - [\sum H_{obj} L]^2. \tag{2.35}$$

$$L = log\left(\frac{\max(H_{obj}, \delta)}{\max(H_{bg}, \delta)}\right). \tag{2.36}$$

and δ is a small value to prevent dividing by zero or taking the log of zero.

For each selected feature, the log likelihood ratio values (Equation 2.36) are back-projected into the image to produce a weight image for use during tracking. In perfect situations, the object pixels contain positive values and background pixels contain negative values. The colours shared by both the target and background tend towards zero, and it is easy to choose a threshold to separate object and background. However, in real video sequences, the colour distributions of object and background are seldom completely separate. Finally, the mean-shift algorithm (Comaniciu et al. [2003]) is applied to this weighted image to estimate the target location in the current frame.

Figure 2.9 summarises the process of generating and selecting features for a given

target. In general, the Collins et al. [2005] approach is slow and some visual information may be lost during back-projection. This approach also assumes that object and background do not change quickly from one frame to the next.

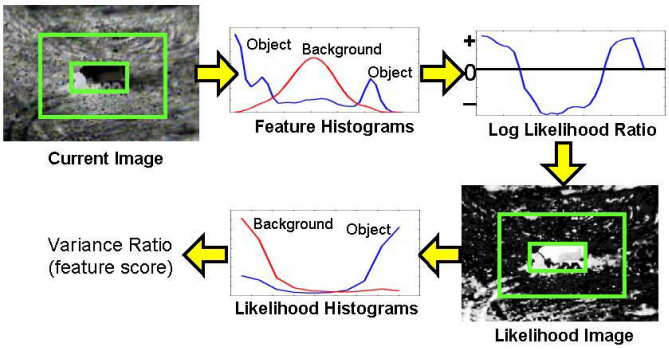


Figure 2.9: Feature generation and selection process (Collins et al. [2005]).

Other approaches used Boosting (Schapire [2002]) to build a discriminative model. Boosting has emerged as a very popular and efficient technique in machine learning and computer vision. Offline training methods have been used and achieved promising results in classification tasks and object detection (e.g. Viola and Jones [2001]). In offline techniques, all training data must be available during a separate training stage. An offline technique, however, is not best suited to tracking, since not all target appearances are known a priori. Tracking requires adaptation to variations in the target, i.e the ability to capture target appearance changes online, as much as possible.

Recently some attention has been given to online Boosting. Online learning has advantages, since it needs only some data at the beginning of tracking and can learn as new data arrive. Oza [2005] showed that if offline and online boosting are given the same training set, the weak classifiers returned by online boosting converge statistically to the one obtained by offline Boosting when the number of iterations $N \rightarrow \infty$. For more details see (Oza [2011]). In online boosting, the number of weak classifiers is fixed at the beginning and one sample is used to update all weak classifiers, whereas in the offline case all samples are used to update one weak classifier. Online AdaBoost has been proposed for use in tracking using feature selection, e.g. by Grabner and Bischof [2006]. The idea of AdaBoost is to construct a strong classifier $H(x)$ (Equation 2.37) as a linear combination of T weak classifiers $h_t(x)$. A weak classifier performs slightly better than a random guess. In binary class, the error rate must be less than 50%. During training, AdaBoost focuses on hard samples, i.e. increases the weight for the wrongly classified samples and decreases the weight for correctly classified samples.

$$H(x) = sign(f(x)). \tag{2.37}$$

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x). \tag{2.38}$$

where

$$h_t(x) : X \to \{-1; 1\}. \tag{2.39}$$

and α_t is the weight of the tth weak classifier $h_t(x)$ contributing to the label prediction of the strong classifier $H(x)$.

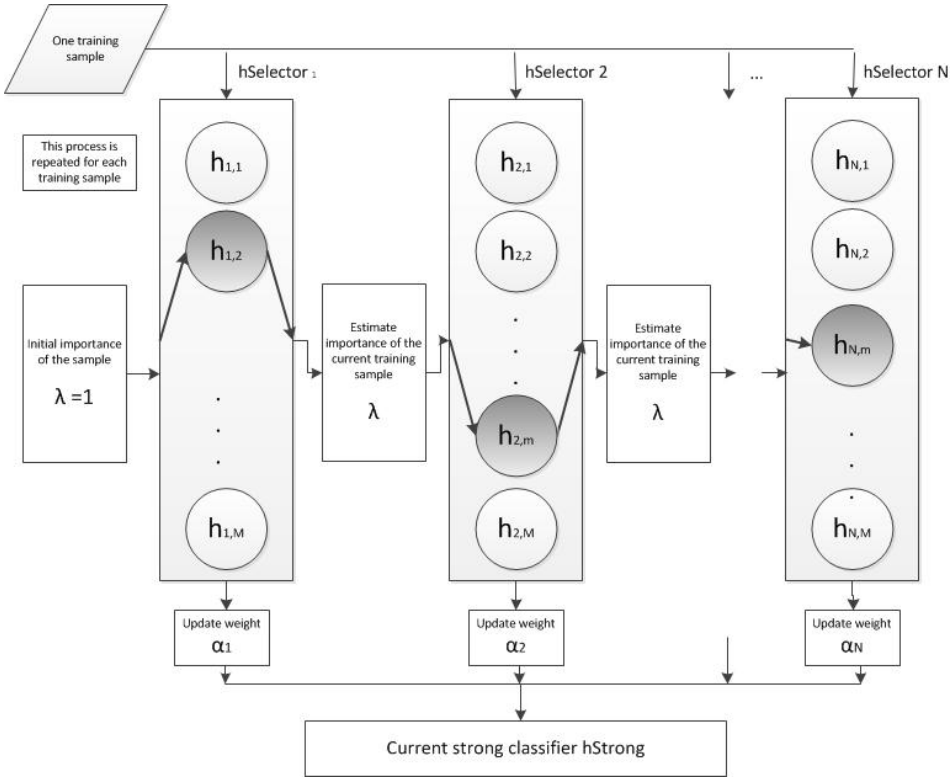


Figure 2.10: Online Boosting using feature selection Grabner and Bischof [2006].

Given a set of M weak classifiers with hypothesis $H^{weak} = h_1^{weak}, ..., h_M^{weak}$, a selector, thought of as a classifier, selects exactly one of those weak classifiers. Haar features are used in this framework. The main idea (Figure 2.10) is to apply Online AdaBoost not directly to the weak classifier but to the selectors. Training a selector means that each weak classifier is trained or updated and the best weak classifier, i.e. with lowest estimated

error is selected. Algorithm 7 describes the training process in Online AdaBoost.

$$h^{sel}(x) = h_m^{weak} \tag{2.40}$$

where m is chosen according to an optimisation criterion. The estimated error e_i of each weak classifier $h_i^{weak} \in H^{weak}$ such that $m = argmin_i e_i$.

In online learning methods tracking (Figure 2.11) is viewed as a classification problem, with the classifier which represents the object being continuously updated to keep it discriminative. Suppose that the object is detected in the current frame at time t, and is represented by an image region.The initial classifier is built using that region as a positive sample and patches in the local neighbourhood as negative samples.

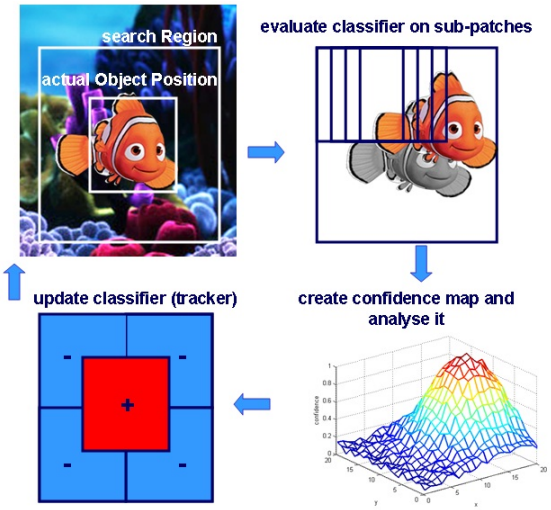


Figure 2.11: Tracking with Online AdaBoost (Grabner and Bischof [2006]).

At time $t+1$, the current classifier is used to evaluate the position of the object in a region of interest around the previous detection. The region surrounding the previous position of the target is divided into several patches. Each patch is evaluated by using current classifier to provide a confidence score which is entered in a confidence map. The confidence map is analysed and the tracking window is shifted to the most likely target position. The classifier is then updated and the process continues.

The update step will help to learn the discrimination between object and the current background. If the appearance of the object changes, then the update process can capture these changes and keep the classifier valid even though the appearance has changed. Online AdaBoost is applied in the update process.

Each weak classifier in (Avidan [2007]) is a linear hyperplane in an 11D feature space

composed of R,G,B colour and a histogram of gradient orientations (8 bins). The ensemble of weak classifiers is combined into a strong classifier using AdaBoost. The strong classifier is then used to label pixels in the next frame as either belonging to the object or the background, giving a confidence map. The peak of the map is considered the new target position, and is found using mean-shift. In the update phase, the algorithm keeps the best K weak classifiers and removes $T - K$ poorly performing weak classifiers. Before adding the new weak classifiers, the remaining K weak classifiers have their weights updated.

Despite its high efficiency and quick adaptation, online learning relies heavily on precise object localization, because it utilises previously learnt classifiers to select positive and negative training samples and then update the current classifiers with the selected training samples. Consequently, any tracking errors will gradually accumulate.

Recently, multiple instance learning (Babenko et al. [2011]) has been proposed in order to handle location ambiguities of positive samples. Samples are extracted and put into bags which are provided a label. The bag is positive if one or more instances in it are positive while the bag is negative when all of the instances in it are negative. Samples near the tracking location are put into the positive bag while samples far from the tracking location are put into the negative bag. This method can achieve robust tracking results but may lose accuracy if the image patches do not precisely capture the object appearance information, i.e. select less informative features.

Another approach is to use Semi-supervised learning, providing a general framework to learn a classifier for different types of objects which may not have enough labelled data. Tracking is treated as a semi-supervised learning problem. Grabner et al. [2008] proposed Semi-supervised boosting to break the self learning loop in Online Boosting (Grabner and Bischof [2006]) by adding a prior. All the samples provided to train this classifier are unlabelled. Despite some success in alleviating drift, this framework does not handle target changes well, because if the appearance change is different from the prior, the tracker is likely to drift off the target. Stalder et al. [2009] extended Grabner et al. [2008] to include a slowly evolving adaptive prior in combination with a fixed prior from the first frame. This method, however, does not cope with sudden appearance changes.

2.3.3 Combination Models

Generative and discriminative models each have their own advantages and disadvantages and are complementary to each other. Provided with sufficient training data, the discrim-

inative approach is expected to yield superior accuracy as compared to generative models (Lasserre et al. [2006]). Conversely, if the model is accurate, the generative approach can perform better with less data (Ng and Jordan [2001]).

Several works have sought to combine generative and discriminative models. Woodley et al. [2007] used a generative model to guide online feature selection and address the occlusion problem. If a region is labelled as occluded, the local features in this region are discarded and new features from non-occluded regions are added. The classifier is trained using the method of Grabner and Bischof [2006]. If the likelihood of one region is below a given threshold value, it is treated as an outlier (i.e. occluded). Similarly, Yu et al. [2008]; Dinh and Medioni [2011] proposed a co-training approach to handle occlusion. The generative model uses a number of low dimensional linear subspaces to describe the appearance of the object. A discriminative classifier is implemented as an online support vector machine, which is trained to focus on recent appearance variations. In the co-training approach, a principled semi-supervised training method (Blum and Mitchell [1998]), is utilised. The basic idea is to train two classifiers on two conditionally independent views of the same data (with a small number of exemplars) and then use the prediction from each classifier to enlarge the training set of the other. It is shown that co-training can find an accurate decision boundary, starting from a small quantity of labelled data, as long as the two feature sets are independent. Currently, this tracker cannot handle the case when there is an abrupt change during occlusion because there is no learned knowledge to predict the changes in the hidden region given the revealed one. Partial occlusions are often regarded as non-object by this method. This is a safe strategy, in that it avoids updating the model with the wrong appearance instances.

Tang et al. [2007] also used a co-training framework to train classifiers. The object was represented using independent features (colour histograms and histograms of oriented gradients) and an online support vector machine (SVM) built for each feature. The predictions from different features are fused by combining the confidence map from each classifier. A semi-supervised learning approach used the output of the combined confidence map to generate new samples and update the SVMs online. This approach increases the robustness of the tracker. It, however, does not handle large variation in appearances.

In general, how to combine generative and discriminative methods into a coherent framework is a classic question within machine learning and needs more research (Yang et al. [2011]), though several works have been done to reduce the drift problem and have achieved promising results.

2.4 Motion Models

Without a target motion model, trackers can only locate the target by detection in each and every frame. Once targets have been detected, data association can be performed to link the object tracks to the currently detected targets. The continuous detection approach is, however, somewhat wasteful (Cannons [2008]). If other information (e.g. motion direction, velocity) or information from previous frames (e.g. the estimated target position), is incorporated the prediction can be made more accurate and the search space reduced drastically. Target location should also become more accurate because it will have less chance to be locked on clutters or distractors.

Some trackers (e.g. Perez et al. [2002]) use a random walk motion model to search for the target. Random walk assumes that the target's velocity is a white noise sequence and so is temporally completely uncorrelated. It describes target dynamics best when the target performs radical accelerations in random directions. When the target moves in a consistent direction (which is often the case in, e.g. surveillance), random walk performs poorly, and is easily trapped in local extrema. Okuma et al. [2004] describes a proposal distribution mixing hypotheses generated by an AdaBoost detector and a standard autoregressive motion model. This approach needs an off-line training step and searches the whole image to detect all possible targets.

Predictive motion models based on previous estimates of target state are widely used, and several works have proposed methods which switch between (Isard and Blake [1998]) or combine multiple motion models. Kristan et al. [2010] proposes a two-stage dynamic model integrating a liberal and a conservative component. The liberal model allows larger perturbations in the target's dynamics and is able to account for motions between random walk dynamics and nearly constant velocity dynamics. This is achieved by explicitly modelling the target's velocity as a non-zero mean Gaussian Markov process. The conservative model assumes smaller perturbations in the velocity and is used to constrain the liberal model to the target's current dynamics. This approach can handle short occlusions well. They, however, are not designed for unexpected and abrupt motions.

Particle Swarm Optimization (PSO) (Kennedy and Eberhart [1995]), a new population based stochastic optimisation technique, has been used in some work (e.g Zhang et al. [2008]). In PSO, particles interact locally with others and with their environment. Each particle has its fitness value and a relevant velocity. In each iteration, each particle moves with its adaptable velocity according to the best state found by itself and the best state found by all particles. Particles move about the search space and cluster in the regions where the optima are located. The advantages of this mechanism are the

simplicity and low cost of the computation associated with each particle. Several parameters, however, must be tuned, (including acceleration constants, maximum velocities) to control how particles move. Moreover, maintaining the sufficient diversity within the particle set can be difficult.

Instead of combining or switching between motion models, Kwon and Lee [2011]; K. and Lee [2013] proposed a sampling method to sample a motion model from a set of motion models, estimates made over the target's recent history, to be used in one tracker. A mechanism allowing trackers to interact with each other should be designed.

The integration of contextual information indirectly modelling the target movement can achieve considerable improvements. Grabner et al. [2010] and Dinh et al. [2011] used Supporters, i.e. local key-points or features (e.g. Harris points (Harris and Stephens [1988])) around the target whose motion is correlated with the target's over a short time period, to predict target locations. In a similar manner, Yang et al. [2009] defined auxiliary objects, i.e. regions which have persistent co-occurrence with the target, consistent motion correlation to the target and are easy to track. This method relied on segmentation mechanisms to exploit auxiliary objects and a brief propagation algorithm was applied to a star topology which connects the target at the centre to other auxiliary objects and no connections among auxiliary objects, to estimate the target location. These methods can handle occlusions or target appearance changes. They, however, assume that supporters or auxiliary objects should lay on objects moving dependently and smoothly with the target.

2.5 Summary

In this chapter, some common tracking approaches were briefly reviewed. Deterministic methods are usually computationally efficient but they easily become trapped in local minima. On the other hand probabilistic methods are usually more robust, but they suffer a large computational load, especially in high-dimensional state spaces. Although considerable work has already been done above, a more effective optimisation method is still needed to support robust visual tracking.

Object representations and appearance models are crucial to tracking. The background is generally unknown in advance, and target appearance may change over time. Adaptive trackers try to capture variations in target appearance, but face the model drift problem if they try to update the target appearance model using non-target regions or when the target is occluded. Combining different features can support the tracker and help it to estimate the position more accurately in uncertain situations such as clutter,

fast motion, even though distractors. These methods, however, need a mechanism to ensure consistency between the measurements generated by different features, and it is hard to assign blame when something goes wrong.

Many state of the art discriminative online learning based tracking methods have been developed and achieved promising results because of their quick adaptation to appearance changes. They are, however, affected by model drift quicker than generative models, and they typically estimate target positions directly from exhaustive search-based methods or a sliding window. It would be reasonable and inspiring to integrate these methods into a stochastic inference framework.

Several works have been proposed to handle drift with prior (e.g. semi-supervised learning) or co-training, but cannot adapt well to appearance changes. Some build a complex motion model with the hope of allocating particles in a particle filter framework to positions which correctly estimate the posterior distribution of the target. Though this approach has achieved promising results, it is limited to specific motion types. In reality, it is very hard to model target motion precisely, especially during fast movements and unexpected changes in motion direction.

In this study, we focus first on building a rich appearance model to capture all possible target appearance changes without being too prone to drifting. After defining a target model, a search method (based on target motion) is needed to select the candidate target locations to be evaluated against the model. A motion estimation approach is introduced which can handle motion variations and enhance target prediction. These two contributions are incorporated in a single particle-filtering based tracking framework, built on Markov Chain Monte Carlo (MCMC). The next chapter describes in detail the proposed target appearance model.

Algorithm 7 Online AdaBoost for feature selection (Grabner and Bischof [2006]).

Require: Training example (x, y), $y \in -1, +1$
Require: strong classifier h^{strong} (initialised randomly)
Require: weights $\lambda^{corr}_{n,m}, \lambda^{wrong}_{n,m}$ (initialised with 1)
Initialise the important weight $\lambda = 1$
//for all selectors
for $n = 1, 2, .., N$ do
  //update the selector h^{sel}_n
  $h^{weak}_{n,m} = update(h^{weak}_{n,m}, (x, y), \lambda)$
  //estimate errors
  if $h^{weak}_{n,m}(x) = y$ then
    $\lambda^{corr}_{n,m} = \lambda^{corr}_{n,m} + \lambda$
  else
    $\lambda^{wrong}_{n,m} = \lambda^{wrong}_{n,m} + \lambda$
  end if
  $e_{n,m} = \frac{\lambda^{wrong}_{n,m}}{\lambda^{corr}_{n,m} + \lambda^{wrong}_{n,m}}$
end for
  //choose weak classifier with the lowest error
  $m^+ = argmin_m(e_{n,m})$
  $e_n = e_{n,m^+}; h^{sel}_n = h^{weak}_{n,m^+}$
  if $e_n = 0$ or $e_n > \frac{1}{2}$ then
    exit
  end if
  //calculate voting weight
  $\alpha_n = \frac{1}{2} ln \left(\frac{1-e_n}{e_n} \right)$
  //update important weight
  if $h^{sel}_n(x) = y$ then
    $\lambda = \lambda . \frac{1}{2.(1-e_n)}$
  else
    $\lambda = \lambda . \frac{1}{2.e_n}$
  end if
  //replace worst weak classifier with a new one
  $m^- = \text{argmax}_m \, e_{n,m}$
  $\lambda^{corr}_{n,m^-} = 1; \lambda^{wrong}_{n,m^-} = 1;$
  get new h^{weak}_{n,m^-}
end for

Chapter 3

Tracking with Multiple Generative Models

3.1 Introduction

The previous chapter presented an overview of common tracking approaches and described two important components of a tracker. Though the appearance model is a key component and contributes significantly to the success of a tracking framework, maintenance of an effective appearance model remains an open problem. Existing methods using sophisticated image observation models tend to be effective but computationally intensive, or efficient but vulnerable to false alarms. Without effective verification, the tracker is likely to drift away gradually or fail.

Appearance-based trackers typically construct an appearance model of the target using features extracted from the first frame, comparing it to measurements recovered from incoming frames at candidate target positions to estimate the most likely target state. Targets, however, move in complex environments and targets' appearance can vary over time as a result of illumination changes, pose variations, full or partial occlusions, deformable targets, etc. It can be difficult to segment the target from its background in real image sequences.

A fixed appearance model, as in Isard and Blake [1996], Birchfield [1998], can soon become insufficient. To achieve long term tracking, many researchers have tried to learn appearance models - to adapt the model to match changes in the target. Two classes of model are used to capture targets' appearance: generative (Comaniciu et al. [2003], Ross et al. [2008], Nummiaro et al. [2003]) and discriminative (Grabner and Bischof [2006], Collins et al. [2005], Babenko et al. [2011]). Regardless of approach, adaptive

appearance-based trackers face a key problem: the model drift that occurs when background information contaminates the model. The risk and degree of drift increase quickly if the tracked target is not well-located. Several methods have been proposed to deal with drift, as discussed in Section 2.3 (Chapter 2). Despite some success in alleviating drift, they do not adapt well to large or sudden appearance changes.

The goal of the work reported here is to produce an online tracker capable of adapting to fast appearance changes without being too prone to drifting, and able to recover following drift and partial or full occlusion. It is not necessary to update the target appearance model at every image frame, as a tracker with one fixed appearance model is likely to track the target well over short periods. Knowing when and where to update an appearance model and how to choose an appropriate appearance model are important questions for an adaptive tracker.

A target representation should be descriptive enough to disambiguate the object from the background, while allowing enough flexibility to cope with changes of target scale, pose, scene illumination and partial occlusions. My simple yet effective method builds appearance models which are a combination of two popular generative models: templates and histograms.

Templates can provide stable matching and good localisation, due to the detailed spatial information they carry. Though templates are very vulnerable to appearance changes, they provide a solid clue that the target has changed its appearance and the tracker should update the appearance model (e.g. a template based tracker should update the template). Histograms, in contrast, do not maintain spatial information and so are more robust to rotation, scaling and partial occlusion. Histograms can be thought of as a more abstract model, as many templates can produce a given histogram. The relative lack of precision of histogram-based representations allows them to capture target appearance during changes in the spatial distribution of target features.

During tracking, especially in unconstrained environments, appearance changes are unpredictable. A fixed set of templates cannot be relied upon to capture the variations that might arise. With careful use, templates and histograms can complement each other. Templates allow the tracker to produce suitable histograms, while histograms allow the tracker to estimate the new target location which in turn allows new templates to be sought, and used to cope with changes in target appearance.

In the proposed method, each new appearance is learnt and maintained in a pool of appearance models. Storing multiple template-histogram pairs allows the tracker to handle variations by automatically switching among models, using template matching to select a histogram which captures target appearance in the current frame. This reduces

the risk of drifting, since it is possible to check the similarity between the new and previous appearances before updating or adding a new appearance model to the pool. Instead of computing appearance changes between temporally adjacent frames, or between the current frame and the first frame, this tracking method evaluates the change by computing differences between the current appearance and a number of learnt models that previously appeared in the image sequence. In case of drifting or occlusion, the tracker can re-initialise the tracking process by selecting a new model from the pool.

This approach to appearance modelling is built into the Markov Chain Monte Carlo (MCMC) based particle filter (Khan et al. [2005]). We extend the proposal distribution of the standard MCMC to propose both the new location, and the histogram that should be used. On completion of each Markov chain, each histogram is assigned a weight reflecting how frequently it was accepted during that chain. The new target location is estimated by identifying particles which have the highest weight and use the most common histogram. This strategy is adopted because, if the chain runs for long enough, the most suitable histogram will be used most.

The rest of this chapter is organised as follows. In Section 3.2, we describe the proposed method. Experimental results and discussions presented and discussed in Section 3.3 and Section 3.4 respectively. Finally, some conclusions are drawn in Section 3.5.

3.2 Proposed Tracking Algorithm

Figure 3.1 shows the main steps in the proposed method, Markov Chain Monte Carlo based Particle filter using sampled appearances (MCMC-SA). This method mainly focuses on the appearance model and assumes that target movement is smooth. The issues raised by more complex motion are addressed in Chapter 4.

In this proposed approach, each appearance model (constructed to represent the target appearance at a specific time) is maintained in an appearance pool. During the MCMC based tracking process, the tracker accesses the appearance pool and selects an appropriate appearance model with which to estimate new target locations. The new target appearance is extracted from the estimated target location and the tracker makes a learning decision on this new appearance. If the new appearance is accepted, it is stored into the appearance pool for future use. These above processes are repeated for each incoming image frame.

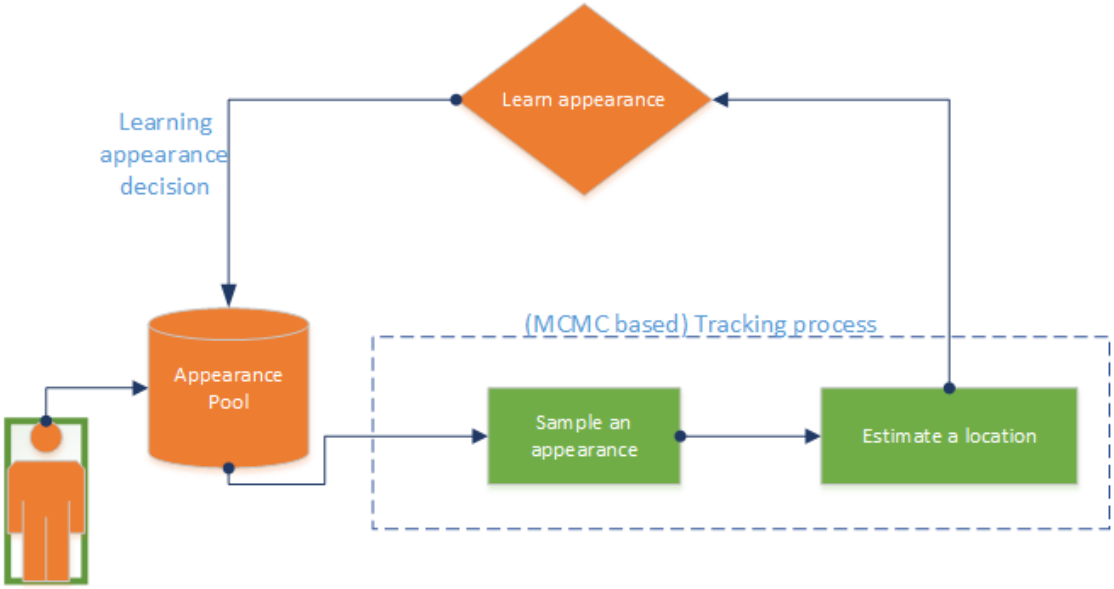


Figure 3.1: Overview MCMC-SA framework.

3.2.1 Motion Model

Dynamic models describe the likely movement of the target between successive frames. Denote the target state at time t by $X_t = \{x_t, s_t\}$ where $x_t = (u, v)$ is the centre target location and $s_t \in (1..k)$ is the index of the selected appearance model at time t, k is the total number of appearance models in the appearance pool.

The first order auto regressive model described by Equation 3.1 is adopted for the dynamics of target locations over time which proposes a new target's state X_t based on the previous states X_{t-1}. This model captures smooth movements of the target within the range of the process noise.

$$x_t = A \cdot x_{t-1} + w_t. \tag{3.1}$$

where x_t is the target location of the state at the time t, x_{t-1} is the target location at the time $t-1$, A is an identity matrix in all experiments of this study and w_t is the process noise, a Gaussian noise with zero mean $N(0, \sigma)$.

The s_t of the target state X_t selected at the time t will be discussed in Section 3.2.3.

3.2.2 Appearance Model

Visual appearance is critical for tracking since the target is tracked or detected based on the match between the observed visual evidence (or measurements) and the visual appearance model. If the assumptions made by the appearance model are incorrect or inaccurate, the tracker may fail. In the current implementation, targets are selected by manual annotation of the first frame in the image sequence. Automatic initialisation could be used to select the target. Target detectors should be trained or modelled; for example Dalal and Triggs [2005] used histograms of oriented gradient (HOG) to detect humans, Viola and Jones [2001] trained AdaBoost with Haar features for face detection, Okuma et al. [2004] used AdaBoost to detect hockey players, Breitenstein et al. [2009] combined object detectors (Implicit Shape Model detector (Leibe et al. [2008]) or HOG detector (Dalal and Triggs [2005])) with a classifier (Grabner and Bischof [2006]) to detect and track multiple people. These techniques are designed for specific objects and applications. Our work, however, focuses on general methods and a rectangle is used to define the target representation. The method is computationally convenient and general enough to allow tracking of different types of targets.

Once target location is specified, its template is extracted and added to the appearance pool. For each template, a histogram model is constructed - an Epanechnikov kernel weighted colour histogram (Comaniciu et al. [2003]). This integrates a kernel into the histogram construction process, assigning a smaller weight to pixels farther away from the target centre location and so more likely to belong to the local background. Integrating the kernel is to reduce background information contaminating the target (histogram) appearance model, i.e. they contribute less weight to the appearance model. Colour is chosen here as a simple, but powerful and reliable feature widely used to model appearance when tracking objects against complex backgrounds. Templates are not robust to rotations or pose changes. They can, however, be used to detect whether the target changes its appearance with an assumption that the tracker locates the target correctly. Figure 3.2 shows an example of building an appearance model and adding it into the appearance pool. Template-histogram pairs will be used to evaluate target state hypotheses and find the most likely target location in each frame.

The colour histogram $p = \{p^{(u)}\}_{u=1..b_c}$ denotes the target model, where b_c is the number of colour histogram bins. A histogram of a candidate target $q_y = \{q_y^{(u)}\}_{u=1..b_c}$ at location y is defined as

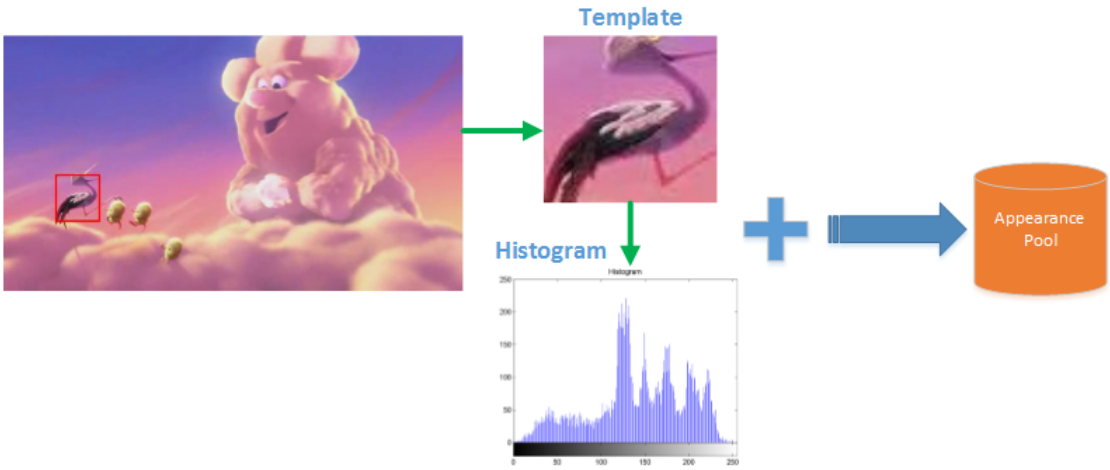


Figure 3.2: Building an appearance model.

$$p_y^{(u)} = C \sum_{i=1}^{n} k\left(\frac{\left\|y - x_i\right\|}{h}\right) \delta\left(b(x_i) - u\right) \tag{3.2}$$

where y is the target centre location, x_i is the pixel location, n is the number of pixels in the region, the function $b(\cdot)$ maps the pixel location to the corresponding histogram bin, $\delta(\cdot)$ is the Kronecker delta function as defined in Equation 3.3, $h = \sqrt{H^2 + W^2}$ is used to adapt the size of the region, H and W are the target size which are fixed in this implementation, and the normalisation factor $C = \frac{1}{\sum_1^n k\left(\frac{||y - x_i||}{h}\right)}$.

The Kronecker delta function is defined as

$$\delta(a) = \begin{cases} 1 & \text{if } a = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The Epanechnikov kernel is given by

$$k(r) = \begin{cases} 1 - r^2 & \text{if } r < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

where r is distance to the centre.

To compare the reference histogram q_t at time t of the target with the candidate histogram p_t of the state vector X_t at time t, we use the Bhattacharyya distance

$$d_t = \sqrt{1 - \rho[q_t, p_t]}. \tag{3.5}$$

where $\rho[q_t, p_t] = \sum_{u=1}^{b_c} \sqrt{q^{(u)} \cdot p^{(u)}}$ is the Bhattacharyya coefficient.

We assume Gaussian density with a constant σ for the likelihood function of the measurement histogram as follows. This function assigns more weight to a candidate target if its histogram is close to the reference histogram.

$$p(z_t|x_t) \propto \mathcal{N}(d_t; 0, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp\left\{-\frac{d_t^2}{2\sigma^2}\right\} \tag{3.6}$$

When comparing template and image data or pairs of templates, we use the Normalised Correlation Coefficient (NCC) (defined in Table 2.1 (Chapter 2)) to reduce the effect of intensity and illumination changes, though any method in Table 2.1 could be used.

The effectiveness of using multiple template-histogram pairs maintained in an appearance pool is examined in Section 3.2.4.

3.2.3 Sampling Appearance Models

The appearance models presented here are embedded into the MCMC method of Khan et al. [2005]. MCMC methods define a Markov Chain over the state space X. A candidate particle X'_t, sampled from the current sample X_t using a proposal $Q(X'_t; X_t)$, is accepted if the acceptance ratio (in Equation 2.32 (Khan et al. [2005])) exceeds 1. MCMC inherits the advantages of particle filtering outlined in Chapter 2. Moreover, each particle can be evaluated at any time; whenever it is accepted, its state is updated. This is in contrast to CONDENSATION, in which all particles are evaluated together, only after all have been generated. This characteristic is needed here. Beside the change of position, a new state proposal comprises a candidate appearance model. If this new state is accepted, the appearance model of the previous state is updated to this candidate appearance model and the weight for this appearance model is increased in the next proposal step. Use of MCMC allows the appearance model to be varied during search.

A maximum a posterior (MAP) has typically been used to find a particle most likely the target over N samples at each time t. At each time step t, an appearance pool containing templates $T_t = \{T^j\}_{j=1..k}$ and equivalent histogram models $H_t = \{H^j\}_{j=1..k}$ is given, where k is the current size of the pool.

Information from previous frames can be used to improve the accuracy of the prediction and reduce the search space; the target's previous location has been used in many

trackers. In our approach, *two* pieces of information are used when predicting target location. *First*, the previous target location is used to decide the centre of the search area. The search area S is double the target size to reduce computational cost introduced by NCC, i.e. to only focus on the area immediately around the target. *Second*, the confidence score matrix $C^j = NCC(T^j, I)$ is calculated by using NCC to compare each template T^j from T_t to each location $I(x, y)$, belonging to S, of image sequence I. It is assumed that the movement displacement of the target between two consecutive frames is not greater than twice of the target size. Also, a sliding window is used to build the confidence score matrix using templates, so that the size of the search area should be at least equal to the size of the template used.

Tracking begins with the initialisation of an MCMC chain via the Metropolis Hastings algorithm (Hastings [1970]). The starting position is the location where the maximum confidence score $C^j(x, y) \geq \theta_d$. If no location satisfies these conditions because no previously learnt templates produce a confidence score which is greater than θ_d, the starting position is determined using the first order auto regressive motion model (as defined in Section 3.2.1). Note that the selected starting point is considered to be the predicted location of the target. Though it may be a little arbitrary, the burn in period of the MCMC algorithm will discard any bias introduced by the starting point. The initial appearance model is the histogram associated with the template that best matches the last recorded target location. The NCC is a measure of the similarity between the image and the template and the score is between $(-1; 1)$. We only consider the value between $(0; 1)$. Because the larger the value, the more similar the image and template are. Selecting values for the threshold θ_d may affect the initial location because the initial position is selected at a location having the highest confidence score when comparing each template with the image region in the search area using NCC. If the threshold is too high, the correct initial position could be discarded. If the threshold is too low, it may not affect the initial position if the highest confidence score is still greater than this threshold but a low threshold will affect the addition of a new template, as discussed in Section 3.2.4.

As the MCMC chain progresses, new states are proposed according to the proposal density $Q(X_t^{'}, X_t)$. The proposal comprises changes in position according to the motion model (Section 3.2.1), and an appearance model (histogram) randomly selected from the appearance pool. The proposal density is designed by

$$Q(X_t'; X_t) = P(X_t'|X_t) = P(x_t'|x_t)P(s_t'|W). \tag{3.7}$$

where $W = \{W^j\}_{j=1..k}$ contains a set of weights associating to each appearance model.

$P(x'_t|x_t)$ describes the motion model as in Equation 3.1, and $P(s'_t|W)$ presents an appearance model randomly selected as described in Algorithm 8. In Algorithm **??**, each appearance model has its own weight. A random number is drawn in the range between $[0; 1]$ which is used to determine the index of the appearance model based on Cumulative Distribution Function.

Each appearance model has an associated weight, which records the number of times it was selected and accepted within the chain. Intuitively, the model that most improves the state hypothesis, and so can be assumed to best describe the target, will have the highest weight. Model selection takes this weight into account, better models are more likely to be selected as the chain develops. Each generated particle records its hypothesised target position, the weight associated with its appearance model, and the Bhatacharya distance between that model and the local image data. At the end of the MCMC process, the most highly weighted appearance model is identified. The particle generated using the model that has the best fit to the local image data provides the new estimate of target location. The motion model is then reapplied and templates matched to the estimated location to initialise processing into the next time frame. The process is summarised in Algorithm 9.

Algorithm 8 Sampling one histogram model from the appearance pool algorithm.

Given the set of the total times that one histogram model is selected $\{S^i\}_{i=1}^k$

1. Calculate each histogram model's weight in the appearance pool $W^i = \frac{S^i}{\sum_{i=1}^k S^i}$.
2. Initialise Cumulative Distribution Function (CDF): $c_1 = W^1$
3. For $i = 2{:}k$
   - Construct CDF: $c_i = c_{i-1} + W^i$.
4. End For
5. Draw a random number $u \sim U[0, 1]$
6. $i = 1$.
7. While $u > c_i$
   - $i = i + 1$
8. End While
9. Return i (i.e. an index of one histogram model)

3.2.4 Updating a Model

Updating an Existing Model

After locating the target in a given frame, a new template is constructed from the local image data, compared to the current template and the NCC computed. If the correlation score is greater than a (high) threshold θ_{max}, the histogram model is updated; i.e. the histogram associated with the current template is replaced by the histogram of the new estimated target location. The effect is to update a generative model (the new histogram) while anchoring it with a related, earlier template. Use of the template to select the initial histogram in the MCMC chain allows the combined model to adapt without excessive risk of drift. The approach is conservative in two ways: the histogram is only updated if new data is a close match to the current best model, and the template remains fixed.

In this approach, templates learnt during tracking are fixed and only corresponding histograms are updated. A high threshold is used to make sure that the histogram does not change significantly from the histogram first constructed by the template. A simple linear update of the reference model (e.g. Nummiaro et al. [2003]; Collins et al. [2005]) could be used in this case. Adding a new model (histogram + template) is discussed in the next section and captures other appearance changes, compensating for this conservative approach.

Adding a New Model

When the new template differs from both the current selected model and the members of the current appearance pool a new model is created and added to the pool, i.e. if the score returned by NCC is between (θ_{min}, θ_d). Using thresholds is necessary because it is redundant to add a new appearance which is fairly similar to appearances already learnt. Effective tracking with a single histogram is possible when target appearance is also (approximately) fixed. Adding more appearance models, however, allows the tracker to respond to future changes in target appearance.

The θ_d and θ_{min} are used to respectively set the upper bound and lower bound of similar levels between a new template and templates maintained in the appearance pool. The θ_d is used with two purposes: *one*, removing image noise. *Second*, if the confidence score returned by NCC when comparing the new template to (any) one template in the pool is greater than θ_d, this new template should not be added into the pool. Similarly, if all confidence scores returned by NCC is less than θ_{min}, this new template should not be added into the pool as well.

In general, by using (θ_{min}, θ_d), the tracker learns a new template if this new template

is not too similar or too different to previously learnt templates (i.e. appearances). The longer difference between θ_{min} and θ_d, the more templates are added. Section 3.3.2 discusses the technical aspects of (θ_{min}, θ_d) selection.

Together, these mechanisms extend the third strategy, Template Update with Drift Correction, of Matthews et al. [2004]. Existing models are kept unchanged, as they may support effective tracking in later frames, and the overall appearance model is updated implicitly by modifying its components. If a poor model is added, the tracker still has a chance to recover by selecting other, more correct appearance models. The proposed update method is different from those mentioned in Section 2.3 of Chapter 2, which contain and explicitly update a single appearance model.

As discussed in Chapter 2, researchers have tried to integrate a reference model or a prior into the update process to alleviate the drift problem. These methods have been shown to deal with drifting but are slow to adapt to appearance changes. Our approach extends the use of one prior (e.g. semi online learning) to multiple priors by using multiple templates to deal with variations of appearance and reduce drift. This approach is also different from the online learning approach (e.g. Online AdaBoost) because it does not discard all information learnt so far. In online learning, when a new image (i.e. sample) is provided, the appearance model is updated according to the new information and most of the information already learnt is discarded.

It is worth to mention that this framework does not neither discard any information learnt from the target appearance nor use any geometric transformations of templates because this framework is trying to apply in general tracking purpose. Moreover, there is no mechanism to predict which target information is invalid in the subsequent frames. Therefore, the tracker needs more time to select an appropriate appearance model to track the target at the current frame. However, storing more target information can help the tracker relocate the target after drift or occlusion.

3.2.5 Handling Occlusion & Re-detecting the Target

Occlusion is detected when both the NCC of the current template and location estimate, and the Bhattacharya distance between the current model histogram and the histogram computed around the location estimate, fall below a threshold, θ_{tc} and θ_{hc} respectively. When this occurs a sliding window technique, commonly applied in tracking by detection and trackers with no prediction mechanism, is used, together with all pooled appearances, to re-detect the target. The location with the best match is taken as the position of the re-appeared target. Note that in our implementation, we have not distinguished occlusion

from drifting. We used simple similarity functions (i.e. NCC and Bhattacharya distance) in appearance model comparisons. With the limitations of these functions, thresholds have been selected to decide whether the target is in occlusion or drifting. An advanced occlusion detection, e.g. employing Semi Boosting (Grabner et al. [2008]), could be embedded into this framework. In this case, one appearance model would contain a template, a histogram model and a semi boosting model.

3.2.6 Algorithm

Define N as the number of particles of the MCMC chain, M as the thinning interval before accepting one particle, B as a burn in period. The tracking process is then as described in Algorithm 9.

Algorithm 9 Appearance model sampling algorithm (MCMC-SA).

1. Initialise the starting point as described in Section 3.2.3.
2. Initially assign an equal weight for each appearance model.
3. Repeat $B + N \times M$ times
   (a) Randomly select one model from the appearance pool for this target.
   (b) Propose a new state $Q(X'_t; X_t)$.
   (c) Compute the acceptance ratio a (in Equation 2.32).
   (d) If $a \geq 1$, then accept X_t: Set the target in X_t to X'_t and update the cached likelihood. Otherwise, accept with probability a. If rejected, leave X_t unchanged.
   (e) Update the weight for each appearance model as described in Section 3.2.3.
4. The set of particles is obtained by storing N best particles.
5. The current posterior $P(X_t|Z_{1:t})$ is approximated by using MAP as described in Section 3.2.3.
6. Check if the target is in occlusion as in Section 3.2.5.
7. Update the target model as in Section 3.2.4.

3.3 Experiments and Results

3.3.1 Data

We used video sequences described in Table 3.1 for experimental evaluation. All were synthesised by Wu et al. [2013], except the Ball and Cup sequences which were from Klein et al. [2010] and the Bird2 sequence from Yang et al. [2014]. The ground truth of the target in each video sequence has been manually annotated to capture the visible part of the target by a rectangular bounding box. The test data can be grouped into two categories: one well suited to tracking using histograms, the other better served by templates. These videos show the target appearance and motion changing smoothly. Unexpected movement can occur, but the target appearance does not change drastically. When the target re-appears after occlusion, its appearance is not completely different from its previous appearance.

Sequence	Challenge	Frames	Video frames
Ball	In-Plane rotation, scale changed, partial occlusion	601	
Doll	In-Plane rotation, pose changes, partial occlusion, fast motion	3872	
David2	Illumination and pose variation	427	

Table 3.1 – continued from previous page

Sequence	Challenge	Frames	Video frames
Boy	Fast motion, in-plane rotation, face expression changed often	602	
Animal	Fast motion, clutter	71	
Jogging	Pose variation, full occlusion, deformation	307	
Jumping	Fast motion, face expression changed often, Distractor	313	
Girl	Scale changed, face expression changed, rotation	500	

Table 3.1 – continued from previous page

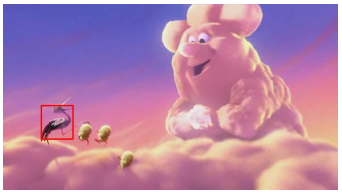
Sequence	Challenge	Frames	Video frames
Bird2	Deformation, rotation, occlusion	98	
Cup	Scale changed, clutter	629	

Table 3.1: Testing video sequences and their challenges.

3.3.2 Experimental Settings

We compared our proposed method MCMC-SA with the following existing methods: conventional MCMC (our implementation), Template-based tracking (TT, our implementation), Online AdaBoost (OAB) (Grabner and Bischof [2006]), Semi Boosting (SB) (Grabner et al. [2008]), and IVT (Ross et al. [2008]). OAB, SB and IVT rely heavily on rich appearance models to find the target and they update the target appearance during tracking.

We used 300 particles, 3 for the thinning interval, 30 for a burn in period and an 8 bin histogram for each colour channel in MCMC-SA and MCMC. In our experiments, these parameters allowed convergence within a reasonable time and produced repeatable results.

In MCMC-SA, we set the parameters of motion model A to $[1.0\ 1.0]^T$, standard deviation of the process noise σ_u to $\sqrt{8.0}$ and σ_v to 2.0 because in most of our video sequences, the target moves faster in the horizontal direction than in the vertical direction. The σ of the likelihood function (Equation 3.6) is set to 0.4 which allows the function to return

values between $(0; 1)$. The threshold on to histogram updates $\theta_{max} = 0.95$; Thresholds to detect the target $\theta_d = 0.4$; Thresholds to detect occlusion $\theta_{tc} = 0.1$ and $\theta_{hc} = 0.6$; Thresholds to add a new (template + histogram) model between $(\theta_{min}, \theta_d) = (0.17, 0.4)$. Note that it is not necessary to have a threshold θ_d to detect the target, but if used, the detected location is more reliable. Table 3.2 presents rules of thumb useful when interpreting values of the Correlation Coefficient (adopted from Taylor [1990]; Dancey and Reidy [2011]; Rumsey [2011]).

We have tested θ_d with one template selected from the first frame of our test video sequences to select a reasonable value θ_d (i.e. this is our Template-based tracking). As mentioned before, more templates will be added if the value of θ_d is set too high since they are sensitive to appearance changes.

Values selected for θ_{min} are empirical. We manually extracted templates in successive image frames, compared them using NCC and chose a suitable value among comparison scores.

Note that we only did this with two sequences (randomly picking the Jogging and Girl video sequence). These thresholds (θ_{min}, θ_d) can vary across video sequences. In our experiments, all values, however, are fixed for all testing video sequences. It would be interesting if the tracker could vary the values of (θ_{min}, θ_d) automatically.

Value	**Strength of Correlation**
1	Perfect
0.7 - 0.9	Strong
0.4 - 0.6	Moderate
0.1 - 0.3	Weak
0	Zero

Table 3.2: The Correlation Coefficient

In the Template tracking, we used Normalised Cross Correlation (NCC) to compare a template with a region in each image frame as described in Section 2.2.1 of Chapter 2. The location whose highest confidence is greater than $\theta_d = 0.4$ is labelled as the new target location.

The search areas of OAB and SB were set to *twice* the target size (i.e. samples extracted from this range are not too far from the target) and of IVT were set 40×40 pixels (the maximum displacement of the centre of the target from one frame to the next). In OAB and SB, we used 100 feature selectors. Each selector maintained 10 features.

In IVT, the standard deviation for the noise of the transition model for the bounding

box scales along the horizontal and vertical dimensions is 0.005 and 0.005, respectively; the forgetting factor is 0.99; a standard deviation of 0.25 for the observation likelihoods.

All values for parameters for compared trackers (e.g. SB, OAB, IVT) are selected as reported by their authors.

3.3.3 Result

Tables 3.3 and 3.4 summarise the results obtained. The numbers in Table 3.3 give the centre location error (in pixels) averaged over all frames of each sequence, i.e the average distance of the predicted bounding box from the centre of the ground truth bounding box. The lower a number is, the better the result. The numbers in Table 3.4 indicate the percentage of successfully tracked frames (score>0.5), where the score is defined by the overlap ratio between the predicted bounding box B_p and the ground truth bounding box B_{gt} and calculated $score = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$ (Everingham et al. [2010]). The higher a number is, the better the result. Each sequence was run three times with each tracking framework. The best result are marked in bold and the second best underlined. Note that frames showing full occlusion are excluded from the comparison but frames following occlusions are still counted.

Table 3.3 shows that MCMC-SA performs most accurately on 4 best of 10 sequences and 5 second best of 10 sequences. Table 3.3 also shows that the combination of template and histogram in one tracker outperforms comparable trackers using them independently.

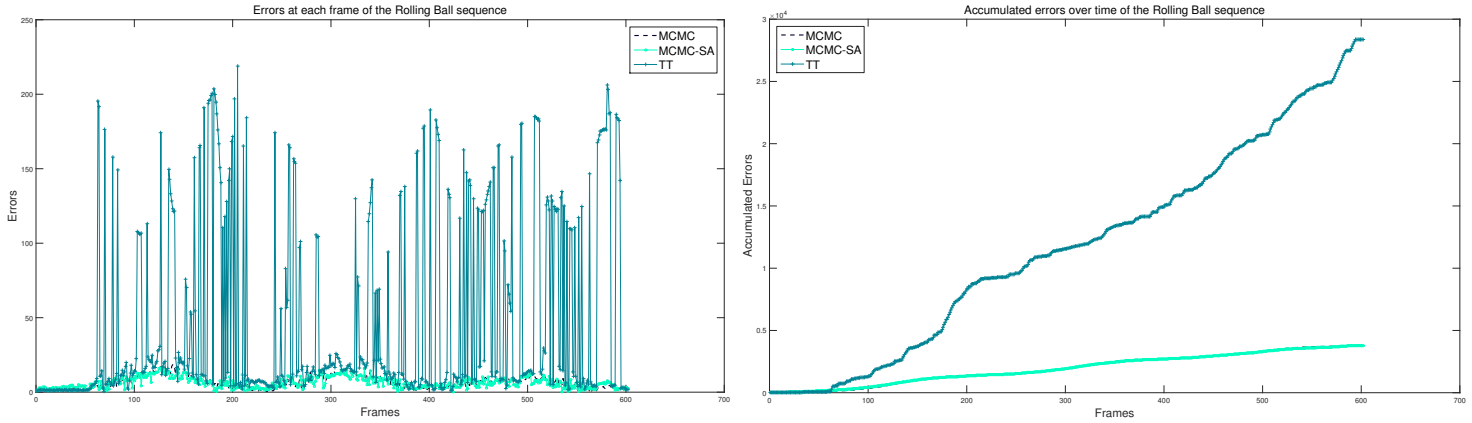
Sequence	**MCMC**	**TT**	**MCMC-SA**	**OAB**	**SB**	**IVT**
Rolling Ball (Figure 3.3)	6.34	47.13	**6.30**	159.21	168.81	98.79
David2 (Figure 3.4)	5.74	137.39	**3.55**	4.78	15.47	67.39
Doll (Figure 3.5)	9.78	16.48	**9.75**	151.76	141.28	122.95
Girl (Figure 3.6)	36.79	13.04	12.64	**3.49**	35.55	609.99
Boy (Figure 3.7)	105.43	11.93	4.19	**2.63**	235.01	210.88
Animal (Figure 3.8)	272.67	55.45	14.22	361.61	48.50	**8.67**
Jogging (Figure 3.9)	29.66	13.32	**7.37**	161.31	55.98	90.84
Cup (Figure 3.10)	**4.62**	79.86	4.80	159.09	54.59	154.62
Bird2 (Figure 3.11)	22.54	91.37	31.65	**7.59**	174.02	164.07
Jumping (Figure 3.12)	111.17	**10.28**	52.60	196.15	77.93	158.79

Table 3.3: The centre location error (in pixels) averaged over all frames of each sequence. All data were presented in corresponding graphs listed below.

Sequence	**MCMC**	**TT**	**MCMC-SA**	**OAB**	**SB**	**IVT**
Rolling Ball	0.81	0.49	**0.83**	0.16	0.17	0.11
David2	0.73	0.19	**0.93**	0.74	0.36	0.24
Doll	0.65	**0.7**	0.58	0.05	0.17	0.05
Girl	0.51	0.76	0.50	**0.96**	0.40	0.13
Boy	0.51	0.94	0.97	**0.99**	0.31	0.19
Animal	0.07	0.82	0.75	0.04	0.38	**1.00**
Jogging	0.67	0.9	**0.98**	0.25	0.71	0.25
Cup	**1.00**	0.41	0.96	0.13	0.59	0.08
Bird2	0.49	0.49	0.25	**0.98**	0.38	0.04
Jumping	0.06	**0.93**	0.06	0.05	0.07	0.08

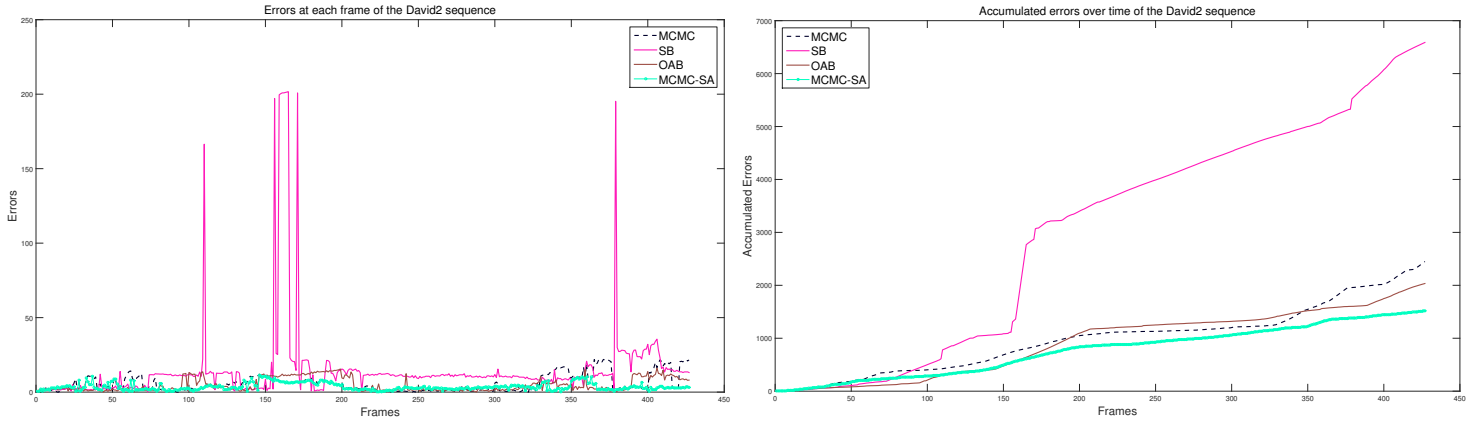
Table 3.4: The overlap ratio between the predicted bounding box and the ground truth bounding box for each testing video sequence.

The following Figures 3.3 - 3.12 show tracking errors for each tracker. Results of some trackers were removed from the figures because those trackers drifted off the target and that produced very high errors comparing to others. Appendix B shows in detail tracking results of trackers at selected frames for each video sequence.



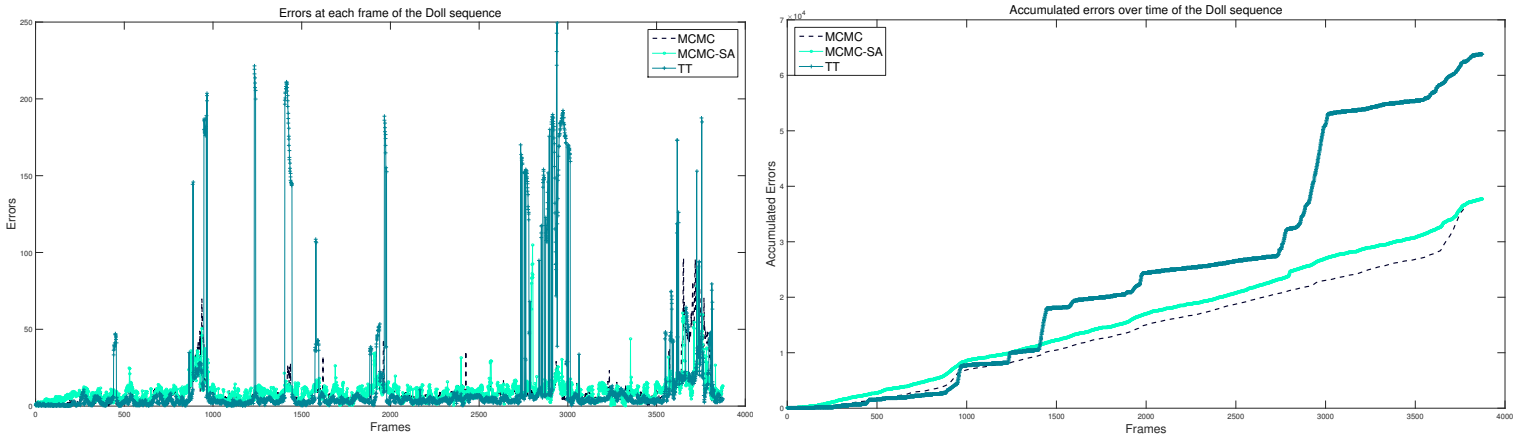
(a) (b)

Figure 3.3: Errors at each frame and accumulated errors over time of trackers for the Rolling Ball sequence. *(Note: IVT, SB, OAB trackers were removed because they drifted off the target).*



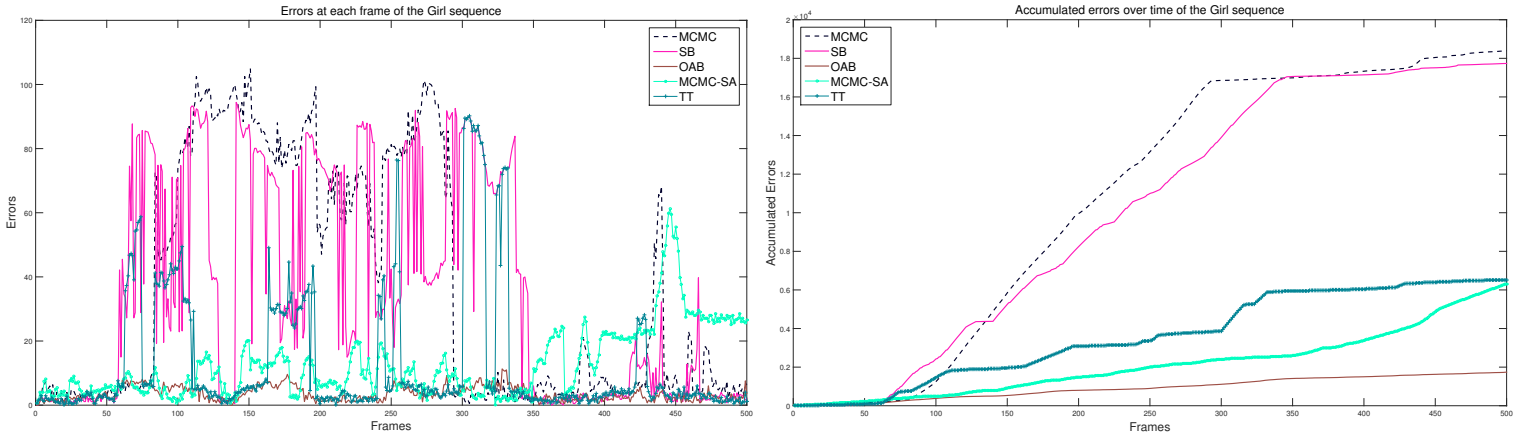
(a) (b)

Figure 3.4: Errors at each frame and accumulated errors over time of trackers for the David2 sequence.*(Note: IVT, TT were removed because they drifted off the target).*



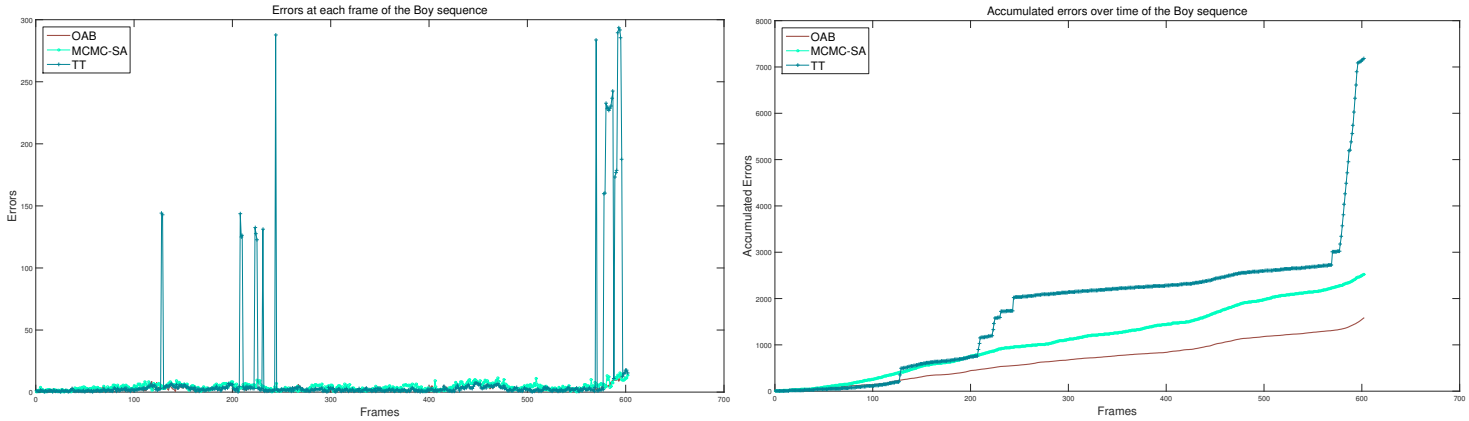
(a) (b)

Figure 3.5: Errors at each frame and accumulated errors over time of trackers for the Doll sequence.*(Note: OAB, SB, IVT were removed because they drifted off the target).*



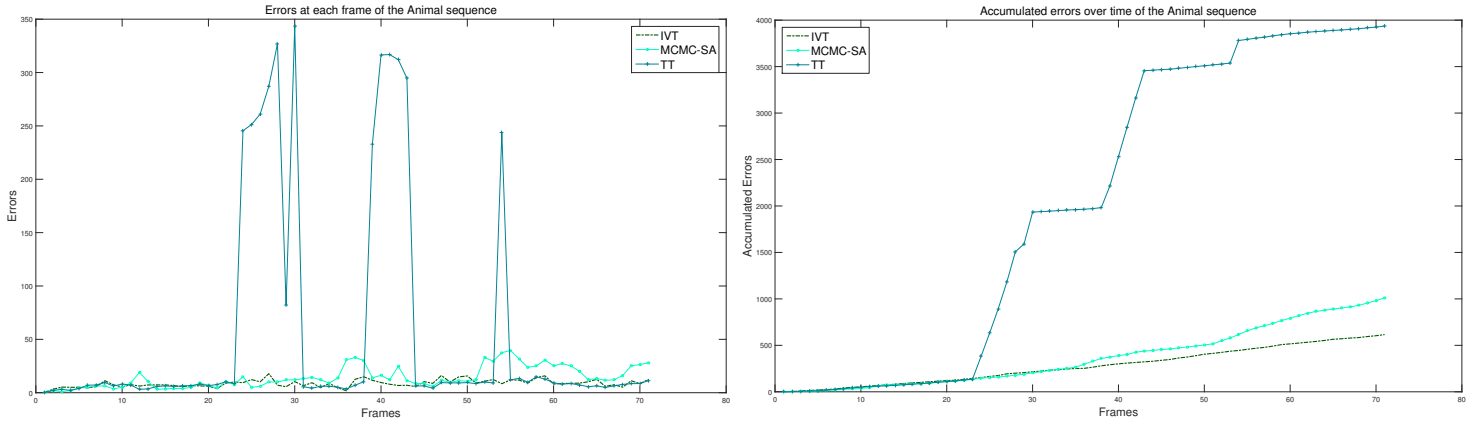
(a) (b)

Figure 3.6: Errors at each frame and accumulated errors over time of trackers for the Girl sequence. *(Note: IVT was removed because they drifted off the target).*



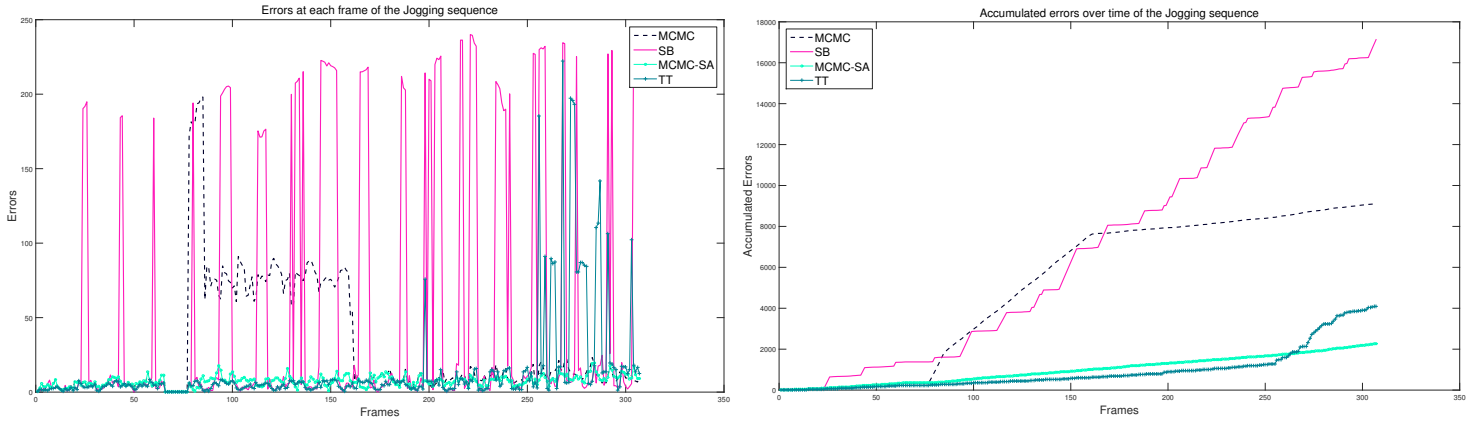
(a) (b)

Figure 3.7: Errors at each frame and accumulated errors over time of trackers for the Boy sequence.*(Note: MCMC, IVT, SB were removed because they drifted off the target).*



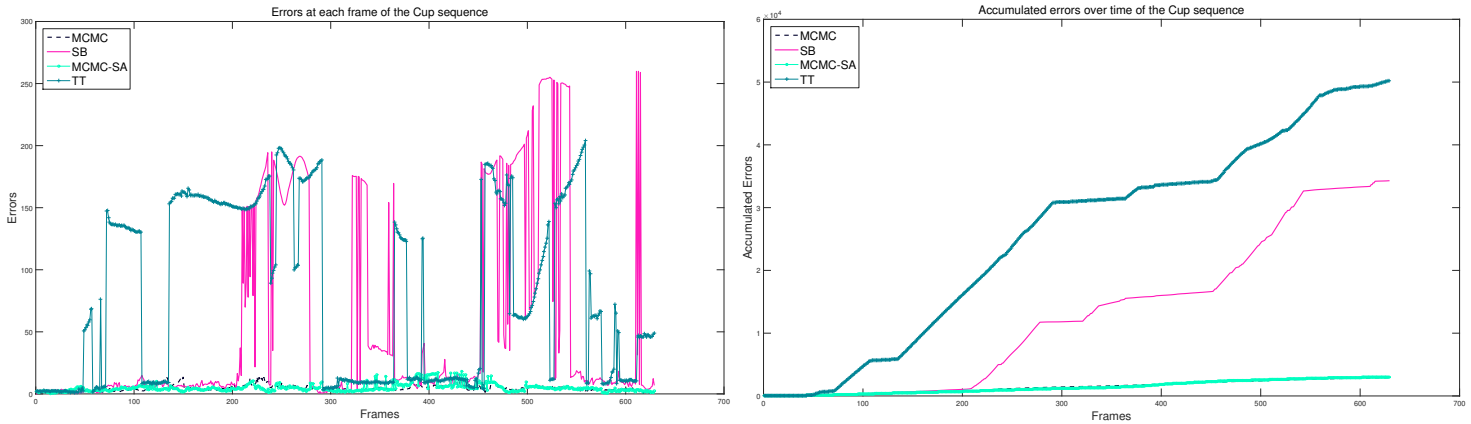
(a) (b)

Figure 3.8: Errors at each frame and accumulated errors over time of trackers for the Animal sequence.*(Note: MCMC, OAB, SB were removed because they drifted off the target).*



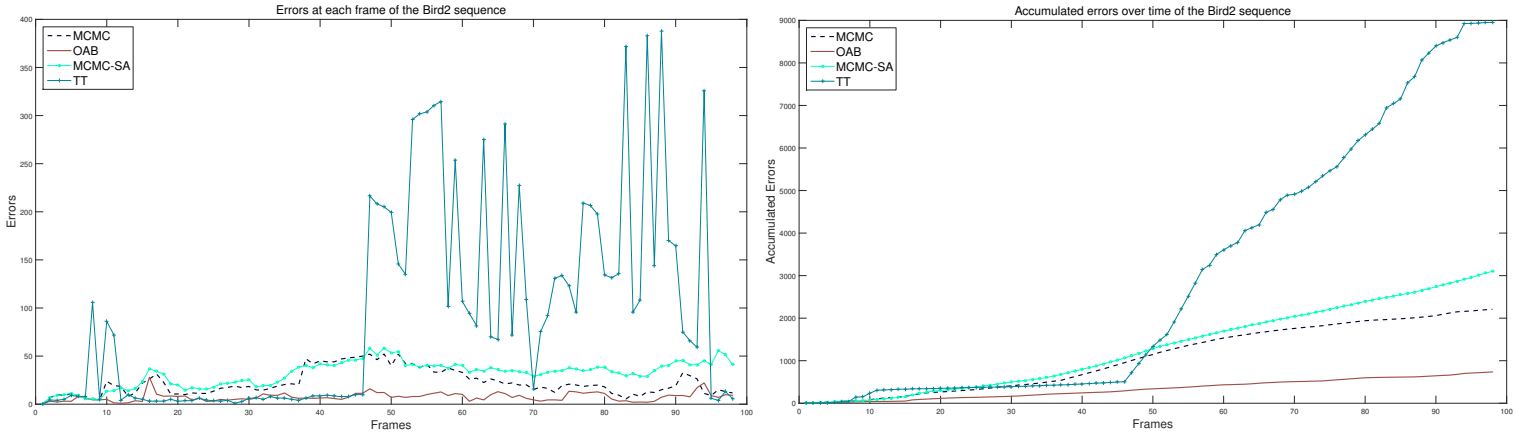
(a) (b)

Figure 3.9: Errors at each frame and accumulated errors over time of trackers for the Jogging sequence.*(Note: OAB, IVT were removed because they drifted off the target).*



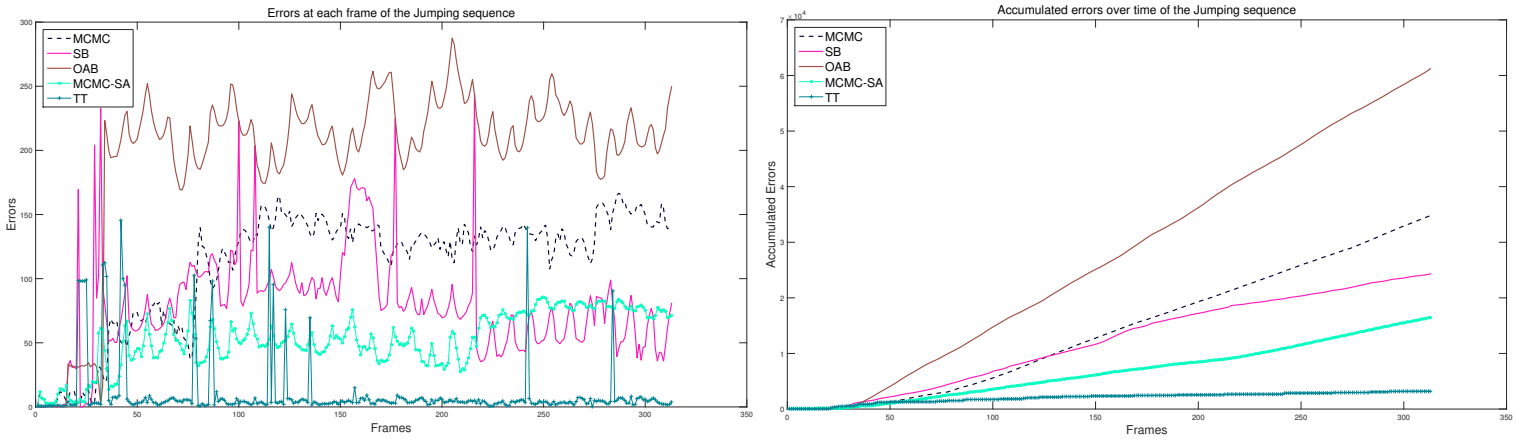
(a) (b)

Figure 3.10: Errors at each frame and accumulated errors over time of trackers for the Cup sequence.*(Note: IVT, OAB were removed because they drifted off the target).*



(a) (b)

Figure 3.11: Errors at each frame and accumulated errors over time of trackers for the Bird2 sequence.*(Note: SB, IVT were removed because they drifted off the target).*



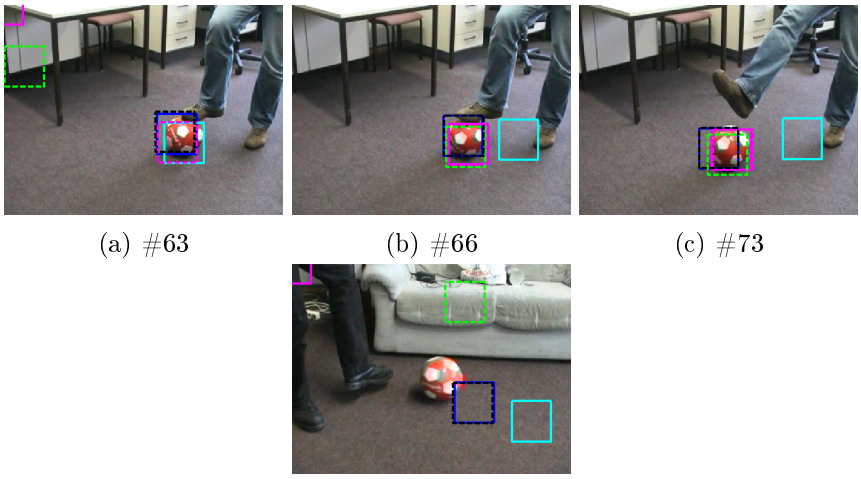
(a) (b)

Figure 3.12: Errors at each frame and accumulated errors over time of trackers for the Jumping sequence.*(Note: IVT was removed because they drifted off the target).*

3.4 Discussion

3.4.1 Appearance change handling

OAB, IVT, SB and MCMC-SA were designed to handle appearance changes. SB uses a prior to control updating the appearance model. OAB is an example of a self-learning tracker. Haar features are generated from the target providing training data for OAB. It selects and learn features which separate the target from the background. As a result, it can quickly adapt to target changes if these changes stay inside the boundary specifying the target. It tracks the target by searching for the highest (positive) confidence score in its search area. OAB worked well in the Boy (Figures B.6 and B.7), Girl (Figure B.5) and Bird2 (Figure B.11) sequences. In the Bird2 sequence, though the boundary defining the target contained more background information, OAB still located correctly the target because it can ignore features belonging to its local background. In the Boy sequence, despite the target moving unexpectedly, OAB could track the target because all appearance changes are still inside the boundary specifying the target.

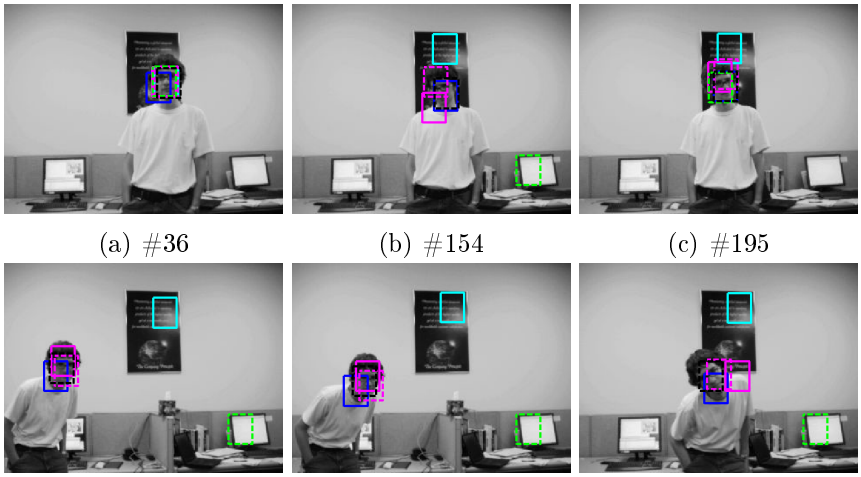


(a) #63 (b) #66 (c) #73

(d) #190

Figure 3.13: Tracking results in selected frames of the Rolling Ball sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

OAB, however, did not perform well on other sequences. In the Ball sequence (Figure B.1), for example, the ball's appearance constantly changes whilst it is rolling. OAB tracked the target until Frame #190 (Figure 3.13) but could not detect the target beyond that point. In the David2 sequence (Figure B.2), it located the target incorrectly (e.g. Frames #154, #195 (Figure 3.13)) when the target looked down. In the Cup sequence



(a) #36 (b) #154 (c) #195

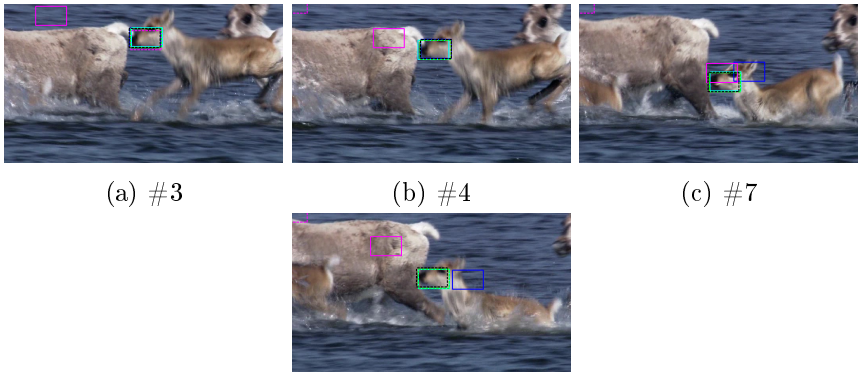
(d) #334 (e) #340 (f) #402

Figure 3.14: Tracking results in selected frames of the David2 sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

(Figure B.10), the background is complex; OAB therefore locked onto the background object instead of the target and could not re-locate the target because it updated the target's appearance model with image data not belonging to the target. In the Animal sequence (Figure B.8), it could track the target until Frame #3 (Figure 3.15) and lost the target at Frame #4 (Figure 3.15) . In the Doll sequence (Figure B.3), it wrongly estimated the target location at Frame #212 (Figure 3.16) and tracked the poster in successive image frames (e.g. Frame #338 (Figure 3.16)).

SB uses an online semi-supervised boosting method, training the classifier with Haar features and adopting search methods similar to OAB. The difference is that all samples extracted from incoming frames are unlabelled; only the sample at the beginning of the sequence is known to be a positive one. It showed that it can alleviate drift and can re-detect the target if the target appearance is similar to what it learns at the beginning of the sequence. For that reason, when the target changes its appearance, it fails. For example, it could not detect the target in the Ball sequence at Frame #63 (Figure 3.13) when the ball rotated. It could only re-detect the ball, e.g. in Frame #66 (Figure 3.13) when its appearance returned to the one it had learnt.

IVT is very sensitive to pose changes and/or partial/full occlusion. Therefore, if it has not learnt these changes, it drifts off the target. It only performed well in the Animal sequence (Figure B.8) because the deer head does not change pose while it is running. In the other sequences, when the target started changing pose, it drifted off. In the Ball



(a) #3 (b) #4 (c) #7

(d) #9

Figure 3.15: Tracking results of the Animal sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

sequence (Figure B.1), it tracked the target until Frame #66 (Figure 3.13) when the ball started moving.

MCMC-SA handles appearance changes by maintaining multiple template-histogram pairs in an appearance pool. MCMC-SA could detect appearance changes and select appropriate appearance models to track the target. So that it performed better than MCMC did in the David2 (Figure B.2), Ball (Figure B.1), Boy (Figure B.6), Girl (Figure B.5) and Doll sequence (Figures B.3 and B.4) whilst the target changed its appearance. Figure 3.17 shows example templates extracted by MCMC-SA. Note that Figures 3.17j, 3.17k still contain the target. However, the tracker has not included any method to handle the scale change, therefore when the target moves toward to the camera, i.e. its size increases, the tracker could not capture the whole target.

(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k)

Figure 3.17: (Enlarged) Girl templates detected during tracking of MCMC-SA.

TT uses only one template, so that it is easy to lose the target when the target changes its appearance and can re-detect the target where it finds the highest confidence score returned by NCC. Such that in the Doll sequence (Figures B.3 and B.4), it tracked the target until Frame # 390 (Figure 3.16) and lost the target at Frame #440 (Figure 3.16). It could re-detect the target at Frames #456, #874 (Figure 3.16).

(a) #212 (b) #338 (c) #390

(d) #440 (e) #456 (f) #874

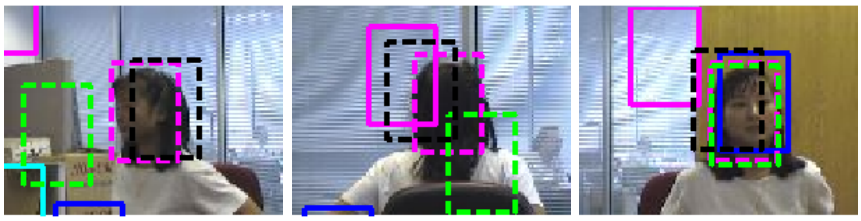
(g) #945 (h) #1420 (i) #1715

Figure 3.16: Tracking results in selected frames of the Doll sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

MCMC uses a fixed kernel weighted colour histogram. It could track the target in the Ball, Doll, David2 sequence because colour distributions (i.e. histograms) representing the target do not change significantly comparing to colour distributions of the target stored in the first frame. It, however, lost the target when the target changed its appearance (e.g. Frames #84, #189 (Figure 3.18) of the Girl sequence, Frames #320,#339 (Figure 3.19) of the Boy sequence). MCMC, by chance, could re-track the target at Frame #294 (Figure 3.18) of the Girl sequence.

3.4.2 Target location improvement

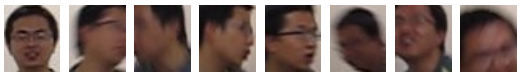
Although MCMC and TT could track the target in the David2, Doll, Boy and Ball sequences, MCMC-SA could locate the target more accurately by using templates to predict target locations, e.g. in Frames #36, #334, #340, #402 (Figure 3.14) of the David2 sequence; Frames #945, #1420, #1961 (Figure 3.16) of the Doll sequence (Fig-



(a) #84 (b) #189 (c) #294

Figure 3.18: Tracking results in selected frames of the Girl sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

ures B.3 and B.4); Frames #128, #138, #203, #320, #403 (Figure 3.19) of the Boy sequence (Figures B.6 and B.7). In the Animal sequence (Figure B.8), the target moved in unexpected directions. It, however, did not change its pose much. With the help of templates, MCMC-SA still tracked the target correctly, while MCMC failed to follow the target (e.g. Frames #7, #9 (Figure 3.15)). Figure 3.20 shows templates extracted by MCMC-SA during tracking.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3.20: (Enlarged) Boy templates detected during tracking of MCMC-SA.

Templates can give an incorrect location estimation (e.g. Frame #2794 of the Doll sequence (Figure B.4)), when the confidence score of a background object is higher than that of the target. MCMC-SA, however, could recover tracking (e.g. Frame #2804) after loss.

In the Bird2 sequence (Figure B.11), the boundary defining the target contained more background information. MCMC-SA cannot eliminate features belonging to the local background so it treated them as describing the target region. The new template extracted at the target location of Frame #58 (Figure 3.21), for instance, contains more background information but it is still added into the appearance pool because the new template still satisfied the appearance learning conditions. This affected the performance of MCMC-SA in subsequent image frames. Figure 3.22 shows templates detected by MCMC-SA. One solution is to increase thresholds controlling addition of a new template into the appearance pool. A better solution is to try to locate the target correctly.

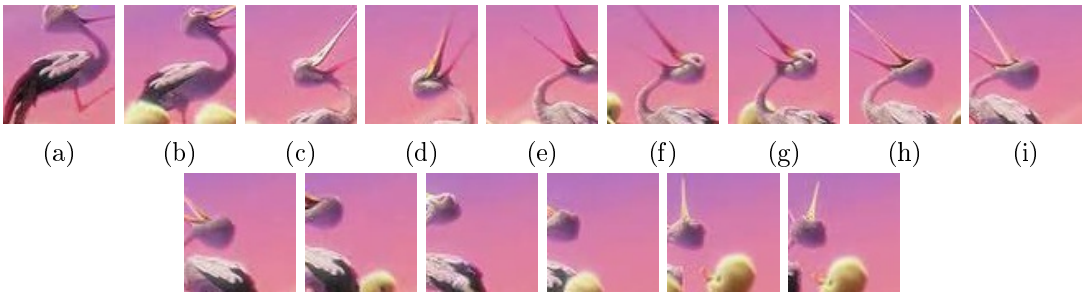
(a) #128 (b) #138 (c) #203

(d) #320 (e) #339 (f) #403

Figure 3.19: Tracking results in selected frames of the Boy sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

(a) #58

Figure 3.21: Tracking results in selected frames of the Bird2 sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



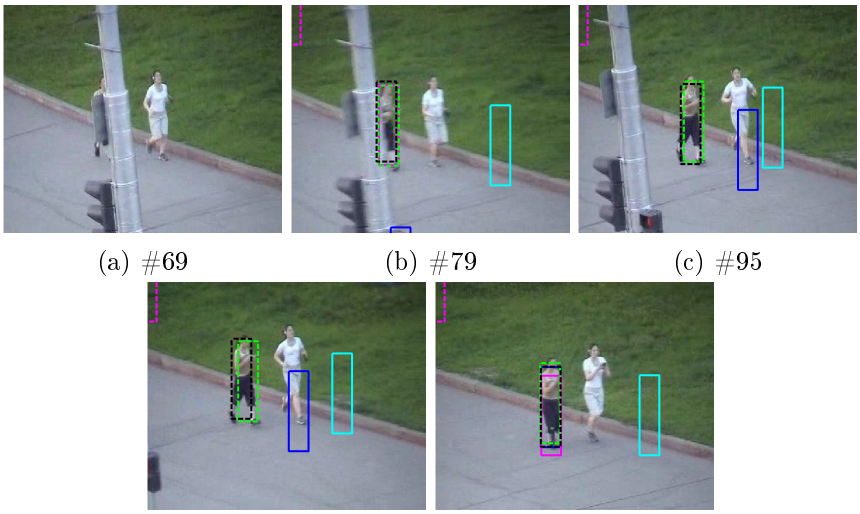
(a) (b) (c) (d) (e) (f) (g) (h) (i)

(j) (k) (l) (m) (n) (o)

Figure 3.22: (Enlarged) Bird templates detected during tracking of MCMC-SA.

3.4.3 Occlusion handling

The target is occluded by a pillar in the Jogging sequence at Frame #69 (Figure B.9). OAB stopped tracking while IVT, MCMC, TT, SB and MCMC-SA tried to find the target. IVT, however, could not. TT and SB track by searching for the highest confidence score, and so can implicitly handle occlusion while MCMC-SA has an explicit occlusion detection step. It is obvious to use thresholds to decide whether the target is in occlusion because appearance models are mainly based on generative models and simple similarity functions are used. SB and MCMC-SA could re-track the target at Frame #79 (Figure 3.23) by using a sliding window technique. However, SB lost its target in several frames (e.g. Frames #95, #113 (Figure 3.23)) because the target changed her pose. MCMC had a chance to re-locate the target at Frame #162 (Figure 3.23) and continued to track the target until the end of the sequence, because it did not update the target's appearance model.



(a) #69 (b) #79 (c) #95

(d) #113 (e) #162

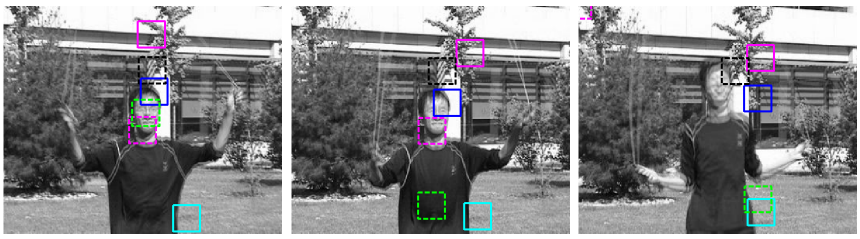
Figure 3.23: Tracking results in selected frames of the Jogging sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

3.4.4 Motion variation handling

In the Animal sequence (Figure B.8), the target moved in unexpected directions. It, however, did not change its pose much. MCMC still lost the target from Frame #9 (Figure 3.15) because its random walk motion model could not handle abrupt motions.

On the other hand, templates in MCMC-SA improved the target location estimate. This handled motion variations implicitly with assumptions that the target moves abruptly in the tracker's search area and does not change its appearance significantly. Similarly in the Boy sequence, the target moved unexpected directions. MCMC-SA learnt target appearance changes and maintained them in an appearance pool. Templates in the appearance pool provided accurate target locations for MCMC-SA.

In the Jumping sequence (Figure B.12), most trackers (e.g. MCMC, MCMC-SA, SB and OAB) were affected by abrupt target motions. They lost the target at Frame #31 (Figure 3.24) when the target started to jump. MCMC-SA could not handle this situation because templates provided incorrect target location estimates, i.e. the location has highest confidence score but it does not belong to the true target. TT uses a whole image as its search area. It, therefore, could re-locate the target frequently, though it mis-located the target several times (e.g. Frames #33, #42 (Figure 3.24)).



(a) #31 (b) #33 (c) #42

Figure 3.24: Tracking results in selected frames of the Jumping sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

3.5 Summary

We have proposed an appearance-based approach which combines two popular generative models, utilising their advantages and complementing each other to improve tracking performance. The MCMC based tracker uses a pool of template-histogram pairs to provide the best fit appearance model, switching among them using a sampling mechanism. Appearance changes are automatically detected and corresponding templates are extracted. These templates are carefully checked for similarity to other templates maintained in the pool before adding them to it. Our appearance model update approach allows the tracker to adapt to target appearance changes and reduces the drift problem by maintaining multiple templates. It does not discard all information learnt so far as the online

learning approach does because this information is useful in future frames.

Experiments showed the MCMC-SA to have performance advantages over other trackers including those using one generative component. IVT is very sensitive to occlusion and pose changes. It is likely to drift off the target when the target changes its appearance. OAB can work well if the target appearance changes are still inside the target boundary. SB could not handle the target appearance changes well because it updates the target appearance according to its prior provided in the first frame. MCMC-SA can adapt to the target appearance better if changes and target movements are smooth. Templates provide a good prediction for target locations and allow the tracker to be able to re-locate the target when mis-locating the target occurs. Occlusion can happen during tracking. By selecting an appropriate (template and histogram) appearance model, the tracker can re-track the target. A mechanism is, however, needed to handle motion variations and enhance the target location prediction to alleviate mis-locations caused by confidence scores returned by NCC. This issue is addressed in Chapter 4.

Chapter 4

Tracking with Multiple Linear Searches

4.1 Introduction

Tracking is an iterative process of model building and search. Emphasis has recently been placed on appearance modelling, and in particular on adapting appearance models to changes in target pose, scale, and illumination (as discussed in Section 2.3 of Chapter 2). Adapting to these changes, however, exposes the tracker to model drift. Many methods have been proposed to construct a rich appearance model with goals to capture appearance variations, distinguish the target from background and alleviate the drift problem. It is, however, computationally expensive and complex to verify the correctness of the appearance model. One approach to the drift problem is to improve target location through more effective search strategies which correctly capture the movement of the target, reducing the search space and the effect of distractors. Search strategies in predictive trackers rely on estimates of target motion.

The tracking algorithms presented in this chapter build upon the Markov Chain Monte Carlo (MCMC) based particle filter (Khan et al. [2005]). One drawback of conventional MCMC tracking is its reliance on a random walk. If the variance of the process noise associated with the random walk does not cover the movement of the target, tracking is likely to fail. If, however, the variance of the noise is too great the search space increases needlessly, increasing the number of particles needed for effective search (and so processing time).

More importantly, large process noise values increase the risk of the tracker being trapped by distractors. The algorithm can report a local, rather than the global, maxi-

mum when the prior distribution is peaked. A mechanism is needed which allows particles to explore all the areas where the target might be, while not being so widely spread that the tracker might lock onto distractors. Ideally, particles should only be generated close to the true target. Unfortunately, determination of the areas on which search should be focussed requires the entire space to be searched. Questions addressed here are

- How to estimate target movements using current evidence (i.e. image data in the current frame)?
- How can the search for the target be reduced via target motion models whilst still providing an accurate estimation of target location?
- How can motion models adapt quickly to best fit the current target motion, which may exhibit sudden changes in direction and velocity?

Variations in target motion can be captured by single or multiple motion models learnt from past tracking results (as described in Section 2.4 of Chapter 2). Models learnt from past results, however, may not provide accurate estimates of current movement. Multiple motion models can provide increased numbers of predictions, but can also reduce tracking performance if these predictions are inconsistent. Moreover, these motion models require parameters to be tuned and may only be suitable for specific types of motions.

The approach presented in this chapter combines bottom-up and top-down techniques to search for the target. The top-down component uses motion models to generate hypotheses (particles). The bottom-up component extracts local motion estimates which inform the motion models, supporting top-down search. Local features of the target are identified, and matched between adjacent frames. These features are stored in a feature pool. While individual feature matches may be incorrect, the distribution of likely motion directions supplied by feature matching provides valuable information that can be used to guide the search. Each feature match constitutes a hypothesis as the direction of motion of the target. The distribution of motion directions provides an implicit representation of complex target movements which are difficult to model explicitly. In the proposed tracking algorithms, the search space is modelled as multiple potential directions and one-dimensional searches are performed in those directions to find the target, reducing and carefully targeting the search.

The remainder of the chapter is organised as follows: Two novel tracking algorithms based on our approach, one using a single fixed motion direction, which is the motion direction of the centroid of target features detected named as FMCMC-C, and one sampling motion directions, which are sampled from the motion distributions constructed

by motion directions of target features named as FMCMC-S, are presented in Section 4.2. Experiments are evaluated in Section 4.3 and followed by discussions in Section 4.4. Finally, conclusions are drawn in Section 4.5.

4.2 Proposed Tracking Algorithm

Figure 4.1 shows the main steps in the proposed method, FMCMC. In this approach, a feature pool maintains features detected and matched between two consecutive frames. It is assumed that the target appearance is approximately fixed. It may change slightly due to illumination, pose and scale changes and rotation but will not become significantly different from the appearance model stored at the beginning of the tracking process. Adapting the target appearance model as well could improve tracking performance further. This issue is addressed in Chapter 5.

Given the location of the target in the first frame, the tracker extracts local features inside the target boundary. At the current frame, local features of the previous frame are tracked to produce local features at the current frame. These two sets of features are maintained in a feature pool. During the MCMC-based tracking process, matching between previous and current features constructs feature based motion estimates. The target is sought along directions sampled directly from these feature based motion estimates. After the target location is estimated, features are re-detected and the feature pool updated. These processes continue until the end of the video sequence.

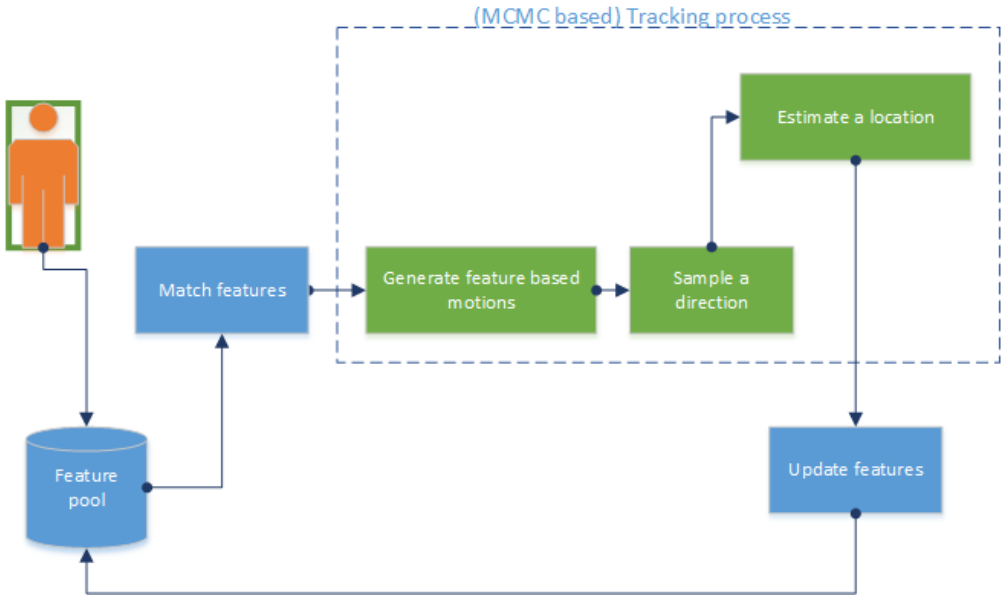


Figure 4.1: Overview of the FMCMC framework.

4.2.1 Appearance Model

In the current implementation, targets are selected by manual annotation of the first frame in the image sequence. The (Epanechnikov) kernel weighted colour histogram (as described in Section 3.2.2 of Chapter 3) is used in all experiments reported in this chapter. The target maintains a fixed histogram model during tracking. To compare the reference histogram q of the target with the candidate histogram p_t of the state vector X_t at time t, we use the Bhattacharyya distance (Equation 3.5). A Gaussian density function (Equation 3.6) is used for the likelihood function of the measurement histogram.

4.2.2 Motion Model

A random walk motion model (as used in the Chapter 3) assumes that the target's velocity is a white noise sequence and is thus temporally completely non-correlated. It describes the target's dynamics best when the target performs radical accelerations in random directions. When the target, however, moves in a certain direction, random walk performs poorly and motion is better described by the nearly constant velocity model. This assumes that velocity is temporally strongly correlated and changes in velocity only arise due to the (white) noise of the acceleration.

To cover a range of different motions, a common solution is to choose either a random walk or a nearly constant velocity model and increase the process noise to account for

the unmodelled dynamic. An obvious drawback of this approach is that poorly modelled dynamics can significantly reduce the tracker's performance. Another drawback, in the absence of additional solutions, is that an increase in process noise requires an increase in the number of particles which can, in turn, slow down the tracking.

The methods discussed above can be considered to take a top down approach. Image data, however, can provide bottom-up cues to target movement.

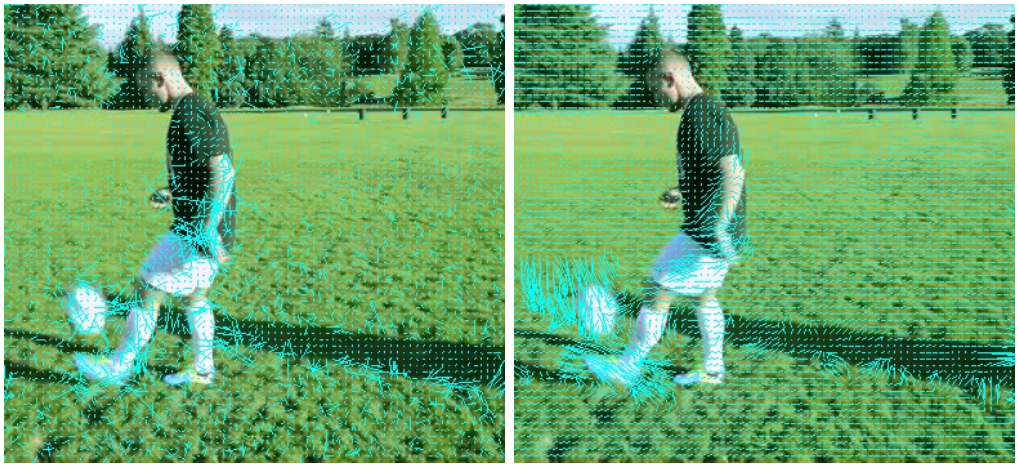
(a) Frame #1 (b) #2

Figure 4.2: Two consecutive frames of the Football sequence.

Optical flow can be used in motion detection and estimation. Figure 4.3 shows estimated movements of each pixel from Frame #1 to Frame #2 as reported by two popular approaches: Horn and Schunck [1981] and Farnebäck [2003]. These approaches consider all pixels in a region; many of which may not describe target movements correctly. These approaches, however, raise an interesting question: how can inappropriate pixel movements be eliminated while keeping the useful movements constructed by reliable pixels? To that end, local features have been selected in our approach.

Instead of using local features to represent the object (e.g. Zhou et al. [2009] used SIFT features, He et al. [2009] used SURF features, Kim [2008] used corner features), our approach utilise them to model the target movement because local features are not detected enough to cover the whole object. Besides that, it is hard to decide the object boundary based on positions of (few) local features. Feature matching, however, provides clues where the target might go. Figure 4.4 shows different types of features detected on the Frame #1 of the Football sequence.

In this section, a new approach is proposed which models target movement implicitly but can handle target motion variations correctly. In the current implementation, after



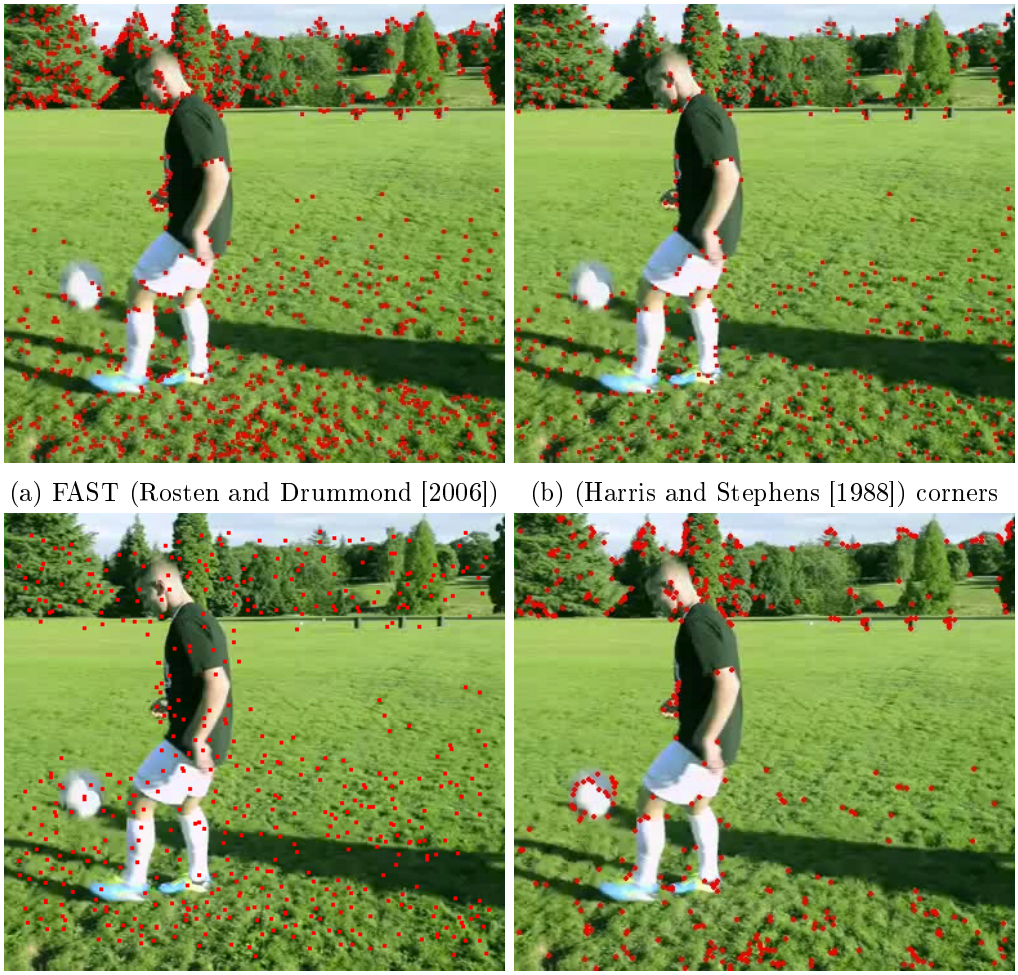
(a) Horn and Schunck [1981] (b) Farnebäck [2003]

Figure 4.3: Optical flow at Frame #2 of Football sequence.

manually selecting a target in the first frame in the image sequence, target features are extracted by applying the method of Shi and Tomasi [1994] within the target's bounding box. Shi and Tomasi proposed an affine model which proved adequate for region matching and provides the repeatable interest points needed to support robust tracking (Serby et al. [2004]).

Features are defined as $f^i = (x^i, y^i, dx^i, dy^i)$ where f^i is the ith feature, (x^i, y^i) is the location of the feature, and (dx^i, dy^i) gives its displacement relative to horizontal and vertical axes. The target maintains a feature pool $F_t = \{F^p_{t-1}, F^c_t\}$ at each time t which contains features detected in the previous tracked frame $F^p_{t-1} = \{f^i_{t-1}\}_{i=1..m}$ and features matched $F^c_t = \{f^i_t\}_{i=1..m}$ in the current frame, where m is the number of features considered.

Each feature point extracted from the target in each frame is matched with features identified in the subsequent frame using a pyramidal implementation of the Kanade − Lucas − Tomasi tracker (Bouguet [2000]) forming a set of vectors $V_t = \{v^i_t\}_{i=1..m}$ linking matched features. This approach was selected for its ability to handle large movements. Each match hypothesises the movement of one feature from one frame to the next. The directions are calculated as $d_i = atan2(dy/dx)$ to specify the angle of the movement



(a) FAST (Rosten and Drummond [2006]) (b) (Harris and Stephens [1988]) corners

(c) SURF (Bay et al. [2006]) (d) Good features (Shi and Tomasi [1994])

Figure 4.4: Features detected at Frame #1 of Football sequence.

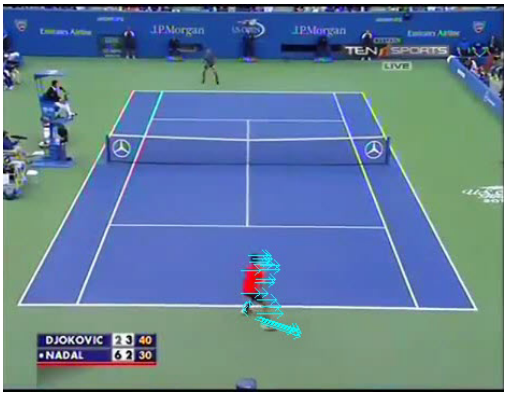
vector of one feature. The direction is defined as

$$\arctan 2(dy, dx) = \begin{cases} \arctan\left(\frac{dy}{dx}\right) & \text{if } dx > 0, \\ \arctan\left(\frac{dy}{dx}\right) + \pi & \text{if } dy \geq 0, dx < 0, \\ \arctan\left(\frac{dy}{dx}\right) - \pi & \text{if } dy \leq 0, dx < 0, \\ +\frac{\pi}{2} & \text{if } dy > 0, dx = 0, \\ -\frac{\pi}{2} & \text{if } dy < 0, dx = 0, \\ undefined & \text{if } dy = 0, dx = 0 \end{cases} \tag{4.1}$$

All the features whose motion is estimated are assumed to arise from the target and provide hypotheses as to the direction of motion of the target. Note that we assume only that features associated with the target will move in broadly the same direction. Figure 4.5 shows motion directions of features detected.

(a) Animal

(b) Table tennis



(c) Tennis sequence

(d) Football sequence

Figure 4.5: Local motion estimates obtained via feature matching. The arrows show the movement of features detected in two consecutive frames.

We use Gaussian kernel density to estimate the motion direction distribution based on the available local feature matches:

$$g(x) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{h_i}K\left(\frac{x - x_i}{h_i}\right) \tag{4.2}$$

$$K(X) = \frac{1}{\sigma\sqrt{2\pi}}exp\left\{-\frac{X^2}{2}\right\} \tag{4.3}$$

where h is the bandwidth of the Kernel Density Estimation (KDE), and m is the number of motion directions considered, each of which is measured in radians.

Algorithm 10 shows steps in constructing motion direction distribution. Figure 4.6

illustrates the construction of the motion direction distribution from a CCTV image sequence. $h = 0.1$ means that the difference between two consecutive motion direction is around 5.7 in degree. The idea is that only directions having a similar angle will be clustered into one group. Also, the more directions of features in one group, the more chance the target might go into these directions.

Algorithm 10 Building a motion direction distribution.

Given the feature sets detected in the previous frame $F^p_{t-1} = \{f^i_{t-1}\}_{i=1..m}$

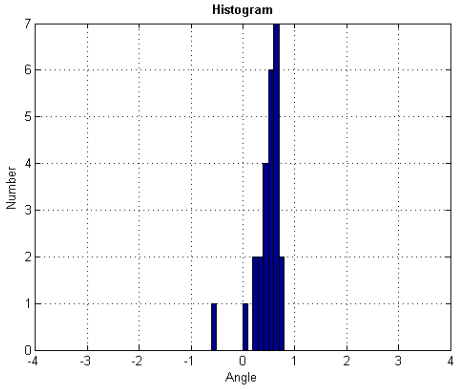
1. Match features F^p_{t-1} in the current frame using KLT to find $F^c_t = \{f^c_t\}_{i=1..m}$
2. Calculate vectors $V_t = \{v^i_t\}_{i=1..m}$
3. Set $angle = \pi$ (i.e. $angle \in (-\pi; \pi]$)
4. While $angle \geq -\pi$
   - $KDE[angle] = g(angle)$ (i.e. using Equation 4.2, and the bandwidth $h = 0.1$)
   - $angle = angle - 0.1$
5. End While
6. Normalise KDE.

Rather than searching the image in two dimensions, the proposed approach divides the search space into multiple linear segments corresponding to directions in which the target might move. In what follows we discuss two specific algorithms, both using the distribution of motion directions obtained from feature matching to support tracking. In each method, search in a given direction starts from the best state of the previously selected (and searched) direction. We adopt $y = slope \times x + intercept$ to specify search lines. The detecting and matching feature process is repeated after the target location is estimated and a new image sequence arrives. This implicitly updates the target motion model.

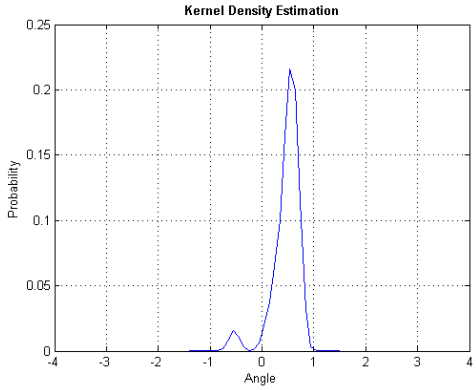
4.2.3 Algorithm

Denote the most likely state at time t of the target by $X_t = \{x_t, d_t\}$ where $x_t = (u, v)$ is the target location and d_t is the selected motion direction at time t. Define X'_t as the most likely state at time t of the target within a selected linear segment. Note that while our experiments focus on single target tracking performance, both the method and current implementation, being MCMC-based, support multiple (independent) target tracking.

(a) Matched features



(b) Histogram



(c) KDE

Figure 4.6: Kernel Density Estimation of the motion direction distribution at Frame #14 of the PETS2009 sequence.

X_t is the most likely state at time t of the target, $X'_t = \{x'_t, d'_t\}$ and $X''_t = \{x''_t, d''_t\}$ are the most likely state at time t of the target within a selected linear segment.

The proposal densities $Q_1(X'_t; X_t)$, $Q_2(X''_t; X'_t)$ are defined by

$$Q_1(X'_t; X_t) = P(X'_t|X_t) = P(x'_t|x_t)P(d'_t|KDE). \tag{4.4}$$

$$P(x'_t|x_t) \sim x'_t = A \cdot x_t + N(0, \sigma). \tag{4.5}$$

$$Q_2(X''_t; X'_t) = P(X''_t|X'_t) = P(x''_t|x'_t, d'_t). \tag{4.6}$$

$$P(x_t''|x_t', d_t'') \sim \begin{cases} u_t'' = u_t' + N(0, \sigma_u) \\ v_t'' = tg(d_t') \cdot u_t'' + b. \end{cases} \tag{4.7}$$

$$b = v_t' - u_t' \cdot s. \tag{4.8}$$

$$s = \tan(d_t'). \tag{4.9}$$

where $P(x_t'|x_t)$ describes the changes of the location, A is an identity matrix and a Gaussian noise with zero mean $N(0, \sigma)$. The proposal density $Q_1(X_t'; X_t)$ considers the changes in location x_t' and the direction d_t'.

KDE contains a motion direction distribution constructed by feature based motions, $P(d_t'|KDE)$ represents a randomly selected motion direction as described in Algorithm 11. In Algorithm 11, each direction has its own weight. A random number is drawn in the range between $[0; 1]$ which is used to determine the index of the direction based on Cumulative Distribution Function.

s (Equation 4.9) and b (Equation 4.8) are the slope and intercept respectively of the line.

In summary, given a state X_t, sample one direction from the KDE and one position X_t' belonging to that direction using Q_1. Within the sampled direction, search for the best state X_t''.

Define M as the thinning interval before accepting one particle, B as a burn in period, N_l is the number of particles used to search one line, L is the total number of lines considered.

Fixed Motion Direction (FMCMC-C)

Algorithms employing a motion direction distribution can exploit that information in a variety of ways, depending upon the assumptions they make about the target. This algorithm assumes rigid motion through a potentially noisy image sequence, i.e. that most of the feature points will move in a similar direction. The method therefore maintains only one direction.

The local motion estimates are clustered on motion direction and the largest group selected. The convex hulls of the two feature point sets concerned (in the current and previous image) are obtained, and their centre points computed. The displacement of the centre point provides a single direction d_f summarising the motion of the feature group. The target is sought along multiple, parallel lines with this direction as Figure

Algorithm 11 Sampling one motion direction model from motion distribution algorithm.

Given the target's feature pool containing two set of features $F_t = \{F_{t-1}^p, F_t^c\}$

1. Compute motion direction distribution as in Algorithm 10.
2. Initialise Cumulative Distribution Function (CDF): $c_1 = KDE^1$.
3. For $i = 2{:}m$
   (a) Construct CDF: $c_i = c_{i-1} + KDE^i$.
4. End For
5. Draw a random number $u \sim U[0, 1]$.
6. $i = 1$.
7. While $u > KDE_i$
   - $i = i + 1$
8. End While
9. Return i (i.e. an index of one motion direction in the angle $(-\pi; \pi]$)

4.7. The motion direction in Figure 4.7 is the vector between two centroids.

Note that in case of a fixed motion direction, the direction drawn from $P(d_t'|KDE)$ (Equation 4.4) is $d_t' = d_f$. Details of this search method are given in Algorithm 12.

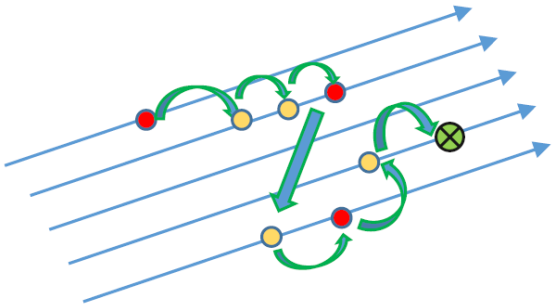


Figure 4.7: A fixed motion direction. Lines are parallel. Red dots are the best states of lines. Yellow dots are states generated. The green dot is the most likely target state. All motion directions sampled from KDE are similar.

Algorithm 12 Linear search with a fixed motion direction.

1. Detect and match features and compute the motion direction distribution as described in Section 4.2.2 and Section 4.2.3.
2. Initialise the start state X_t for the target to its current location.
3. Repeat L times
   (a) Propose a new location x'_t according to $Q_1(X'_t; X_t)$.
   (b) Calculate the intercept b for the line using the slope s and new x'_t
   (c) Repeat $B + M \cdot N_l$ times
      i. Generate x''_t of X''_t from x'_t according to the s and b using $Q_2(X''_t; X'_t)$.
      ii. Compute the acceptance ratio $a = \frac{P(X''_t|Z_t)Q_2(X'_t;X''_t)}{P(X'_t|Z_t)Q_2(X''_t;X'_t)} \approx \frac{P(Z_t|X''_t)}{P(Z_t|X'_t)}$
      iii. If $a \geq 1$, then accept X''_t: Set the target in X'_t to X''_t and update the cached likelihood. Otherwise, accept with probability a. If rejected, leave X'_t unchanged.
   (d) If the state X'_t is better than X_t then move X_t to X'_t.
4. The set of particles is obtained by storing N_l best particles at each direction.
5. The current posterior $P(X_t|Z_{1:t})$ is approximated using MAP.
6. Re-detect features for each target.

Sampling Motion Directions (FMCMC-S)

Non-rigid objects, and those undergoing complex 3D motion, often exhibit features which move in different directions. Here the cluster selection approach becomes problematic, as there may be many similar and small clusters and it is unclear how one should be selected. In this algorithm, we allow the tracker to explore a wider range of possible motion directions by sampling directly from the motion direction distribution as shown in Figure 4.8. This sampling approach gives higher weight to directions with higher probability reflecting target motion. Details of this search method are given in Algorithm 13.

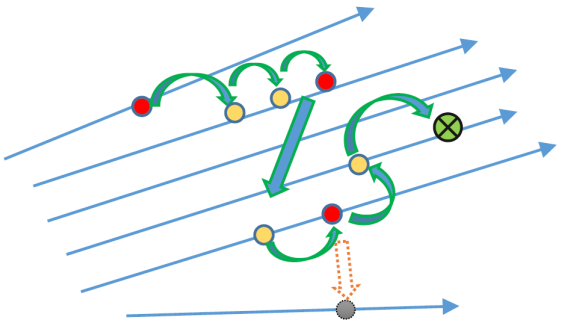


Figure 4.8: Sampling motion directions from KDE.

Red dots are the best states of lines. Yellow dots are states generated. The green dot is the most likely target state. Motion directions are sampled from KDE. The Dark dot is the best state of one motion direction but that state is not accepted.

4.3 Experiments and Results

4.3.1 Data

We used 11 video sequences (described in Table 4.1) for experimental evaluation: *Data11, Data12, Bouncing1, Bouncing2, Table Tennis, Tennis match, Football* are from own our collection; *Emilio face* from Maggio and Cavallaro [2005a]; *Hand* from AVSS2007; *Animal* from Kwon and Lee [2010]; *PETS 2009* from http://www.cvg.rdg.ac.uk/PETS2009. Note that Bouncing1, Bouncing2, Table Tennis, Tennis Match and Football sequence are public videos that we selected because of their fit to our research experiments.

The artificial Data11 and Data12 video sequences show several objects moving about a scene with a noise filled background. Moving objects are each governed by the first order auto-regressive $X_t = X_{t-1} + \mathcal{N}(0, \sigma)$ with larger direction and velocity changes which can be manipulated manually. The background contains several static objects with different sizes and shapes. The colour of the moving and static objects are randomly generated. The size of those objects remains constant throughout each sequence. Some objects come into close proximity, some introduce partial occlusion.

The test data forms three groups: synthesised (Data11, Data12, Bouncing1 and Bouncing2), indoor (Table Tennis, Emilio, Hand and Girl) and outdoor environments (Tennis Match, Animal, Football and PETS2009). Tracked targets in these videos do not change their appearance significantly compared to their appearance in the first frame. All these image sequences are challenging. The targets' motions are complex, i.e. they

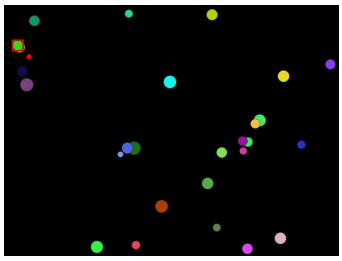
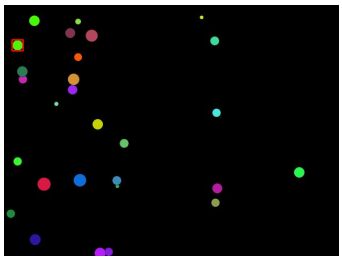
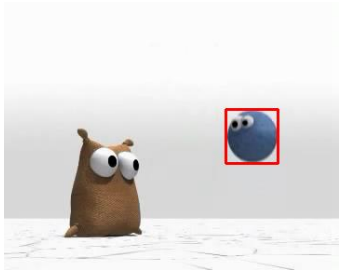
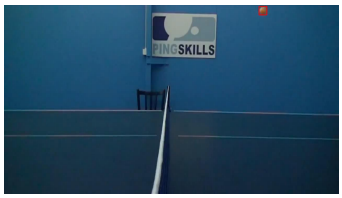
Algorithm 13 Linear search with motion direction sampling.

1. Detect and match features and compute the motion direction distribution as described in Section 4.2.2
2. Compute Motion Direction Distribution (Algorithm 10).
3. Initialise the start state X_t for the target to its current location.
4. Repeat L times
   (a) Randomly select one direction from KDE of this target using Algorithm 11
   (b) Calculate the slope s of the selected direction.
   (c) Propose a new state $Q(X'_t; X_t)$.
   (d) Calculate the intercept b for the line using the slope s and new X'_t.
   (e) Repeat $B + M \cdot N_l$ times
      i. Generate X''_t from X'_t according to the s and b.
      ii. Compute the acceptance ratio $a = \frac{P(X''_t|Z_t)Q_2(X'_t;X''_t)}{P(X'_t|Z_t)Q_2(X''_t;X'_t)} \approx \frac{P(Z_t|X''_t)}{P(Z_t|X'_t)}$
      iii. If $a \geq 1$, then accept X''_t: Set the target in X'_t to X''_t and update the cached likelihood. Otherwise, accept with probability a. If rejected, leave X'_t unchanged.
   (f) If the state X'_t is better than X_t then move X_t to X'_t.
5. The set of particles is obtained by storing N_l best particles at each direction.
6. The current posterior $P(X_t|Z_{1:t})$ is approximated by using MAP.
7. Re-detect features for each target.

can move either smoothly or variably in unexpected directions. Also, the backgrounds contain objects of similiar appearance which come close to and partially occlude the target.

The ground truth of the target in each video sequence has been manually annotated to capture the visible part of the target by a rectangular bounding box.

Sequence	**Challenge**	**Frames**	**Video frames**

Sequence	Challenge	Frames	Video frames
Data11	Smooth movement, clutter	101	
Data12	Smooth movement, clutter, occlusion	101	
Bouncing1	Fast & unexpected movement, deformation	654	
Table Tennis	Unexpected movement	138	
Animal	Fast motion, clutter	71	

Sequence	Challenge	Frames	Video frames
Football	Fast motion, clutter	124	
PETS2009	Smooth motion, clutter	221	
Emilio	Fast & unexpected motion, scale changed, occlusion	280	
Tennis Match	Unexpected movement, deformation	1650	
Bouncing2	Fast & unexpected motion, rotation	90	

Sequence	Challenge	Frames	Video frames
Girl	Scale changed, face expression changed, rotation	500	

Table 4.1: Testing video sequences and their challenges.

4.3.2 Experimental Settings

We compared our proposed methods FMCMC-C and FMCMC-S with the following existing methods: conventional MCMC (our implementation), Semi Boosting (SB) (Grabner et al. [2008]), FragTrack (Frag) (Adam et al. [2006]), and IVT (Ross et al. [2008]). SB, FragTrack and IVT rely heavily on rich appearance models to find the target. We selected these to investigate the extent to which our proposed motion model, applied to only a basic appearance model, can provide high-performance tracking.

We used 300 particles, 3 for the thinning interval, 30 for a burn in period and an 8 bin histogram for each colour channel in FMCMC-C, FMCMC-S, and MCMC. In our experiments, these particles in FMCMC-C, FMCMC-S, and MCMC allowed them to converge and produced result consistently via multiple running times.

In MCMC, FMCMC-C and FMCMC-S, the σ of the likelihood function (Equation 3.6) is set to 0.4, parameters of motion model A to $[1.0\ 1.0]^T$, standard deviation of a process noise is σ_u to $\sqrt{8.0}$ and σ_v to $\sqrt{4.0}$ since the target moves in horizontal direction more than in the vertical direction. All values are fixed for all testing video sequences.

The search areas of SB were set to *twice* the target size (i.e. samples extracted from this range are not too far from the target) and of IVT and FragTrack were set 40x40 pixels (the maximum displacement of the centre of the target from one frame to the next). In SB, we used 100 feature selectors. Each selector maintained 10 features.

In IVT, the standard deviation for the noise of the transition model for the bounding box scales along the horizontal and vertical dimensions is 0.005 and 0.005, respectively; the forgetting factor is 0.99; a standard deviation of 0.25 for the observation likelihoods.

All values for parameters for compared trackers (e.g. SB, FragTrack, IVT) are selected as reported by their authors.

4.3.3 Result

Tables 4.2 and 4.3 summarise the results obtained. The numbers in Table 4.2 give the centre location error (in pixels) averaged over all frames of each sequence, i.e the average distance of the predicted bounding box from the centre of the ground truth bounding box. The lower a number is, the better the result. The numbers in Table 4.3 indicate the percentage of successfully tracked frames (score$>$0.5), where the score is defined by the overlap ratio between the predicted bounding box B_p and the ground truth bounding box B_{gt} and calculated as $score = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$ (Everingham et al. [2010]). The higher a number is, the better the result. Each sequence was run three times with each tracking framework. The best result is marked in bold and the second best underlined.

Table 4.2 shows that FMCMC-S performed more accurately on 8 of the 11 sequences and 2 second best, including the two most challenging outdoor examples. FMCMC-C worked best on 4 of the 11 sequences and 3 second best. On artificial data, or sequences containing rigid objects, there was little difference between FMCMC-C and FMCMC-S: most features moved to the same direction and so individual and cluster sampling produce the same directions. When tracking non-rigid objects and in the presence of distractors, however, FMCMC-S performed very well, its sampling strategy increasing the likelihood that it would investigate the target's true direction of motion. FMCMC-C used a local average direction, which approximates but might not correspond to the true motion direction.

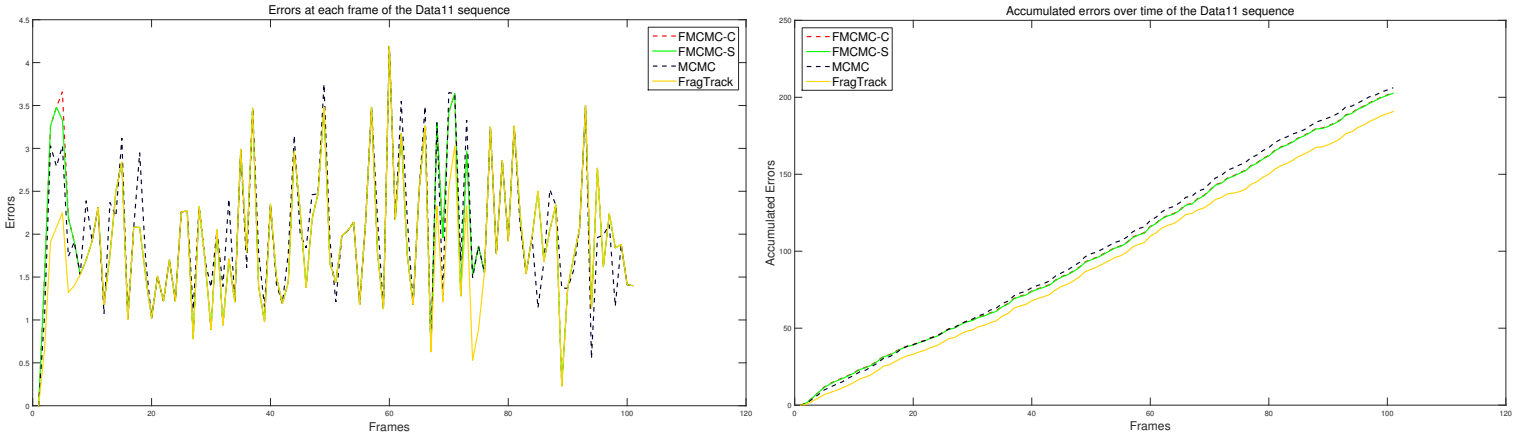
Sequence	FMCMC-C	FMCMC-S	MCMC	SB	Frag	IVT
Data11 (Figure 4.9)	**2.00**	**2.00**	2.04	317.78	1.91	315.45
Data12 (Figure 4.10)	**2.67**	**2.67**	2.90	4.61	3.87	7.93
Bouncing1 (Figure 4.11)	**3.84**	**3.84**	8.78	28.61	4.30	5.49
Bouncing2 (Figure 4.12)	1.94	**1.93**	34.80	216.21	56.56	161.86
Tennis Match (Figure 4.13)	7.16	**7.14**	7.28	141.65	11.83	101.99
Emilio (Figure 4.14)	8.34	**7.79**	8.99	226.87	206.40	68.46
Animal (Figure 4.15)	10.65	10.63	272.67	48.50	62.13	**8.67**
Table Tennis (Figure 4.16)	**3.32**	3.33	3.59	153.26	13.36	251.10
Football (Figure 4.17)	59.22	**8.55**	76.24	60.78	31.32	114.11
PETS2009 (Figure 4.18)	269.91	**4.39**	308.52	180.41	7.44	5.85
Girl (Figure 4.19)	65.99	66.10	36.79	35.54	**6.84**	609.99

Table 4.2: The centre location error (in pixels) averaged over all frames of each sequence. All data were presented in corresponding graphs listed below.

Sequence	FMCMC-C	FMCMC-S	MCMC	SB	Frag	IVT
Data11	**1.00**	**1.00**	**1.00**	0.02	**1.00**	0.04
Data12	**1.00**	**1.00**	**1.00**	0.92	0.93	0.84
Bouncing1	**1.00**	**1.00**	0.94	0.86	0.96	0.99
Bouncing2	**1.00**	**1.00**	0.76	0.21	0.46	0.01
Tennis Match	**0.96**	**0.96**	0.95	0.23	0.55	0.01
Emilio	0.77	**0.81**	0.76	0.08	0.11	0.27
Animal	0.92	0.90	0.07	0.38	0.39	**1.00**
Table Tennis	**1.00**	**1.00**	**1.00**	0.06	0.74	0.14
Football	0.31	**0.67**	0.18	0.06	0.41	0.07
PETS2009	0.24	0.97	0.21	0.19	**0.99**	0.92
Girl	0.15	0.15	0.51	0.40	**0.75**	0.13

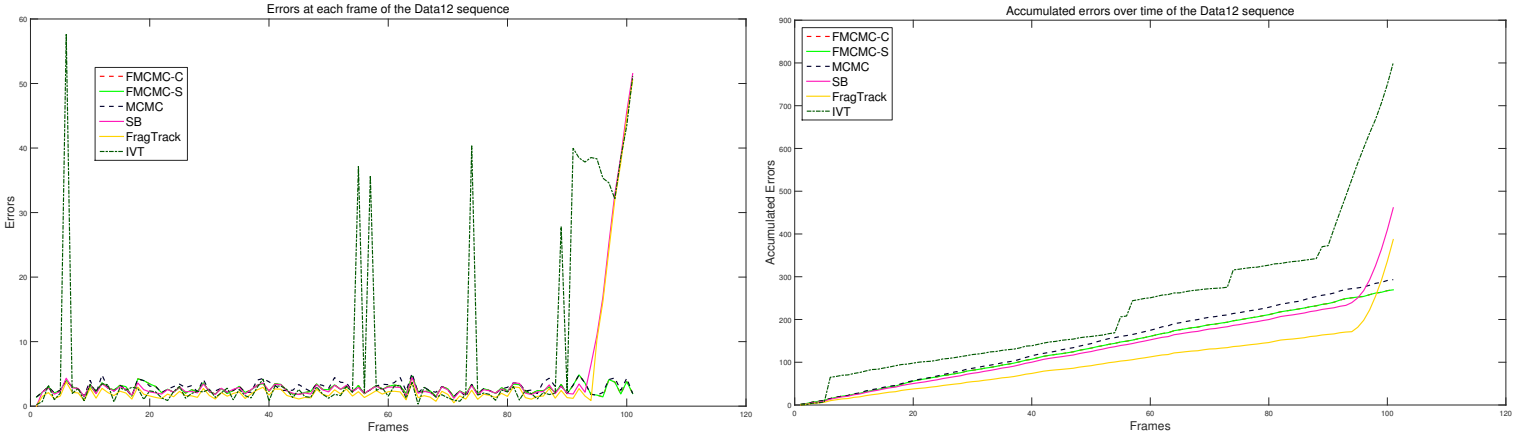
Table 4.3: The overlap ratio between the predicted bounding box and the ground truth bounding box for each testing video sequence.

The following Figures 4.9 - 4.19 show tracking errors for each tracker. Results of some trackers were removed from the figures for a better view because those results had very high errors comparing to others. Tracking results are shown in more details in Appendix C.



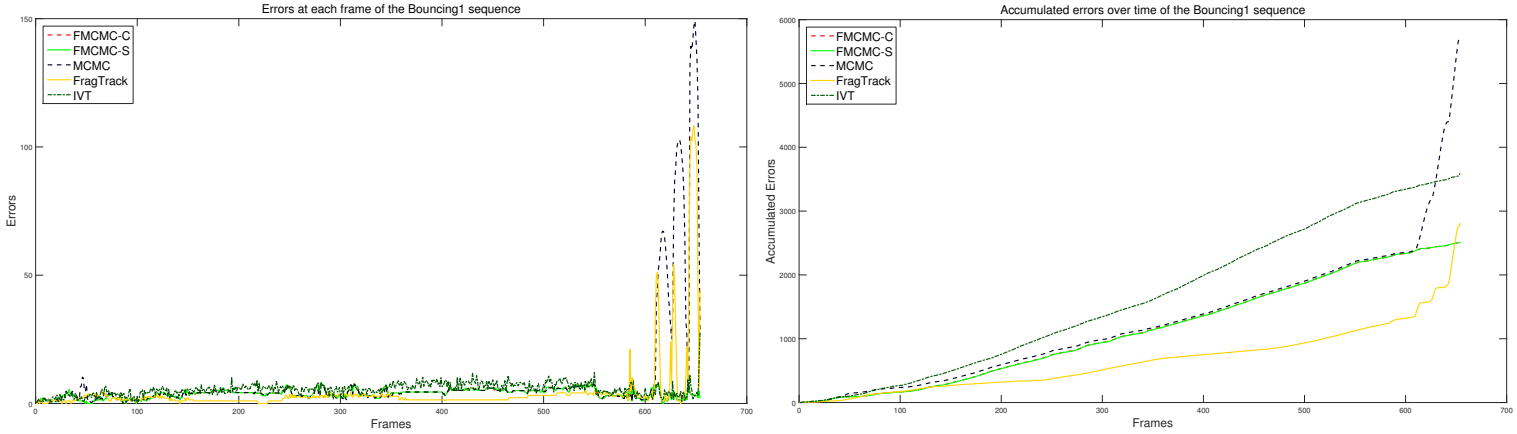
(a) (b)

Figure 4.9: Errors at each frame and accumulated errors over time of trackers for the Data11 sequence.*(Note: IVT, SB were removed because they drifted off the target).*



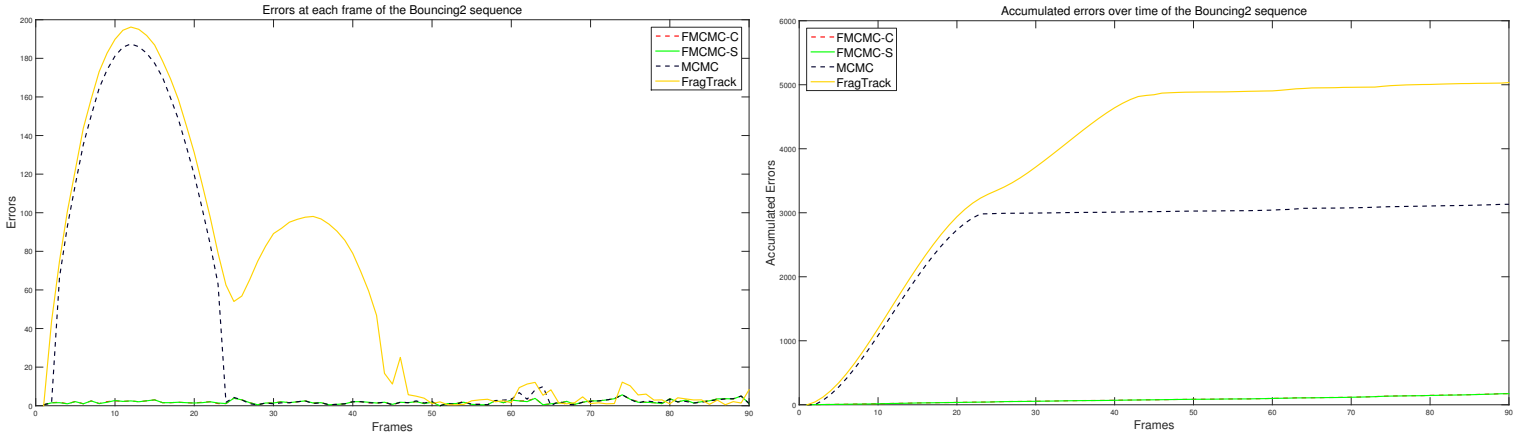
(a) (b)

Figure 4.10: Errors at each frame and accumulated errors over time of trackers for the Data12 sequence.



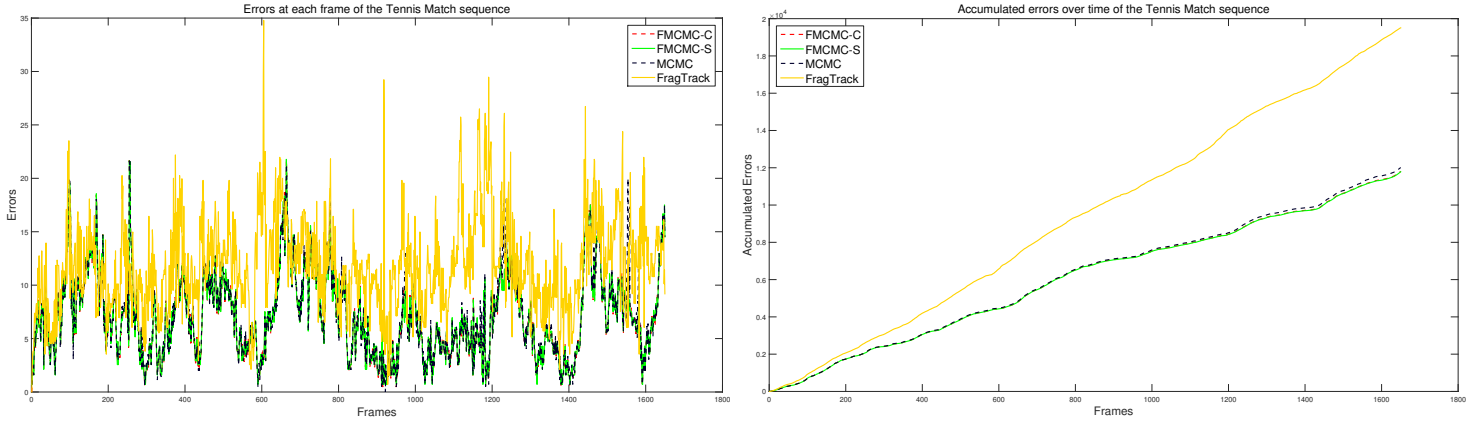
(a) (b)

Figure 4.11: Errors at each frame and accumulated errors over time of trackers for the Bouncing1 sequence.*(Note: SB was removed because they drifted off the target).*



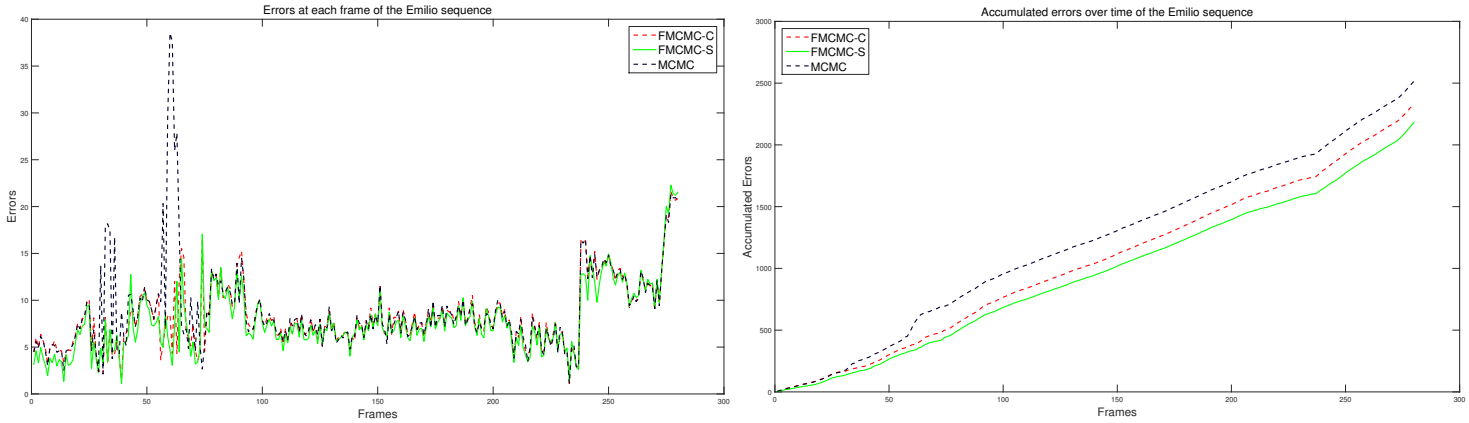
(a) (b)

Figure 4.12: Errors at each frame and accumulated errors over time of trackers for the Bouncing2 sequence.*(Note: IVT, SB were removed because they drifted off the target).*



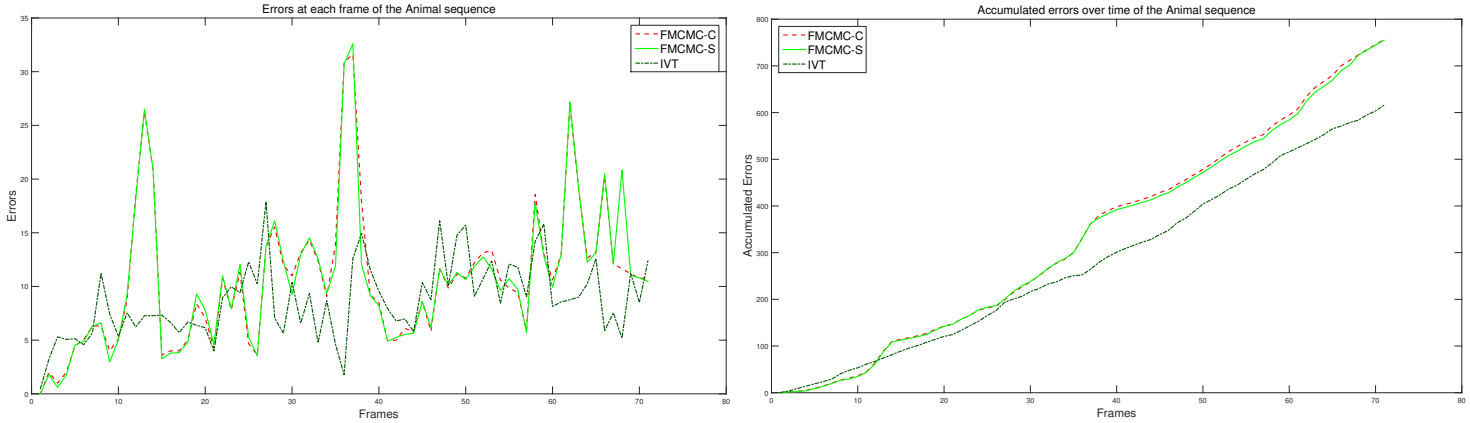
(a) (b)

Figure 4.13: Errors at each frame and accumulated errors over time of trackers for the Tennis Match sequence.*(Note: IVT, SB were removed because they drifted off the target).*



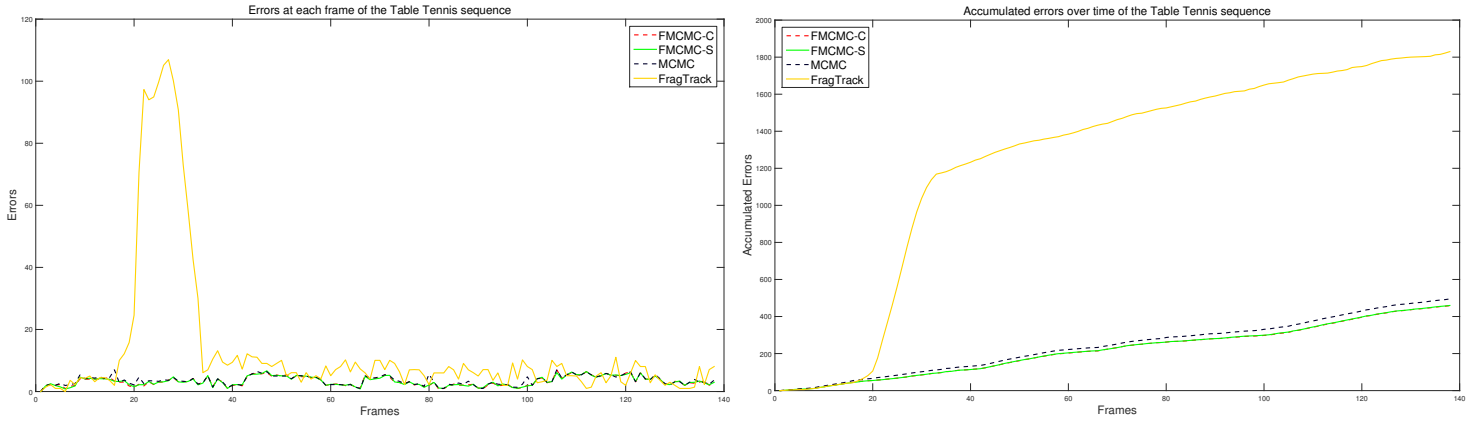
(a) (b)

Figure 4.14: Errors at each frame and accumulated errors over time of trackers for the Emilio sequence.*(Note: IVT, SB, FragTrack were removed because they drifted off the target).*



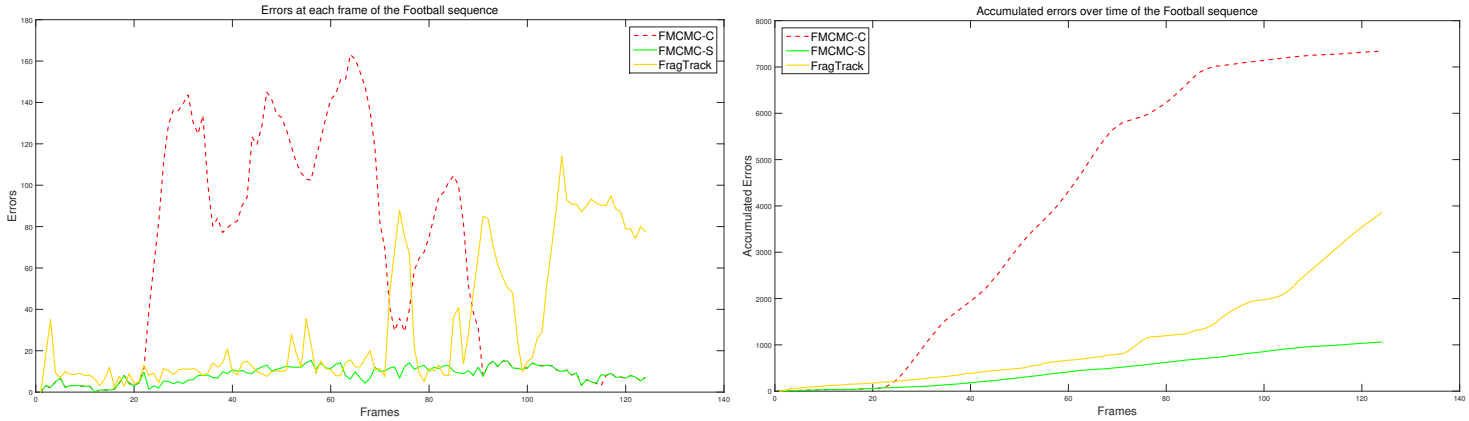
(a) (b)

Figure 4.15: Errors at each frame and accumulated errors over time of trackers for the Animal sequence.*(Note: MCMC, SB, FragTrack were removed because they drifted off the target).*



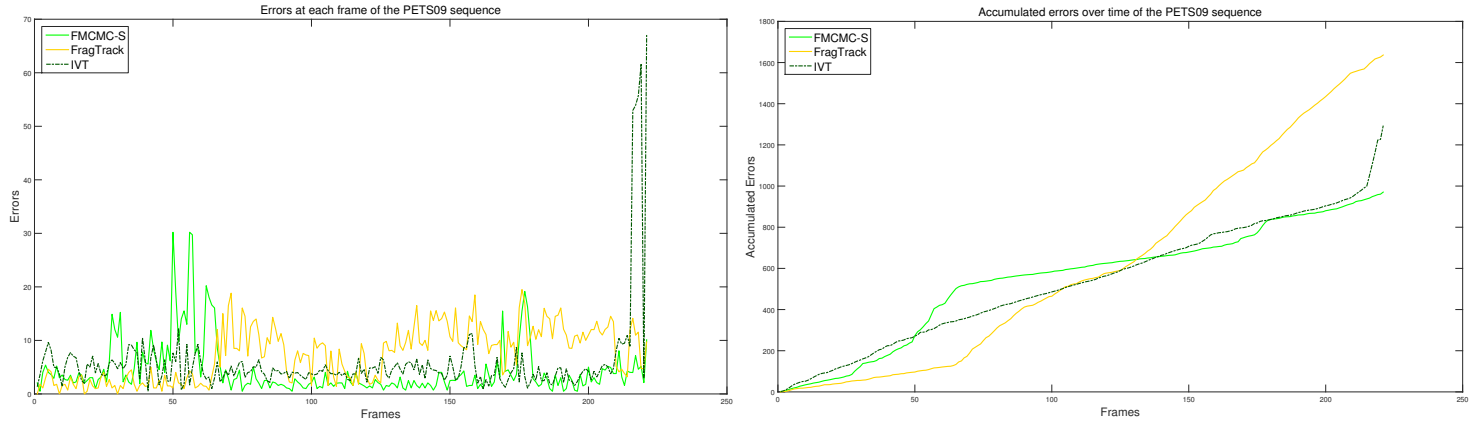
(a) (b)

Figure 4.16: Errors at each frame and accumulated errors over time of trackers for the Table Tennis sequence.*(Note: IVT, SB were removed because they drifted off the target).*



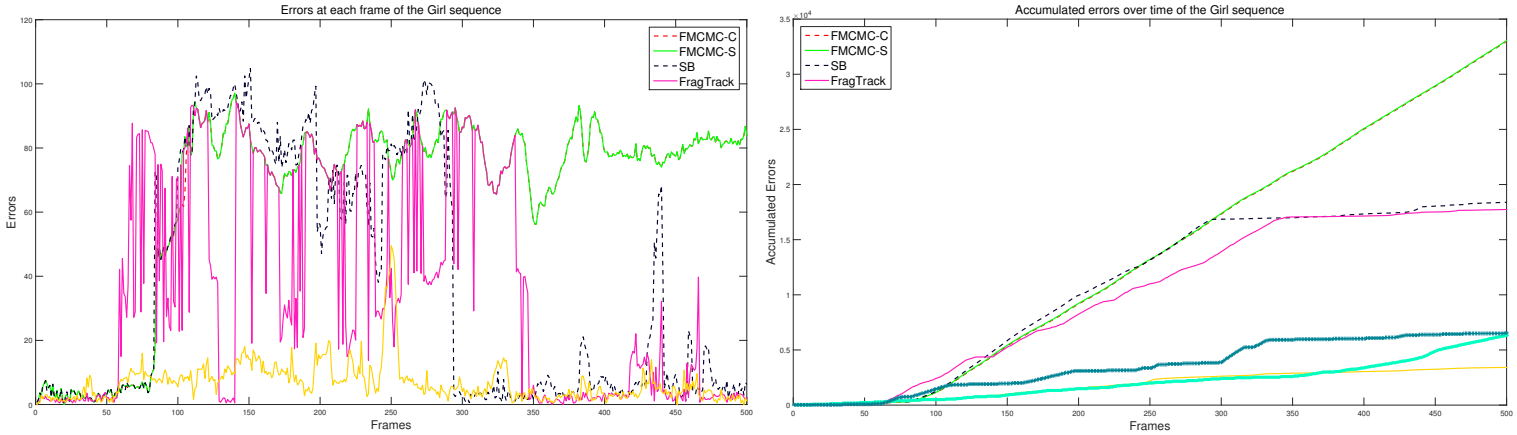
(a) (b)

Figure 4.17: Errors at each frame and accumulated errors over time of trackers for the Football sequence.*(Note: MCMC, SB, IVT were removed because they drifted off the target).*



(a) (b)

Figure 4.18: Errors at each frame and accumulated errors over time of trackers for the PETS09 sequence.*(Note: MCMC, FMCMC-C, SB were removed because they drifted off the target).*



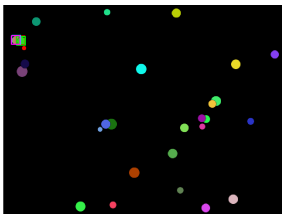
(a) (b)

Figure 4.19: Errors at each frame and accumulated errors over time of trackers for the Girl sequence.*(Note: IVT was removed because they drifted off the target).*

4.4 Discussion

4.4.1 Smooth Motion Handling

SB, IVT, and FragTrack rely on the appearance model to find the target in their search area. Though the target in the Data11 sequence (Figure C.1) is rigid and has a smooth movement and the displacement between two consecutive frames are still inside their search area, SB and IVT failed to track the target in Data11 sequence in the first few frames. They started to track the occluded object at Frame #4 (Figure 4.20)



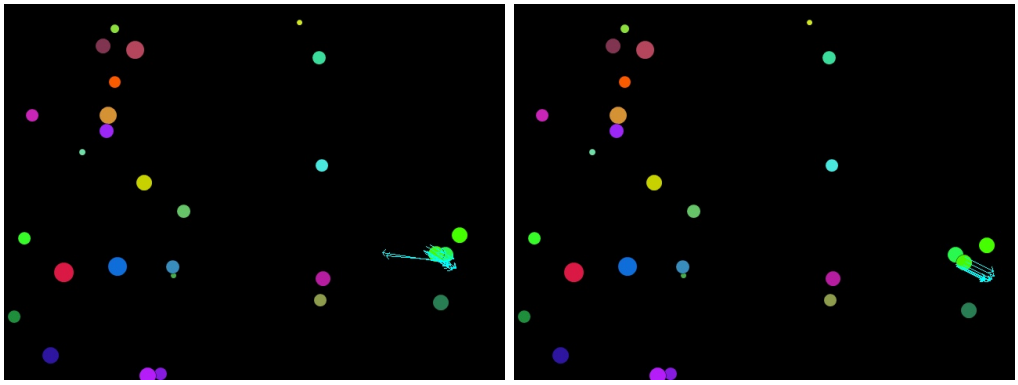
(a) #4

Figure 4.20: Tracking results a selected frame of the Data11 sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

MCMC used a random walk motion model, FMCMC-C and FMCMC-S with a proposed motion model also worked well in the Data11, Data12 sequences. The accuracy of FMCMC-C and FMCMC-S are similar because local motions of features detected are moving in similar directions, though they are more slightly correct than MCMC at several frames as shown in Figure 4.9(a) and Figure 4.10(a).

In the Data12 sequence (Figure C.2), when the target occluded a similar appearance object at Frame #92, some outlier motion directions were introduced (Figure 4.21(a)). FMCMC-C and FMCMC-S still tracked the target well because FMCMC-C discarded these outliers by only considering dominant motion directions. While motion directions with high weight had more chance to be selected in FMCMC-S. With motion directions correctly selected, FMCMC-C and FMCMC-S could avoid the distractor at Frame #96 (Figure 4.21(b)).

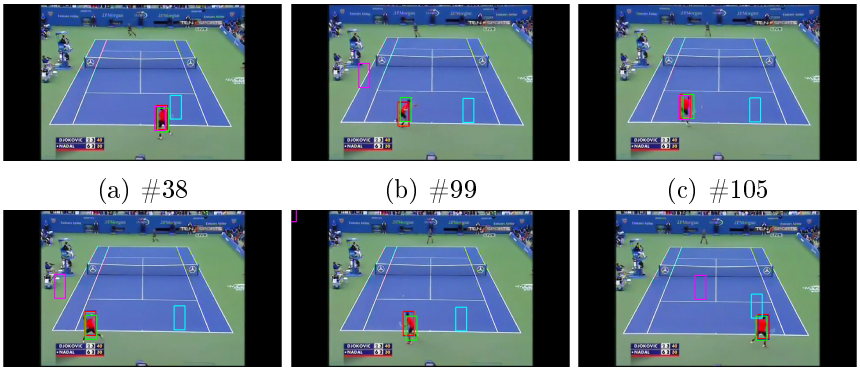
In Tennis Match sequence (Figure C.6), the target does not change its appearance significantly but move smoothly, i.e. it does not quick change its direction or velocity. IVT, however, lost the target at Frame #38 and could not recover the tracking. SB lost the target at Frames #99, #193, #973 (Figure 4.22) when the target changed its pose and re-tracked at Frames #105, #237 (Figure 4.22) when the target returned to the appearance similar to what SB learnt. Therefore, SB and IVT rely on appearance model



(a) Frame #92 (b) #96

Figure 4.21: Local motion directions at selected frames of the Data12 sequence.

to track the target regardless on its motion.



(a) #38 (b) #99 (c) #105

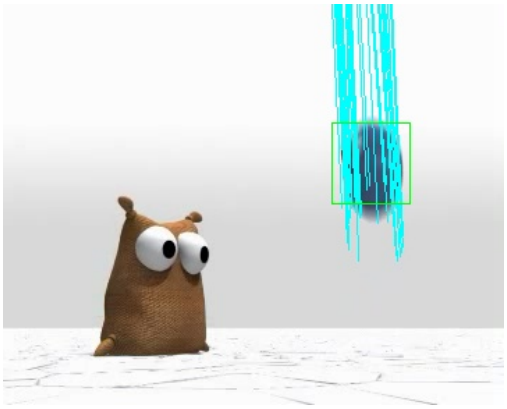
(d) #193 (e) #237 (f) #973

Figure 4.22: Tracking results in selected frames of the Tennis match sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

4.4.2 Unexpected Motion Handling

In Bouncing1 (Figure C.3 and C.4), Bouncing2 (Figure C.5), Emilio (Figure C.7 and C.8) and Animal sequence (Figure C.9), most trackers (e.g. MCMC, FragTrack, SB) suffered when the target moved in unexpected directions and acceleration variations, i.e. the target can change directions and velocities at any time. With the use of feature based motion modelling, FMCMC-C and FMCMC-S, however, predicted target locations correctly.

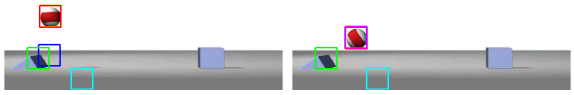
In the Bouncing1 sequence (Figures C.3, C.4), only FMCMC-C and FMCMC-S captured the target at Frame #611 (Figure 4.23) or Frame #643 when the ball jumps up. The remaining trackers could not use a search area large enough to cover the whole target without risking being trapped in local extrema. The two stage (local feature matching - direction selection) approach of the FMCMC algorithms allows a large search space to be used for local motion estimation, safe in the knowledge that the search will be constrained by the linear searches that follow selection. Note that the ball changes shape (deform) during this part of the sequence.



(a) Frame #611

Figure 4.23: Local motion directions at a selected frame of the Bouncing1 sequence.

In the Bouncing2 sequence (Figure C.5), MCMC and FragTrack lost the target at Frame #3 (Figure 4.24) when the ball quickly moved up and they tracked the target by chance at Frame #24 (Figure 4.24).



(a) #3 (b) #24

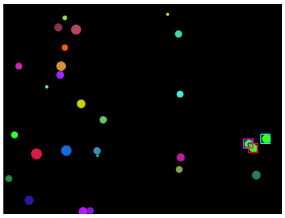
Figure 4.24: Tracking results in selected frames of the Bouncing2 sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

Recently, Kwon and Lee (Kwon and Lee [2008]) proposed the Wang-Landau Monte Carlo sampling method to handle abrupt target motion. The image is divided into mul-

tiple subregions, and Density of State (DoS) used to control jumps from one subregion to another. DoS allows the tracker to spend more time in subregions with a higher likelihood of containing the target. The search used here is, in comparison, less flexible. While the use of carefully selected linear search areas allows our method to track effectively, it would be interesting to incorporate the Wang-Landau method into our algorithms. We anticipate that this would lead to further performance gains.

4.4.3 Distractor Handling

In Data12 sequence (Figure C.2), SB, IVT, and FragTrack worked well until Frame #95 (Figure 4.25) because they locked on distractors whose appearance are most similar to their target.



(a) #96

Figure 4.25: Tracking results in a selected frame of the Data12 sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

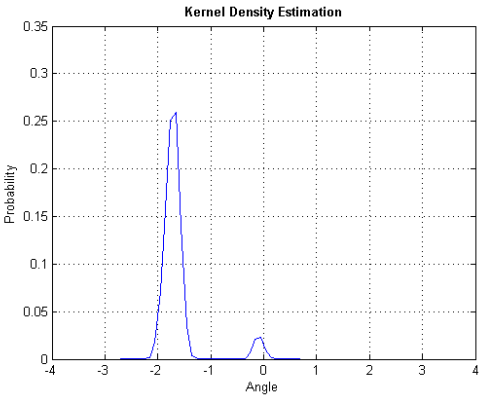
MCMC used a kernel based colour histogram to model the appearance of the target. Colour histograms record colour distribution but lose spatial information. Trackers using this representation can easily be distracted by other objects with the same colour distribution. This was demonstrated in the PETS2009 (Figure C.13) and Football sequences (Figure C.11). In the Football sequence, the football, socks and shorts of the player have similar appearance. SB and MCMC locked onto the player's ankle. FMCMC-C estimated the football's motion direction wrongly because it took the average direction, which also indicated the player's ankle (Figure 4.26a).

Figure 4.26 explains why FMCMC-S performs well on the Football sequence. Figure 4.26a shows the initial local motion estimates. Figure 4.26b shows the KDE resulting from these local motions. During motion direction sampling, most of the selections (around 90% from Accumulated Probability) will be angles in the range (-1.9;-1.5) radians. These point downwards, towards the ground beneath the ball, rather than towards the player's ankle.

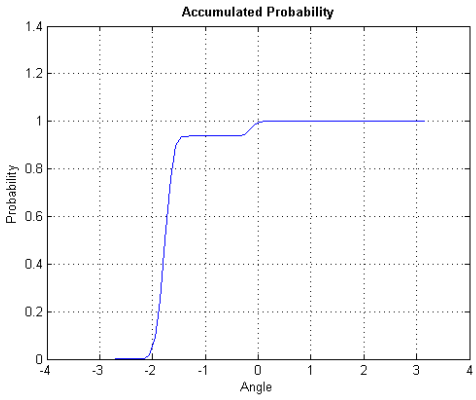
FragTrack tracked the target better, but became trapped on the player's socks and

shorts in frames #55, #72, #86, #93 (Figure 4.27) and mis-located the target in Frames#52, #72 - #76 (Figure 4.27) when the ball changed direction.

(a) Features detected

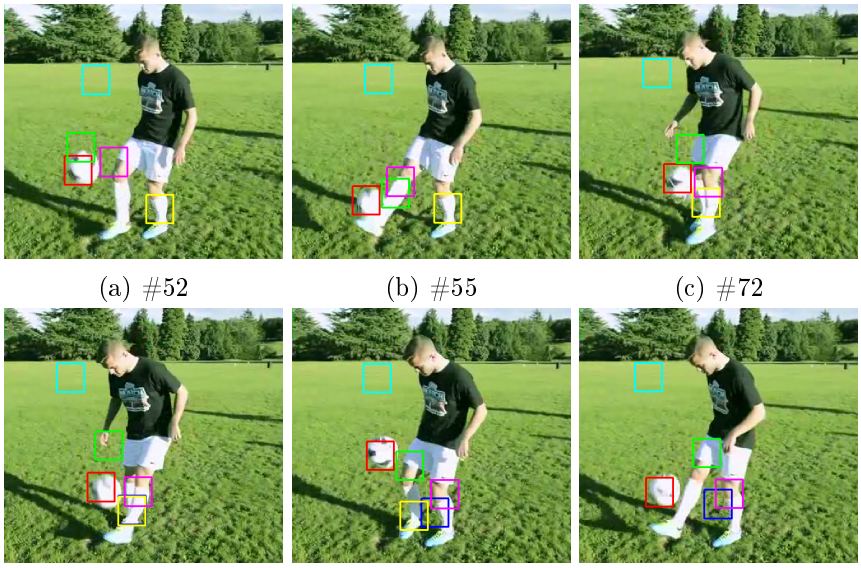


(b) KDE



(c) Accumulated Prob.

Figure 4.26: KDE at Frame #22 of the Football sequence. Angles are calculated in radian.



(a) #52 (b) #55 (c) #72

(d) #76 (e) #86 (f) #93

Figure 4.27: Tracking results in selected frames of the Football sequence(Part 2). MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

(a) #70 (b) #80 (c) #186

Figure 4.28: Tracking results in selected frames of the PETS09 sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

In PETS2009 (as shown in Figure C.13), FMCMC-C was distracted by the person in the crowd since it estimated the movement of the target incorrectly as shown in Figure C.13j. FragTrack could track the target correctly since it represented the target by multiple patches which contain spatial information. In frames #70 to #186 (Figure 4.28), however, the target slightly changed its size that made FragTrack estimated incorrectly the target. FMCMC-S could track the target more precisely at Frame #56 because it performed a restricted search in the true direction of motion of its target (Figure 4.29).

Figure 4.29: Local motions at Frame #56 of the PETS2009 sequence.

4.4.4 Occlusion Handling

IVT is very sensitive to partial occlusion, e.g. in the PETS2009 sequence (Figure C.13), and could not handle the pose changes in the Tennis sequence well (Figure C.6). It lost the target and could not recover it. As the targets in these video sequence display variable motion they are hard to recover once lost. Moreover, IVT blindly updates the target appearance model, therefore, the appearance model is invalid when it mis-locates

the target and wrongly updates the appearance model.

FMCMC-C and FMCMC-S rely on features detecting and matching to generate the target motion model to support target estimation. The motion model is updated implicitly by re-detecting and matching features. Therefore, if the target is occluded, most of the features detected do not belong to the true target. In consequence, the tracking performance decreases.

4.4.5 Appearance Change Handling

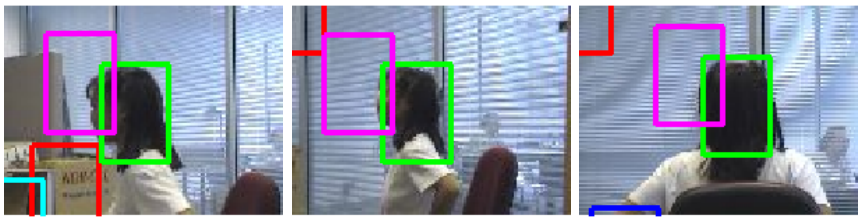
FMCMC-C and FMCMC-S assume that during tracking, the target appearance does not become significantly different from the one learnt at the beginning of the sequence. Adaptive appearance model should be considered. This is demonstrated in the Girl sequence (Figure C.14). FMCMC-C and FMCMC-C lost the target since Frame #90 (Figure 4.30). They could not relocate the target because their motion models were updated by features not belonging to the true target. MCMC, however, had a chance to re-track the target at Frame #303 (Figure 4.30) because it used a random walk motion model and the target was not too far away where the tracker drifted off the target.

FragTrack represents the target appearance more flexible by dividing the target into multiple parts voting for the target location. Multiple part approach can help FragTrack locate though it does not update the target appearance. FragTrack, however, can fail to track the target if the target appearance changes significantly and are not inside its search area.

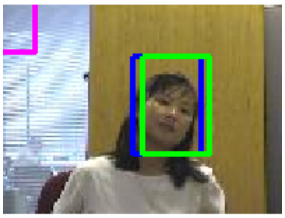
SB could not handle well when the target deformed because SB used an online semi-supervised boosting method. It could work well in the Data12 (Figure C.2), Bouncing1 sequence (Figure C.3) where the target does not change appearance much compared to the appearance at the first frame. It also could work at several frames of other sequences when the target appearance returned to the appearance that it has learnt before such as in Frames #173, #189 (Figure 4.30) of the Girl sequence (Figure C.14).

4.5 Summary

We have proposed an approach which relies upon the distribution of motion directions of local image features to locate a target during visual tracking. These local motion directions are extracted directly from two consecutive frames and provide information used to guide an MCMC-based search for rigid and deformable objects. Two algorithms, FMCMC-C and FMCMC-S, have been proposed. FMCMC-C only considers the group



(a) #90 (b) #173 (c) #189



(d) #303

Figure 4.30: Tracking results in selected frames of the Girl sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

containing the largest number of features going in similar directions. The search direction is estimated from the average direction of this group. FMCMC-S has the potential to search any and all possible directions making up the motion direction distribution.

Experiments showed the FMCMC-S algorithm to have performance advantages over other trackers relying on rich appearance models. There is little difference in accuracy when FMCMC-S and FMCMC-C tracked rigid objects on artificial and recorded video sequences. When tracking non-rigid targets, FMCMC-S outperformed FMCMC-C because it allowed the tracker to investigate more motion directions and increase the chance that some of which are close to the target's true motion direction. Moreover, the FMCMC-S algorithm can handle target motion variations without using any more prior knowledge of movement than FMCMC-C. In constrast, SB and FragTrack do not have motion models and blindly search the target by detecting a location with highest confidence score. So, they do not handle motion variations explicitly.

In the presence of distractors, other objects whose appearances are similar to the target, MCMC and SB easily get distracted. In FMCMC-S, the search is guided by following directions which have the high probability as the target movement and help the tracker be able to avoid distractors. In some cases, FMCMC-C wrongly estimates

the target motion via an average direction.

If the target appearance changes, a fixed kernel weighted histogram FMCMC-C and FMCMC-S might not be able to follow the target. When occlusion occurs, features detected are not belong to the true target and this makes FMCMC-C and FMCMC-S could not recover to the true target because they rely on feature based motions. These issues are addressed in Chapter 5.

Chapter 5

An Unified Tracking Algorithm

5.1 Introduction

In Chapter 3, the MCMC-based tracking algorithm (MCMC-SA) contains an appearance pool which maintains multiple examples of the target's appearance. The appearance model is updated by modifying the existing histogram models and adding new template-histogram pairs to the appearance pool. Experiments have shown that it can handle appearance changes well if the target moves smoothly. When the target movement is complex, however, the algorithm locates the target incorrectly. Another issue is confidence scores returned by NCC can result in mis-locating the target.

On the other hand, the FMCMC tracker proposed in Chapter 4 has a motion estimation component which supports multiple linear searches, replacing the random walk search strategy in MCMC. The motion model is updated implicitly by redetecting target features and maintaining a feature pool. Experiments have demonstrated this tracker handles target motion variations well without using any prior knowledge of movement. It, however, assumes the target appearance does not change significantly from the appearance learnt at the beginning of tracking.

In this chapter, a new unified tracking method is proposed which combines the adaptive appearance and motion models developed in previous chapters to utilise the advantages of each: adapting to target appearance changes and capturing target motion to enhance target prediction, which in turn supports appearance model update and learning. During MCMC-based tracking, at each search iteration, a motion direction is sampled directly from the motion direction distribution. The proposal density proposes a new state along the selected direction. The proposal comprises changes in position and an (histogram) appearance model index which is randomly selected from the appearance

pool. The new target location is estimated by identifying particles which have the highest weight and use the most commonly accepted histogram.

During drifting or occlusion, the tracker stops updating the motion models (i.e. updating target features) and re-initialises the tracking process by selecting an appropriate appearance model from the appearance pool.

The rest of this chapter is organised as follows. In Section 5.2, we describe the proposed method. Experimental results are presented in Section 5.3 and discussed in Section 5.4. Finally, conclusions are drawn in Section 5.5.

5.2 Proposed Tracking Algorithm

Figure 5.1 shows the main steps in the proposed method, Feature based Markov Chain Monte Carlo using multiple models (FMCMC-MM). This approach maintains one appearance pool which contains appearance variations learnt during tracking and one feature pool to support target motion estimation.

Given the target boundary in the first frame, the tracker extracts a target template and constructs a corresponding histogram. This template-histogram pair is entered into the appearance pool. Local target features are extracted and stored in the feature pool. During tracking, features in the previous frame are tracked to identify features in the current frame, linking sets of previous and current features to build a motion direction distribution. At each search iteration, a motion direction is sampled directly from the motion direction distribution. An appearance is also sampled from the appearance pool to support the search for the target

After the new target location is estimated, a new template and its corresponding histogram are extracted. A learning appearance step is invoked to decide whether this new template should be added into the appearance model (pool). A feature update process is executed to update the model of the target's motion.

5.2.1 Appearance Model

As in Chapter 3, targets are selected by manual annotation of the first frame in the image sequence. Once the target location is specified, its template is extracted and added to the appearance pool. For each template, an Epanechnikov kernel weighted colour histogram (Comaniciu et al. [2003]) is constructed. To compare the reference histogram p_t of the target with the candidate histogram q_t of the state vector X_t at time t, we use the Bhattacharyya distance. A Gaussian density function (Equation 3.6) is used

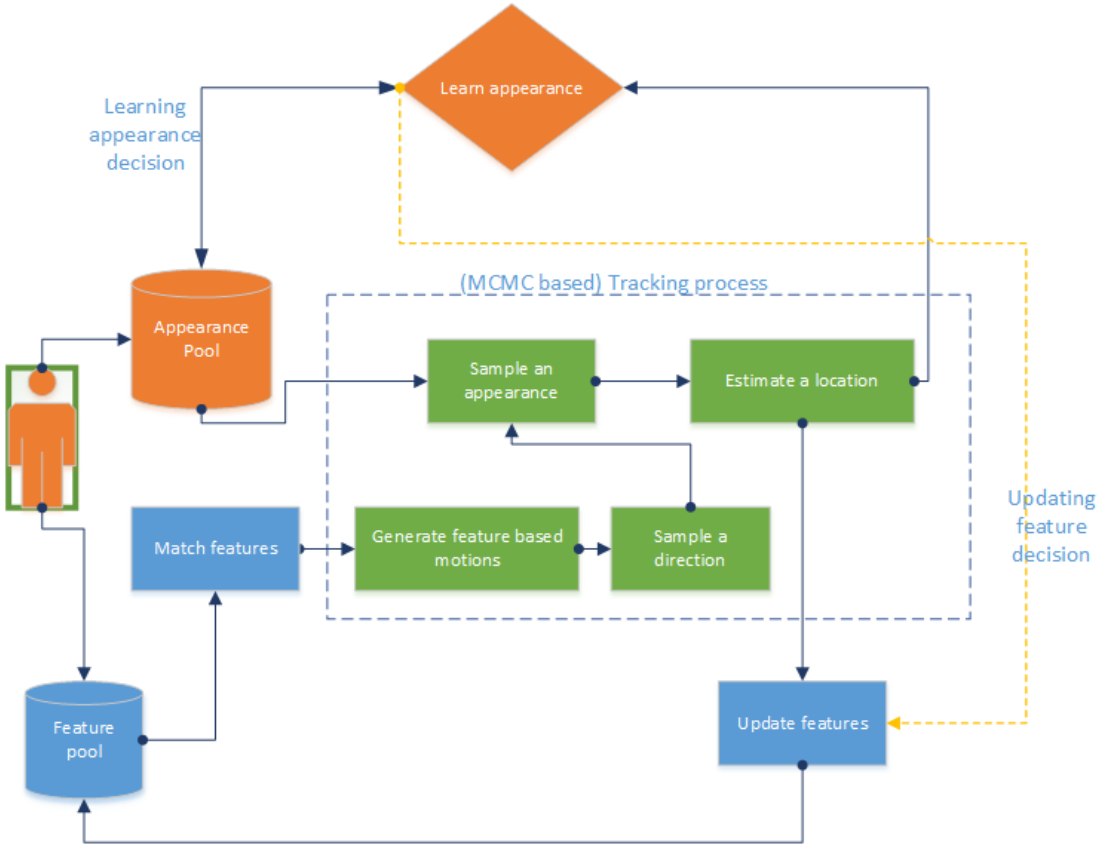


Figure 5.1: Overview of the proposed approach.

for the likelihood function of the measurement histogram. When comparing template and image data or pairs of templates, we use the Normalised Correlation Coefficient (NCC).

5.2.2 Motion Model

Each feature extracted from the target is represented as presented in Section 4.2.2 of Chapter 4. Features are matched using a pyramidal implementation of the Kanade - Lucas – Tomasi tracker (Bouguet [2000]), forming a set of vectors $V_t = \{v_t^i\}_{i=1..m}$ at time t. Gaussian kernel density is applied to estimate the motion direction distribution. The motion direction sampling approach from Section 4.2.3 of Chapter 4 is selected because it is more general and flexible than the framework using one fixed motion direction. It allows the algorithm to search in any and all possible directions without an explicit direction selection step.

5.2.3 Sampling Appearance & Motion Models

The motion and appearance models presented here are embedded into the MCMC method of Khan et al. [2005]. At each time t, an appearance pool containing templates $T_t = \{T^j\}_{j=0..k}$, equivalent generative models $G_t = \{G^j\}_{j=0..k}$ and feature pool $F_t = \{F^p_{t-1}, F^c_t\}$ are given, where k is the current size of the pool.

In this approach, *three* pieces of information are used when predicting target location. *First*, the previous target location is used to decide the centre of the search area. The search area S is double the target size. *Second*, the confidence score matrix $C^j = NCC(T^j, I)$ is calculated by using NCC to compare each template T^j from T_t to each location $I(x, y)$ of image sequence I belonging to S. *Third*, features matched from the previous image are used to improve the initial location of an MCMC chain. Define $m^f = \sum_{i=1}^{m}(f^i \subset P)$ as the number of features in the current frame belonging to an image patch P defined by the target's bounding box.

Tracking begins with the initialisation of an MCMC chain. The starting position is determined where the confidence score at that location $C^j(x, y) \geq \theta_d$ and contains the maximum number of m^f. Simply taking the location with the maximum confidence score can cause mis-locations as demonstrated in experiments of Chapter 3. Integrating target features enhances the predicted position, using the threshold θ_d reduces the effect of image noise. If no location satisfies these conditions because no available templates produce a confidence score which is greater than θ_d, the starting position is determined using the previous target location. The initial appearance model is the histogram associated with the template that best matches the last recorded target location.

As the MCMC chain progresses, new states X'_t are proposed according to the proposal density $Q_1(X'_t, X_t)$. The proposal from Q_1 comprises changes in position according to the motion model, from which a motion direction is randomly selected (Section 5.2.2). The target is sought along the selected direction. New states X''_t are proposed using the proposal density $Q_2(X''_t, X'_t)$. The proposal from Q_2 comprises changes in position according the motion direction selected and an appearance model (histogram) randomly selected from the appearance pool. Each appearance model has an associated weight, which records the number of times it was selected and accepted within the chain.

At the end of the MCMC process, the most highly weighted appearance model is identified. The particle generated using the model that has the best fit to the local image data provides the new estimate of target location. The motion direction sampling is then reapplied and templates matched to the estimated location to initialise processing into the next time frame. The tracking process is described in Algorithm 14.

5.2.4 Updating Appearance Model

Updating an existing model and adding a new model have been discussed in Section 3.2.4 of Chapter 3.

5.2.5 Handling Occlusion & Re-detecting the Target

Occlusion detection has been mentioned in Section 3.2.5 of Chapter 3. The occlusion detection step is necessary because the target motion relies on feature detection. Successful occlusion detection prevents features lying on the occluding object over-ruling those belonging to the true target.

5.2.6 Updating Motion Model

The target motion model depends on feature detection and matching. Features help the tracker handle motion variation and abrupt motion naturally by allowing the tracker develop a good sense of where the target might be. The features used should be updated as tracking progresses, as some will become invisible and others appear over time. Features are only updated if there is no occlusion.

A bounding box does not always provide a good fit to the target boundary, and some detected features may be outliers, i.e. belong to the local background. The motion direction sampling method can overcome this problem, assuming that most of the features considered lie within the true target boundary.

5.2.7 Algorithm

Given a target state $X_t = \{x_t, d_t, s_t\}$ where $x_t = (u_t,\ v_t)$ is the target location, d_t is the direction considered, $s_t \in (1..k)$ is the selected appearance model at time t, k is the number of appearance models in the appearance pool. X_t is the most likely state at time t of the target, $X'_t = \{x'_t, d'_t, s'_t\}$ and $X''_t = \{x''_t, d''_t, s''_t\}$ are the most likely state at time t of the target within a selected linear segment. The tracking process is then as described in Algorithm 14.

The proposal densities $Q_1(X'_t; X_t)$, $Q_2(X''_t; X'_t)$ are designed by

$$Q_1(X'_t; X_t) = P(X'_t|X_t) = P(x'_t|x_t)P(d'_t|KDE). \tag{5.1}$$

$$P(x'_t|x_t) \sim x'_t = A \cdot x_t + N(0, \sigma). \tag{5.2}$$

$$Q_2(X_t''; X_t') = P(X_t''|X_t') = P(x_t''|x_t', d_t')P(s_t''|W). \tag{5.3}$$

$$P(x_t''|x_t', d_t'') \sim \begin{cases} u_t'' = u_t' + N(0, \sigma_u) \\ v_t'' = \tan(d_t') \cdot u_t'' + b. \end{cases} \tag{5.4}$$

$$b = v_t' - u_t' \cdot s. \tag{5.5}$$

$$s = \tan(d_t'). \tag{5.6}$$

where $P(x_t'|x_t)$ describes the changes of the location, A are constants (i.e an identity matrix) and a Gaussian noise with zero mean $N(0, \sigma)$. The proposal density $Q_1(X_t'; X_t)$ considers the changes in location x_t' and the direction d_t'. The s_t' of X_t' are similar as s_t of X_t.

KDE contains a motion direction distribution constructed by feature based motions, $P(d_t'|KDE)$ presents a motion direction randomly selected as described in Algorithm 11.

$P(s_t''|W)$ describes an appearance model randomly selected as described in Algorithm 8, s (Equation 5.6) and b (Equation 5.5) are the slop (i.e. gradient) and intercept respectively of the selected line, W is a set of each histogram model's weight in the appearance pool.

In summary, given a state X_t, sample one direction from the KDE and one position X_t' belonging to that direction using Q_1. Within the sampled direction, search for the best state X_t''. During the search, one appearance model of the target state is sampled to be evaluated with the current image data at the sampled position.

Define B as the burn in period, M as the thinning interval before accepting one particle, N_l as the number of particles used to search one line, L as the total number of lines considered, $T = \{T^i\}_{i=0}^k$ as a pool of templates and $G = \{G^i\}_{i=0}^k$ as a list of corresponding histogram models.

Algorithm 14 Multiple appearance models and motion direction sampling (FMCMC-MM).

1. Detect and match features and compute the motion direction distribution KDE as described in Section 5.2.2
2. Initialise the start state X_t for the target using features detected and templates in the pool as described in Section 5.2.3.
3. Initialise equal weight for each histogram model.
4. Repeat ($l = 1..L$) times
   (a) Propose a new location x'_t according to $Q_1(X'_t; X_t)$.
   (b) Randomly select one direction from the KDE of the target.
   (c) Calculate the slope s of the selected direction.
   (d) Calculate the intercept b for the line using the slope s and new X'_t.
   (e) Repeat $B + M \times N_l$ times
      i. Generate x''_t of X''_t from x'_t according to the s and b using $Q_2(X''_t; X'_t)$.
      ii. Propose a candidate appearance model for X''_t according to the appearance weight.
      iii. Compute the acceptance ratio $a = \frac{P(X''_t|Z_t)Q_2(X'_t;X''_t)}{P(X'_t|Z_t)Q_2(X''_t;X'_t)} \approx \frac{P(Z_t|X''_t)}{P(Z_t|X'_t)}$
      iv. If $a \geq 1$, then accept X''_t: Set the target in X'_t to X''_t, increase the weight for the selected histogram and update the cached likelihood. Otherwise, accept with probability a. If rejected, leave X'_t unchanged.
   (f) If the state X'_t is better than X_t then move X_t to X'_t (i.e. $P(Z_t|X'_t) \geq P(Z_t|X_t)$). Otherwise, keep X_t unchanged.
5. The set of particles is obtained by storing N_l best particles at each direction.
6. The current posterior $P(X_t|Z_{1:t})$ is approximated by using MAP.
7. Check if the target is in occlusion as in Section 5.2.5.
8. Update the target model as in Section 5.2.4.
9. Re-detect features for the target (i.e. update motion model).

5.3 Experiments and Results

5.3.1 Data

We used the test video sequences described in Tables 3.1, 4.1 and a new set of video sequences summarised in Table 5.1 for experimental evaluation, all synthesised by Wu et al. [2013] from recent literature, except the Hand sequence which was from (AVSS2007). The idea is to compare the performance of all proposed tracking methods developed in Chapter 3 and Chapter 4. Ground truth data was generated by manually annotating the sequence, capturing the visible part of the target with a rectangular bounding box.

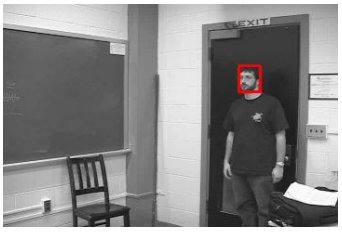
Sequence	Challenge	Frames	Video frames
Tiger1	Fast motion, target rotates, occlusion, appearance deformed	354	
Freeman1	Scale changed, Face expression changed, rotation	326	
Hand	Fast motion, Deformation, Clutter, Scale changed	334	

Table 5.1: Testing video sequences and their challenges.

5.3.2 Experimental Settings

We compared our new proposed method FMCMC-MM to MCMC-SA developed in Chapter 3, FMCMC-C and FMCMC-S implemented in Chapter 4 and other existing methods: conventional MCMC (our implementation), Template matching (TT) (our implementation), Online AdaBoost (OAB) (Grabner and Bischof [2006]), Semi Boosting (SB) (Grabner et al. [2008]), FragTrack (Frag) (Adam et al. [2006]), IVT (Ross et al. [2008]) and Visual Tracking Decomposition (VTD) (Kwon and Lee [2010]).

OAB, SB, FragTrack and IVT rely heavily on rich appearance models to find the target. VTD contains multiple basic observation model (4 in their implementation), representing specific appearances of the target and constructed by sparse principle component analysis (SPCA), and multiple basic motion models (2 in their implementations), covering different motion types, to form multiple trackers. Each tracker contains one basic observation model and one basic motion model. This tracker was selected because it used sampling methods to generate appearance and motion models to construct trackers. Although their approach is different from ours, their sampling strategy is similar.

All settings for FMCMC-MM, MCMC-SA, FMCMC-C, FMCMC-S, MCMC, TT were as in previous chapters. We used 300 particles, 3 for the thinning interval, 30 for a burn in period and an 8 bin histogram for each colour channel in MCMC-SA and MCMC.

In FMCMC-MM and MCMC-SA, we set the parameters of motion model A to $[1.0\ 1.0]^T$, standard deviation of a process noise σ_u to $\sqrt{8.0}$ and σ_v to 2.0. The σ of the likelihood function (Equation 3.6) is set to 0.4 which allows the function to return values between $(0; 1)$. Thresholds to allow update of a histogram $\theta_{max} = 0.95$; Thresholds to detect the target $\theta_d = 0.4$; Thresholds to detect occlusion $\theta_{tc} = 0.1$ and $\theta_{hc} = 0.6$; Thresholds to add a new (template + histogram) model between $(\theta_{min}, \theta_d) = (0.17, 0.4)$.

In Template tracking (TT), the threshold is set 0.4 and the search area is the whole image.

The search areas of OAB and SB were set to *twice* the target size (i.e. samples extracted from this range are not too far from the target) and of FragTrack and IVT were set 40x40 pixels (the maximum displacement of the centre of the target from one frame to the next). In OAB and SB, we used 100 feature selectors. Each selector maintained 10 features.

In IVT, the standard deviation for the noise of the transition model for the bounding box scales along the horizontal and vertical dimensions is 0.005 and 0.005, respectively; the forgetting factor is 0.99; a standard deviation of 0.25 for the observation likelihoods.

Parameter values for SB, OAB, IVT, FragTrack and VTD are as reported by their

authors.

5.3.3 Result

Tables 5.2 and 5.3 summarise the results obtained. The numbers in Table 5.2 give the centre location error (in pixels) averaged over all frames of each sequences, i.e the average distance of the predicted bounding box to the centre of the ground truth bounding box.

The numbers in Table 5.3 indicate the percentage of successfully tracked frames (score >0.5), where the score is defined by the overlap ratio between the predicted bounding box B_p and the ground truth bounding box B_{gt} and is calculated as $score = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$ (Everingham et al. [2010]).

Each sequence was run three times with each tracking method. The best result is marked in bold and the second best underlined. Note that, (X) in the cell of Table 5.2 and Table 5.3 means that this tracking method could not be applied to this video because runtime errors occurred in the original implementation.

Table 5.2 shows that FMCMC-MM performed more accurately on 10 of the 22 sequences and was second best 4 times. MCMC-SA was second best on 3 of the 22 sequences. FMCMC-C performed more accurately on 2 of the 22 sequences and was second best on 6. FMCMC-S performed more accurately on 2 of the 22 sequences and was second best on 9.

Table 5.3 shows that FMCMC-MM overlapped the true target to a greater degee on 11 of the 22 sequences and was second best on 9. MCMC-SA overlapped the true target precisely on 4 of the 22 sequences and was second best on 2 sequences. FMCMC-C overlapped the true target precisely on 7 of the 22 sequences and scored on one second best. FMCMC-S overlapped the true target more precisely on 6 of the 22 sequences and was second best on 4.

Overall, FMCMC-MM outperformed in most of testing video sequences. Tracking results are listed in Appendix D. The following Figures 5.2 - 5.23 show tracking errors in each video sequence for each tracker. Results of some trackers were removed from the figures for a better view because those results had very high errors comparing to others.

Sequence	FMCMC-MM	FMCMC-C	FMCMC-S	MCMC	SB	Frag	IVT	VTD	OAB	MCMC-SA	TT
Data11 (Figure 5.2)	2.71	2.00	2.00	2.04	317.79	**1.91**	315.45	2.47	319.59	2.66	13.01
Data12 (Figure 5.3)	3.17	2.67	2.67	2.90	4.61	3.87	7.93	**2.07**	5.40	5.08	7.54
Bouncing1 (Figure 5.4)	**3.00**	3.84	3.84	8.78	28.61	4.30	5.49	11.87	28.07	7.12	**3.00**
Table Tennis (Figure 5.5)	3.46	**3.32**	3.33	3.59	153.26	13.36	251.10	380.51	642.12	3.54	46.75
Emilio (Figure 5.6)	**6.67**	8.34	7.79	8.99	226.87	206.40	68.46	20.30	235.37	8.46	22.29
Tennis Match (Figure 5.7)	**7.14**	7.16	**7.14**	7.28	141.65	11.83	101.99	(X)	13.59	9.36	11.01
Animal (Figure 5.8)	11.12	10.65	10.63	272.66	48.50	62.13	**8.67**	208.14	361.61	14.22	55.45
Football (Figure 5.9)	**5.73**	59.22	8.55	76.24	60.78	31.32	114.11	34.92	60.56	6.48	98.30
PETS2009 (Figure 5.10)	5.81	269.91	4.39	308.52	180.41	7.44	5.85	**3.23**	258.89	6.32	146.03
Bouncing2 (Figure 5.11)	2.32	1.94	**1.93**	34.80	216.21	56.56	161.86	153.59	152.34	2.38	43.93
Rolling Ball (Figure 5.12)	**5.66**	6.33	6.40	6.34	168.81	8.84	98.79	33.24	159.21	6.30	47.13
Doll (Figure 5.13)	**6.39**	8.51	11.17	9.78	141.28	10.22	122.95	(X)	151.76	9.75	16.48
David2 (Figure 5.14)	**2.11**	4.25	5.48	5.74	15.47	55.07	67.39	3.60	4.78	3.55	137.39

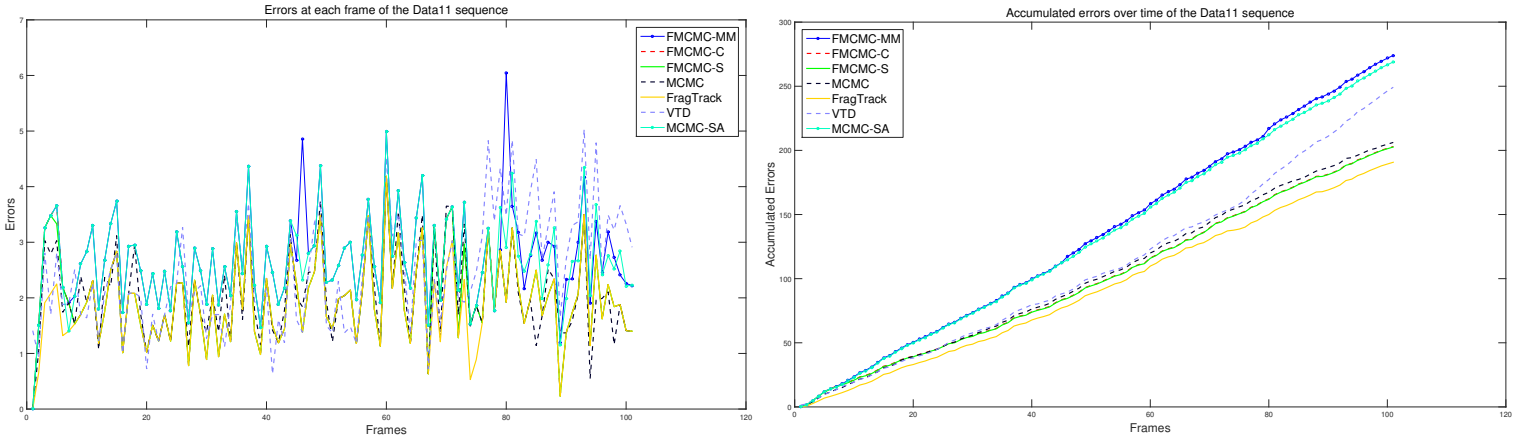
Table 5.2 – continued from previous page

Sequence	FMCMC-MM	FMCMC-C	FMCMC-S	MCMC	SB	Frag	IVT	VTD	OAB	MCMC-SA	TT
Boy (Figure 5.15)	2.67	4.31	7.90	105.43	235.01	39.19	210.88	2.68	**2.63**	4.19	11.93
Jogging (Figure 5.16)	**5.08**	160.51	160.51	29.66	55.98	15.55	90.84	92.40	161.31	7.37	13.32
Jumping (Figure 5.17)	8.89	53.18	15.28	111.17	77.93	**6.38**	158.79	71.83	196.15	52.60	10.28
Girl (Figure 5.18)	6.76	65.99	66.10	36.79	35.54	6.84	609.99	7.28	**3.49**	12.64	13.04
Bird2 (Figure 5.19)	15.78	22.02	22.14	22.54	174.02	29.03	164.07	111.83	**7.59**	31.65	91.37
Cup (Figure 5.20)	5.18	**4.61**	4.63	4.62	54.59	8.73	154.62	5.36	159.09	4.80	79.86
Hand (Figure 5.21)	**6.27**	56.65	13.11	54.03	127.28	94.08	95.10	86.86	124.87	47.28	47.73
Tiger1 (Figure 5.22)	**23.43**	24.35	23.77	24.52	122.26	63.17	280.84	109.22	63.25	29.71	41.18
Freeman1 (Figure 5.23)	14.01	16.99	17.63	93.15	93.31	**10.07**	854.54	10.64	149.42	11.72	75.65

Table 5.2: The centre location error (in pixels) averaged over all frames of each sequence. All data were presented in corresponding graphs listed below.

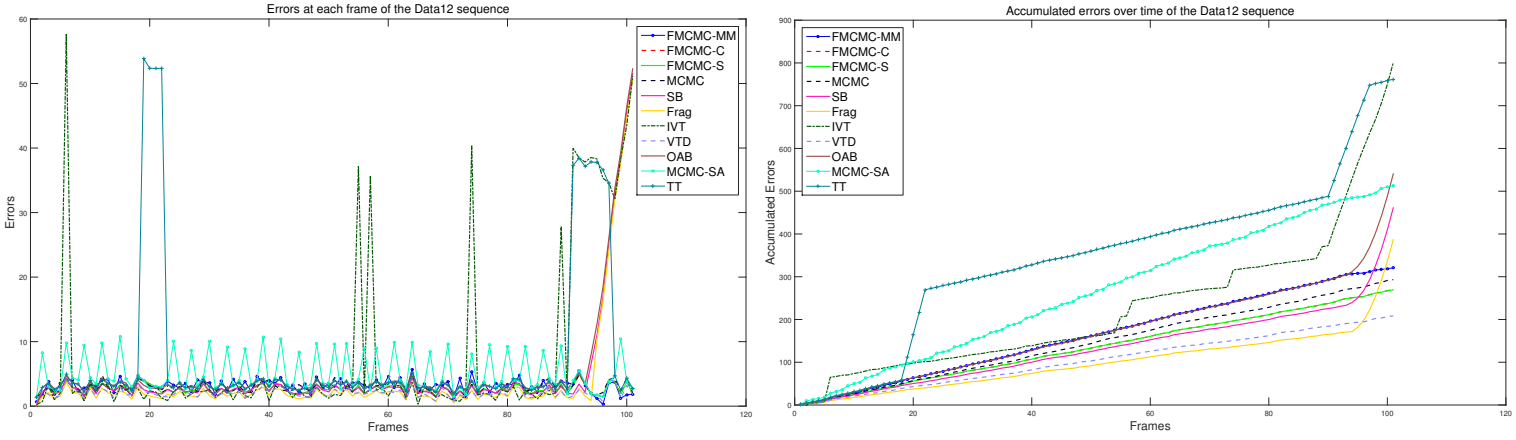
Sequence	FMCMC-MM	FMCMC-C	FMCMC-S	MCMC	SB	Frag	IVT	VTD	OAB	MCMC-SA	TT
Data11	0.99	**1.00**	**1.00**	**1.00**	0.02	**1.00**	0.04	**1.00**	0.02	**1.00**	0.93
Data12	0.98	**1.00**	**1.00**	**1.00**	0.92	0.93	0.84	**1.00**	0.92	0.70	0.89
Bouncing1	**1.00**	**1.00**	**1.00**	0.94	0.86	0.96	0.99	0.93	0.88	0.98	**1.00**
Table Tennis	0.99	**1.00**	**1.00**	**1.00**	0.06	0.74	0.14	0.06	0.06	0.98	0.91
Emilio	**0.87**	0.77	0.81	0.76	0.08	0.11	0.27	0.65	0.08	0.79	0.68
Tennis Match	**0.96**	**0.96**	**0.96**	0.95	0.23	0.55	0.01	(X)	0.57	0.64	0.87
Animal	**1.00**	0.92	0.90	0.07	0.38	0.39	**1.00**	0.06	0.04	0.75	0.82
Football	**0.94**	0.31	0.67	0.18	0.06	0.41	0.07	0.45	0.14	0.87	0.20
PETS2009	0.97	0.24	0.97	0.21	0.19	**0.99**	0.92	**0.99**	0.22	0.97	0.39
Bouncing2	**1.00**	**1.00**	**1.00**	0.76	0.21	0.46	0.01	0.01	0.01	**1.00**	0.67
Rolling Ball	**0.83**	0.80	0.80	0.81	0.17	0.72	0.11	0.50	0.16	**0.83**	0.49
Doll	**0.71**	0.65	0.65	0.65	0.17	0.68	0.05	(X)	0.05	0.58	0.7
David2	**1.00**	0.80	0.74	0.73	0.36	0.33	0.24	0.87	0.74	0.93	0.19
Boy	**0.99**	0.94	0.90	0.51	0.31	0.51	0.19	0.98	**0.99**	0.97	0.94
Jogging	0.96	0.25	0.25	0.67	0.71	0.75	0.25	0.25	0.25	**0.98**	0.90
Jumping	0.61	0.06	0.18	0.06	0.07	0.80	0.08	0.11	0.05	0.06	**0.93**
Girl	0.78	0.15	0.15	0.51	0.40	0.75	0.13	0.64	**0.96**	0.50	0.76
Bird2	0.72	0.51	0.5	0.49	0.38	0.32	0.04	0.13	**0.98**	0.25	0.49
Cup	0.98	**1.00**	0.99	**1.00**	0.59	0.84	0.08	0.96	0.13	0.96	0.41
Hand	**0.92**	0.41	0.80	0.53	0.04	0.14	0.01	0.09	0.12	0.27	0.51
Tiger1	0.53	0.46	0.48	0.48	0.41	0.34	0.01	0.18	0.47	0.36	**0.74**
Freeman1	0.2	0.11	0.1	0.08	0.16	0.2	0.01	**0.22**	0.15	0.07	0.19

Table 5.3: The overlap ratio between the predicted bounding box and the ground truth bounding box for each testing video sequence.



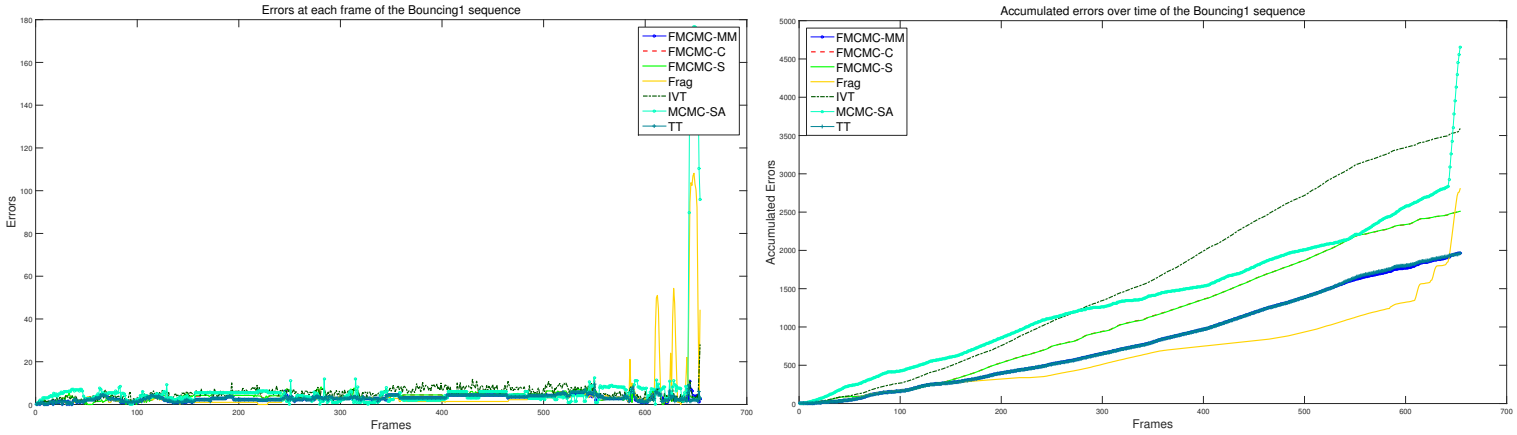
(a) (b)

Figure 5.2: Errors at each frame and accumulated errors over time of trackers for the Data11 sequence.*(Note: SB, IVT, OAB, TT were removed because they drifted off the target).*



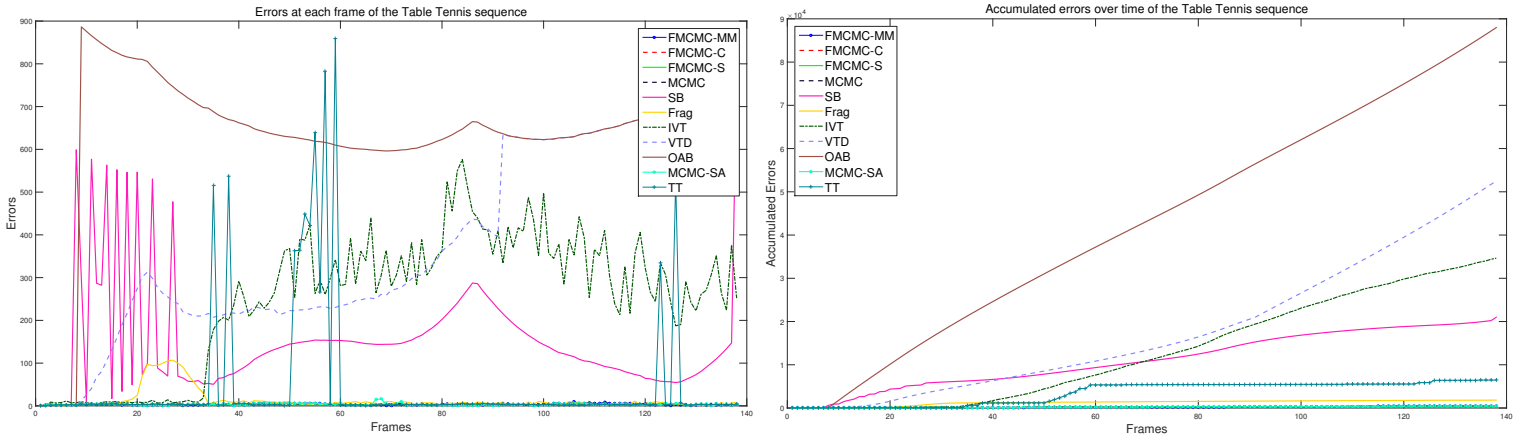
(a) (b)

Figure 5.3: Errors at each frame and accumulated errors over time of trackers for the Data12 sequence.



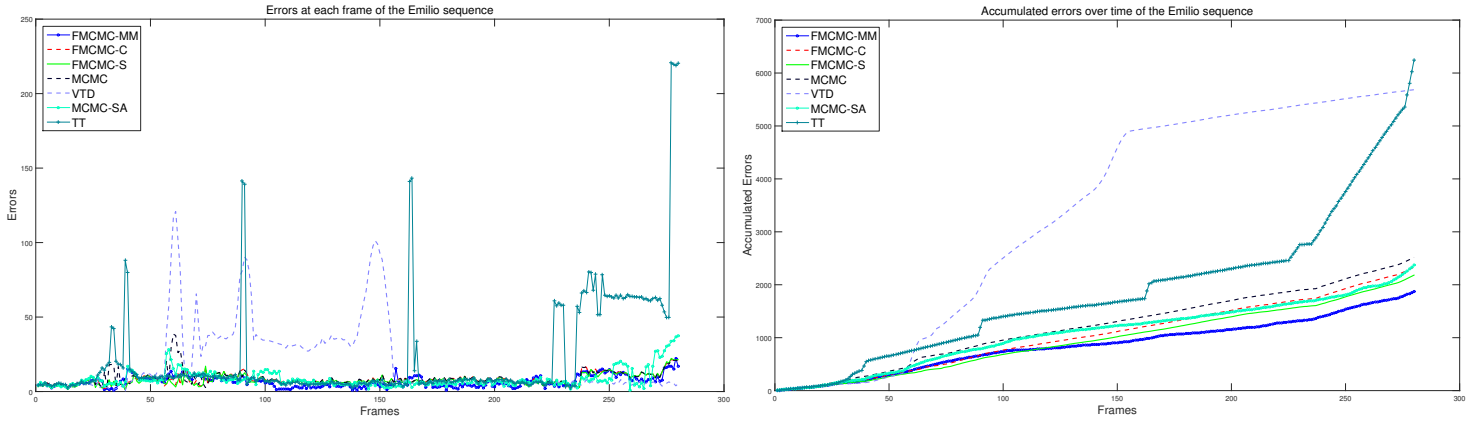
(a) (b)

Figure 5.4: Errors at each frame and accumulated errors over time of trackers for the Bouncing1 sequence.



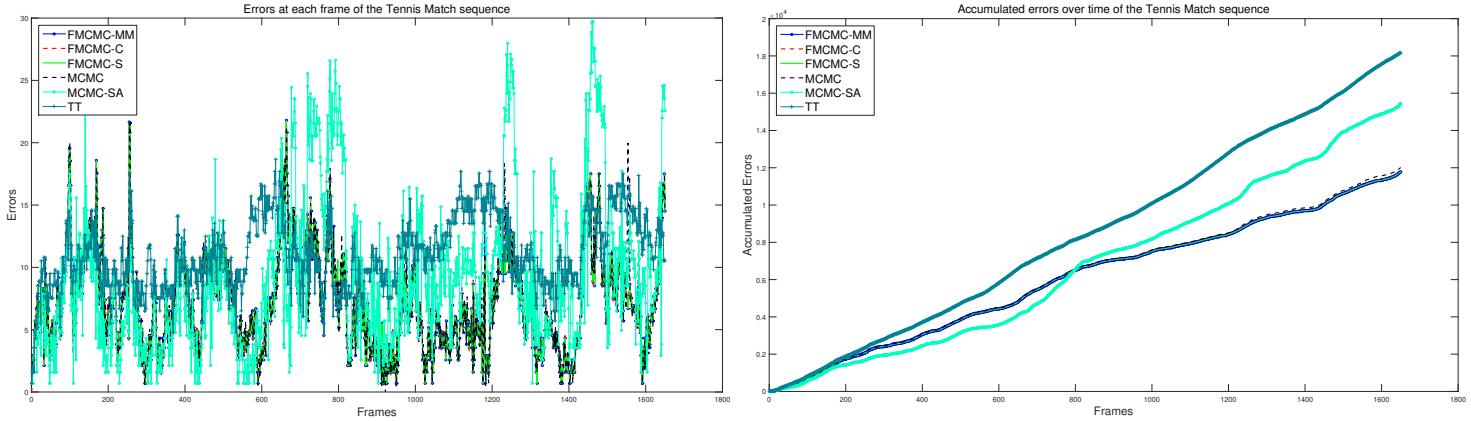
(a) (b)

Figure 5.5: Errors at each frame and accumulated errors over time of trackers for the Table Tennis sequence.



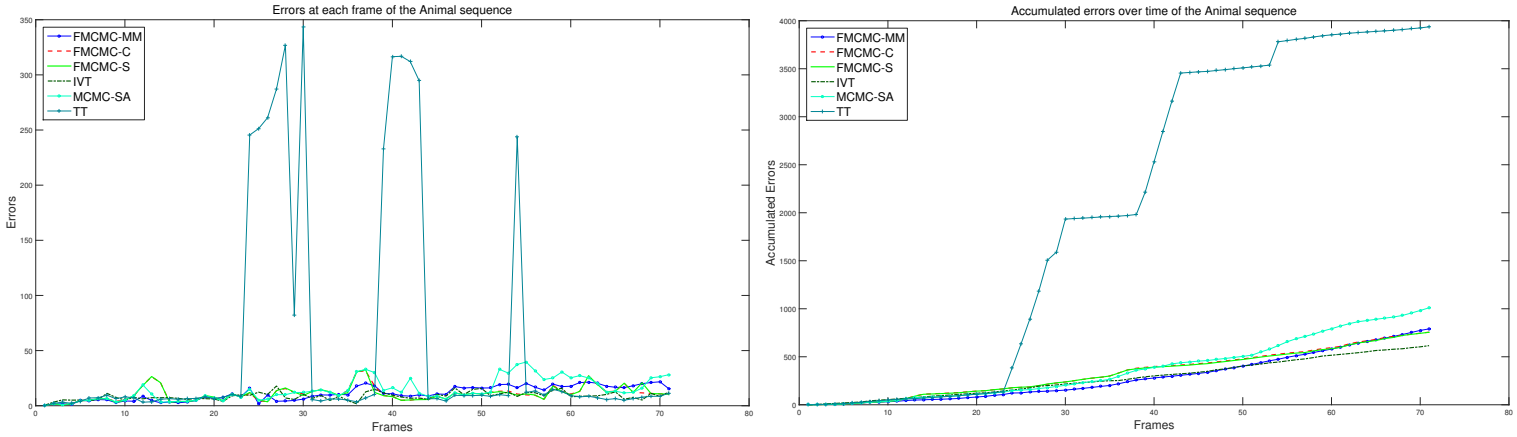
(a) (b)

Figure 5.6: Errors at each frame and accumulated errors over time of trackers for the Emilio sequence.*(Note: SB, Frag, OAB, IVT were removed because they drifted off the target).*



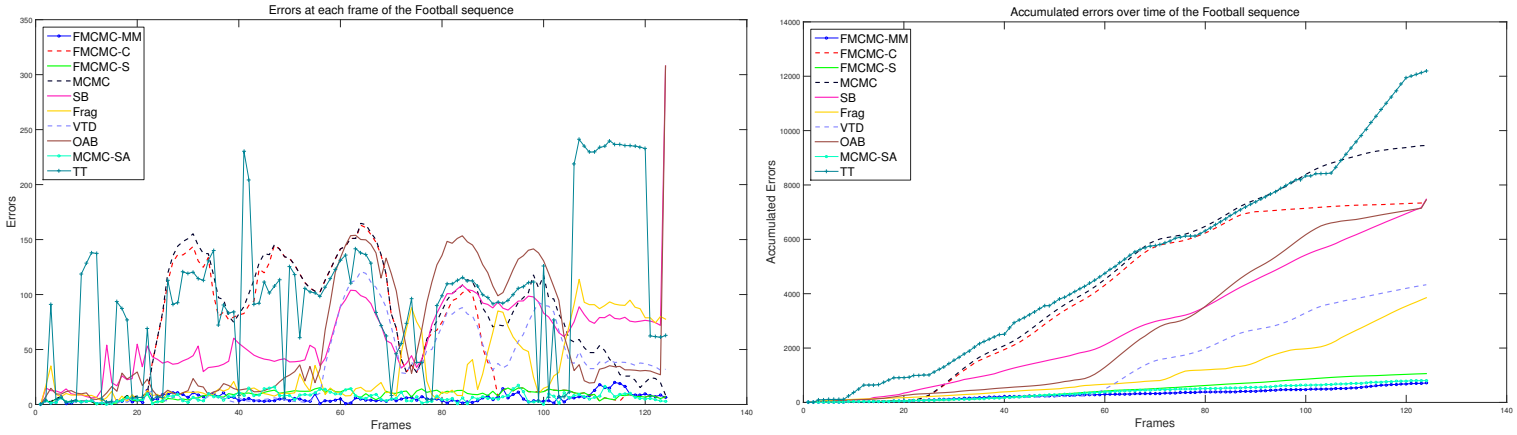
(a) (b)

Figure 5.7: Errors at each frame and accumulated errors over time of trackers for the Tennis Match sequence.*(Note: VTD, IVT, OAB, SB, FragTrack were removed because they drifted off the target).*



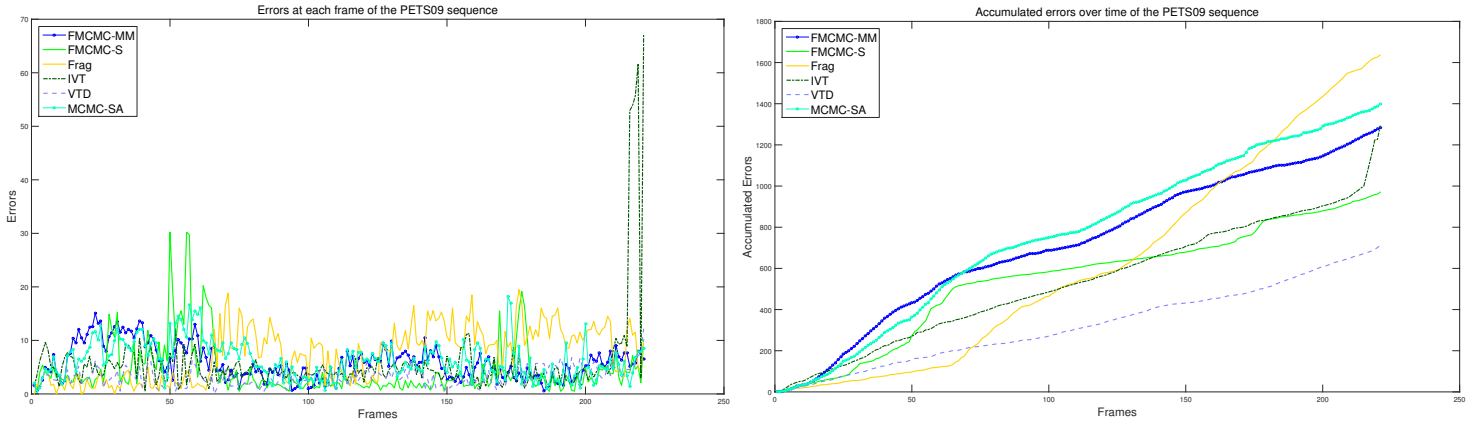
(a) (b)

Figure 5.8: Errors at each frame and accumulated errors over time of trackers for the Animal sequence.*(Note: MCMC, VTD, FragTrack, SB, OAB were removed because they drifted off the target).*



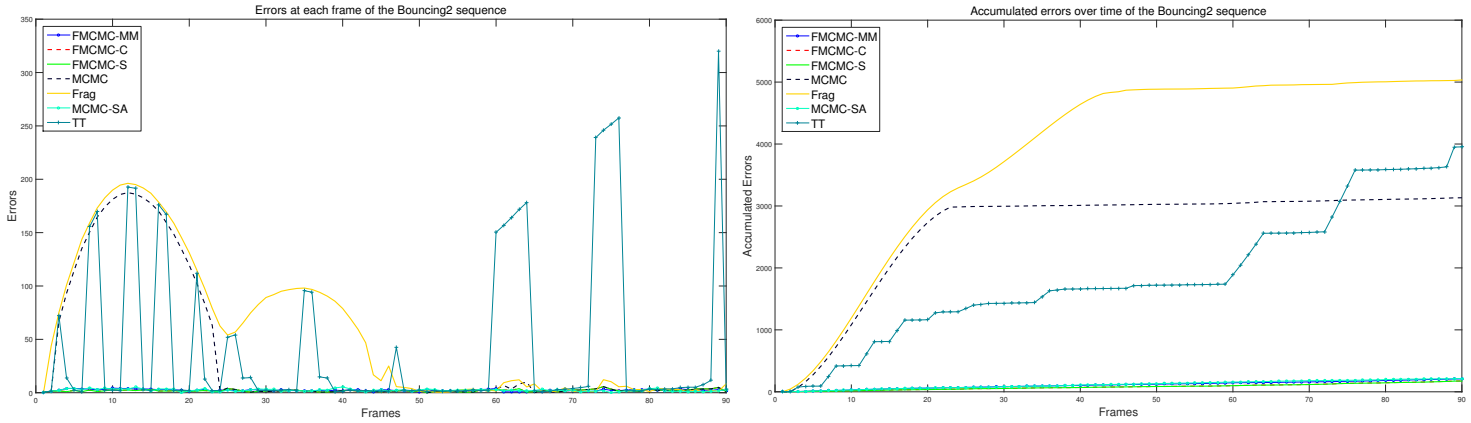
(a) (b)

Figure 5.9: Errors at each frame and accumulated errors over time of trackers for the Football sequence.*(Note: IVT was removed because they drifted off the target).*



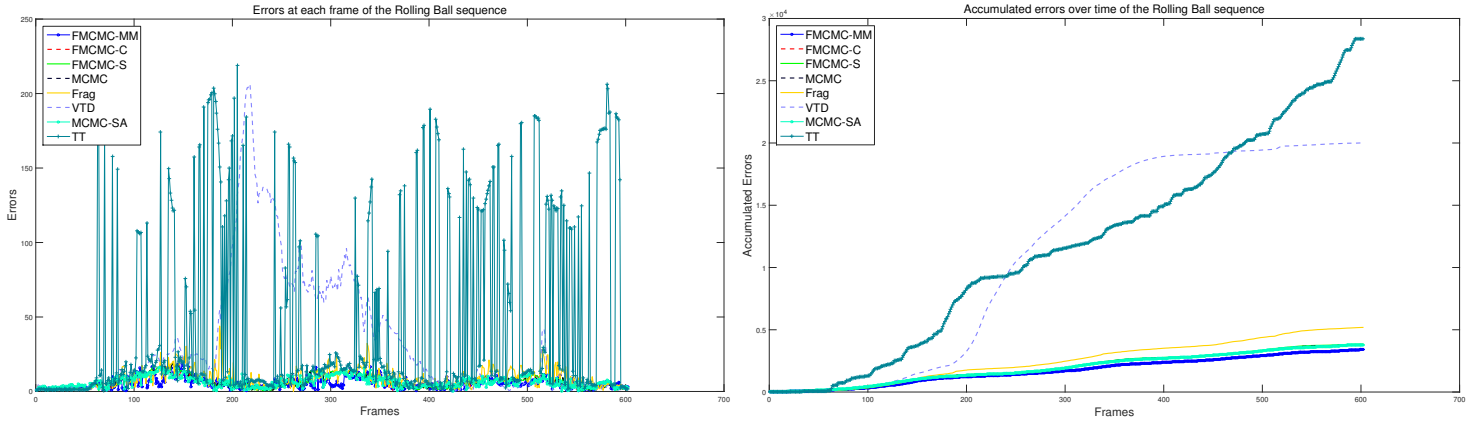
(a) (b)

Figure 5.10: Errors at each frame and accumulated errors over time of trackers for the PETS09 sequence.*(Note: MCMC, FMCMC-C, SB, OAB, TT were removed because they drifted off the target).*



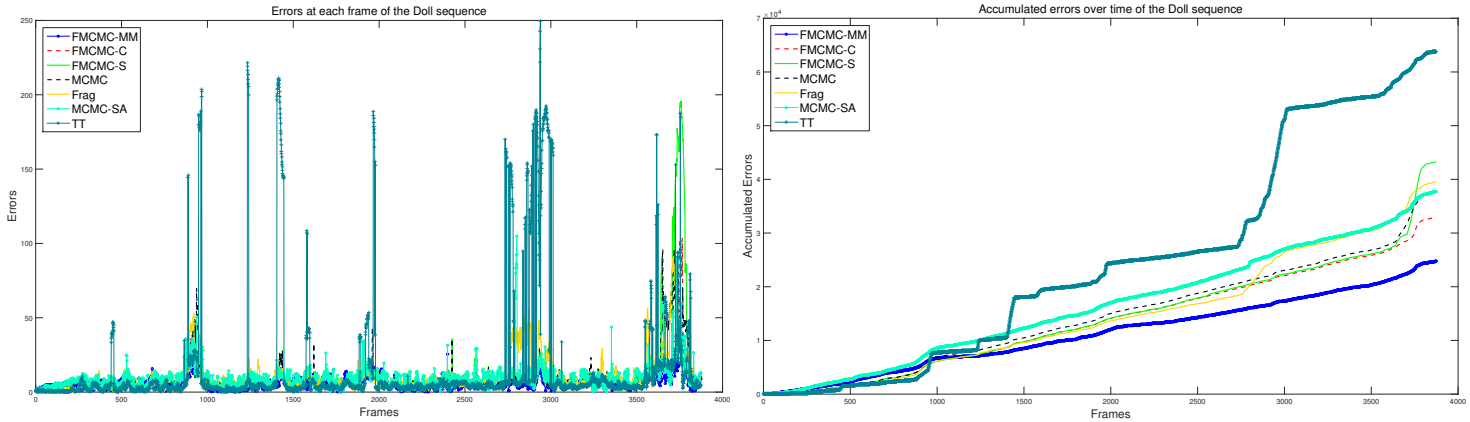
(a) (b)

Figure 5.11: Errors at each frame and accumulated errors over time of trackers for the Bouncing2 sequence.*(Note: SB, IVT, VTD, OAB were removed because they drifted off the target).*



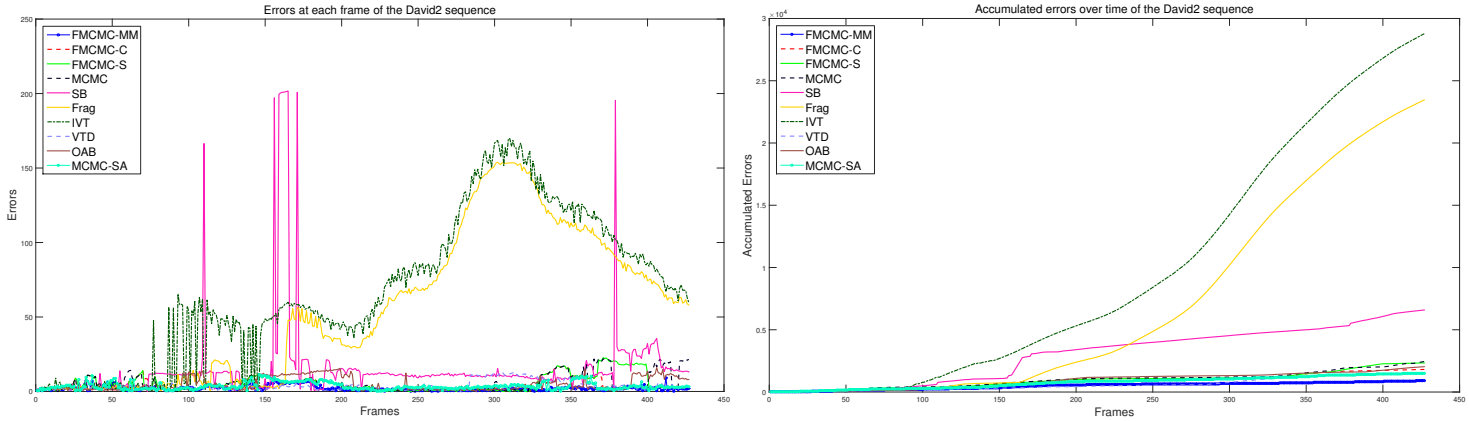
(a) (b)

Figure 5.12: Errors at each frame and accumulated errors over time of trackers for the Rolling Ball sequence.*(Note: SB, IVT, OAB were removed because they drifted off the target).*



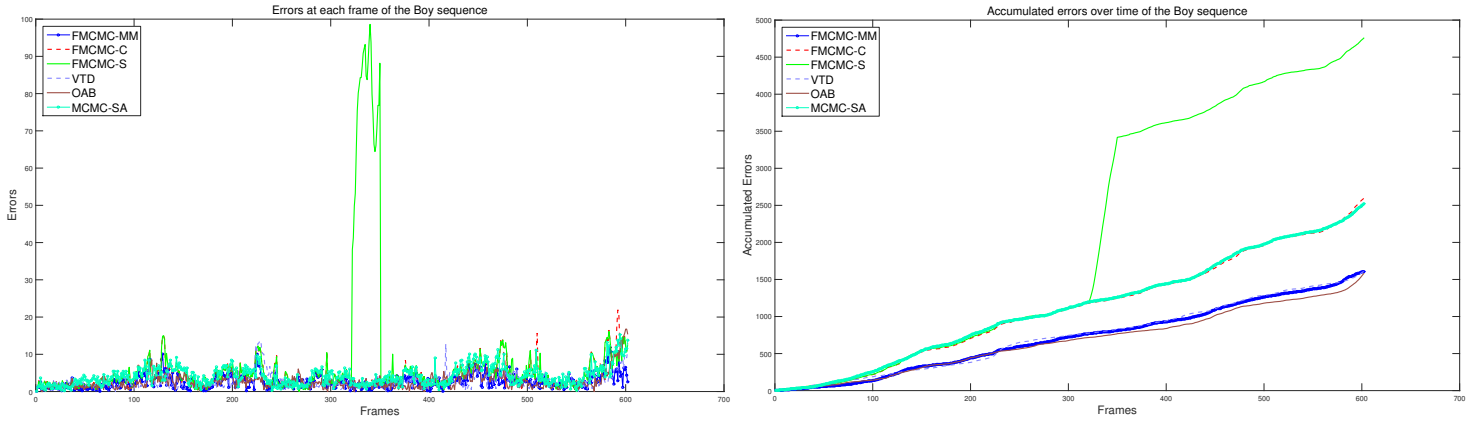
(a) (b)

Figure 5.13: Errors at each frame and accumulated errors over time of trackers for the Doll sequence.*(Note: SB,IVT,VTD,OAB were removed because they drifted off the target).*



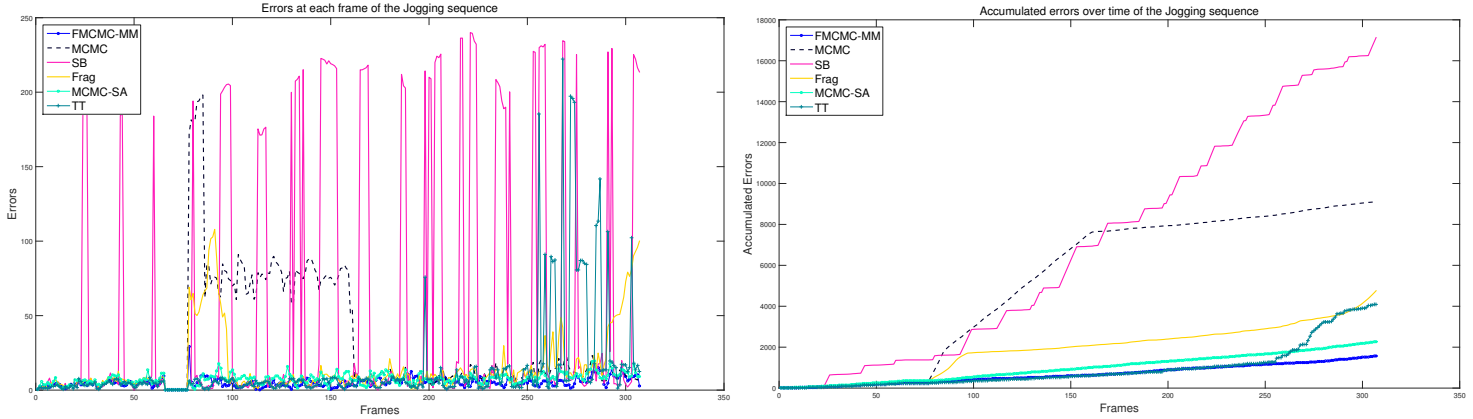
(a) (b)

Figure 5.14: Errors at each frame and accumulated errors over time of trackers for the David2 sequence.*(Note: TT was removed because they drifted off the target).*



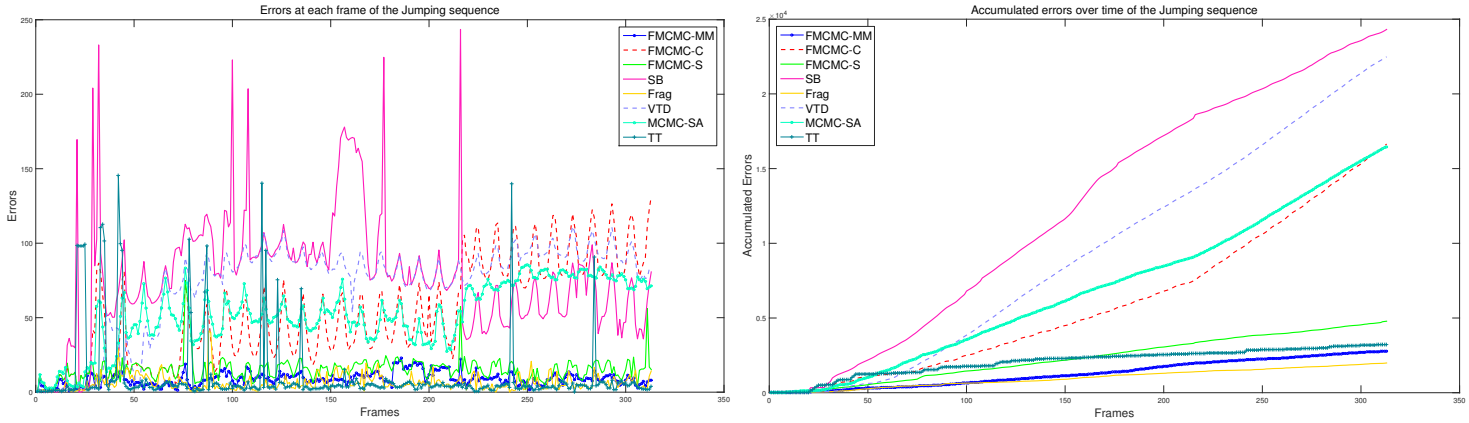
(a) (b)

Figure 5.15: Errors at each frame and accumulated errors over time of trackers for the Boy sequence.*(Note: MCMC, SB, IVT, FragTrack, TT were removed because they drifted off the target).*



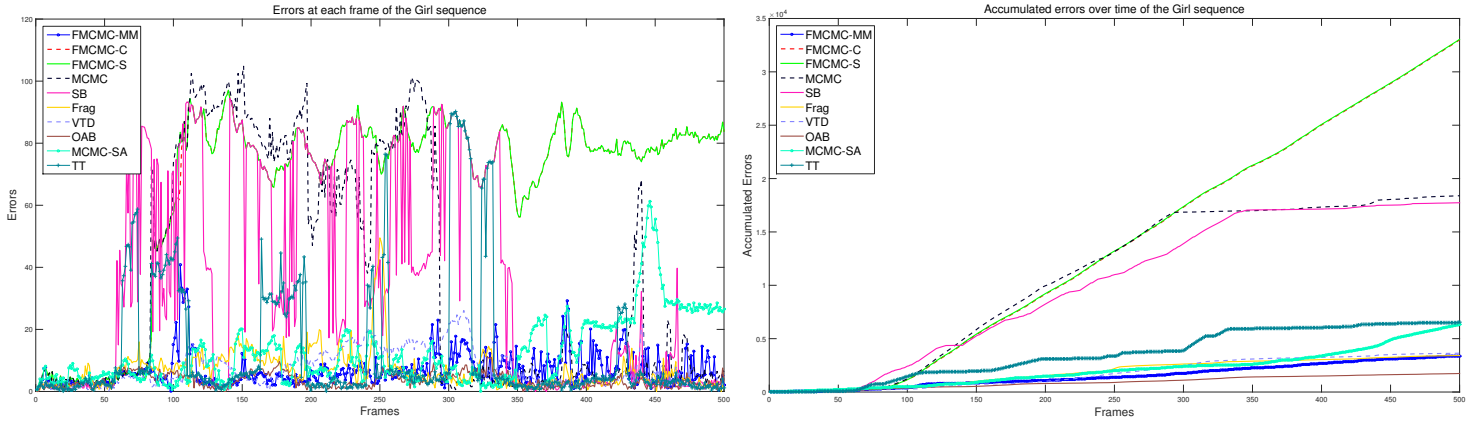
(a) (b)

Figure 5.16: Errors at each frame and accumulated errors over time of trackers for the Jogging sequence.*(Note: FMCMC-C, FMCMC-S, IVT, VTD, OAB were removed because they drifted off the target).*



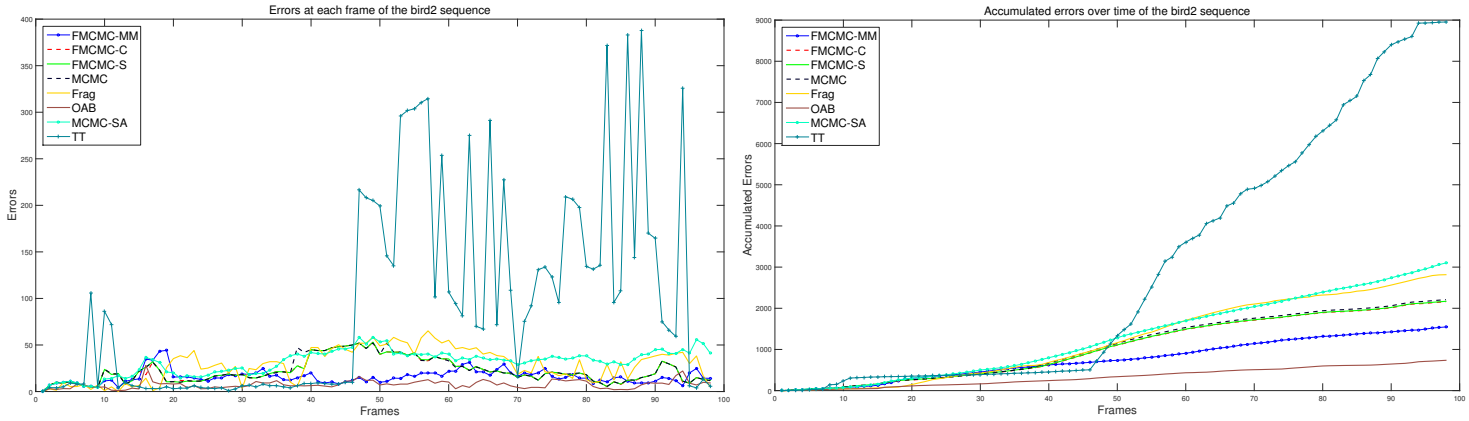
(a) (b)

Figure 5.17: Errors at each frame and accumulated errors over time of trackers for the Jumping sequence.*(Note: MCMC, IVT, OAB were removed because they drifted off the target).*



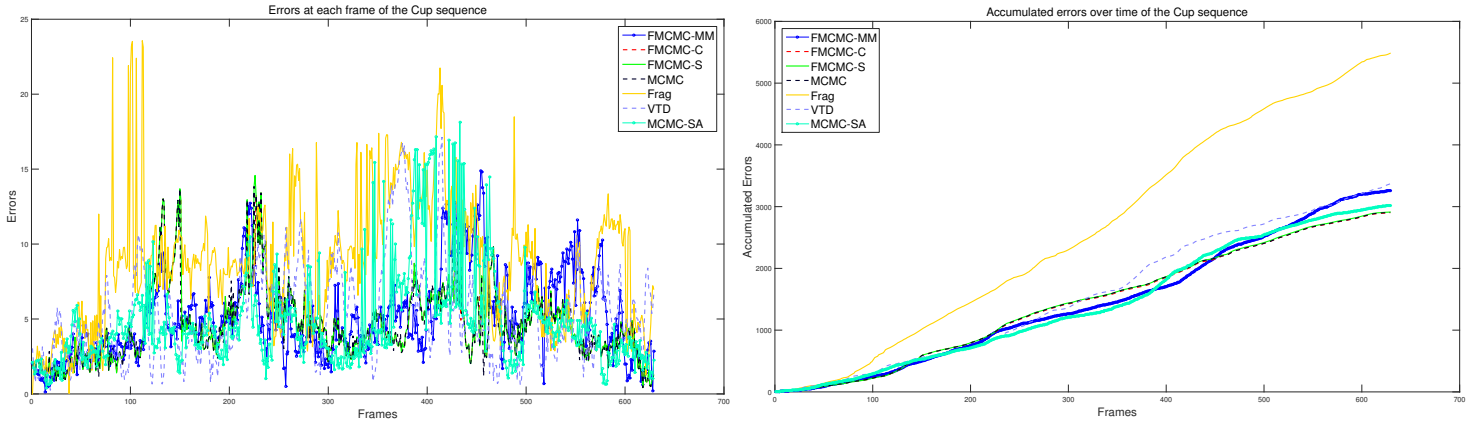
(a) (b)

Figure 5.18: Errors at each frame and accumulated errors over time of trackers for the Girl sequence.*(Note: IVT was removed because they drifted off the target).*



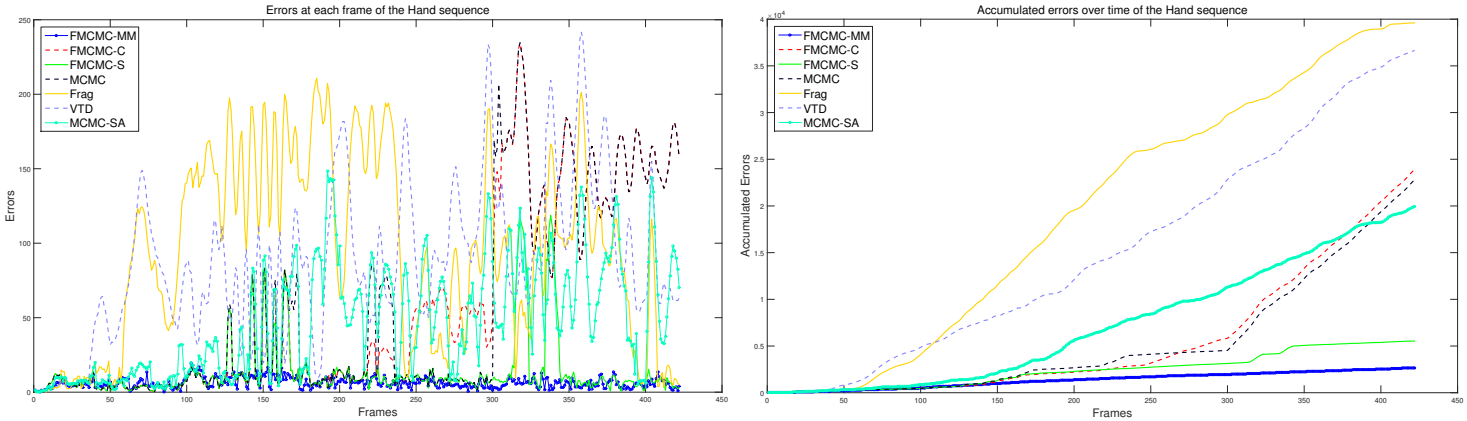
(a) (b)

Figure 5.19: Errors at each frame and accumulated errors over time of trackers for the Bird2 sequence.*(Note: SB, IVT, VTD were removed because they drifted off the target).*



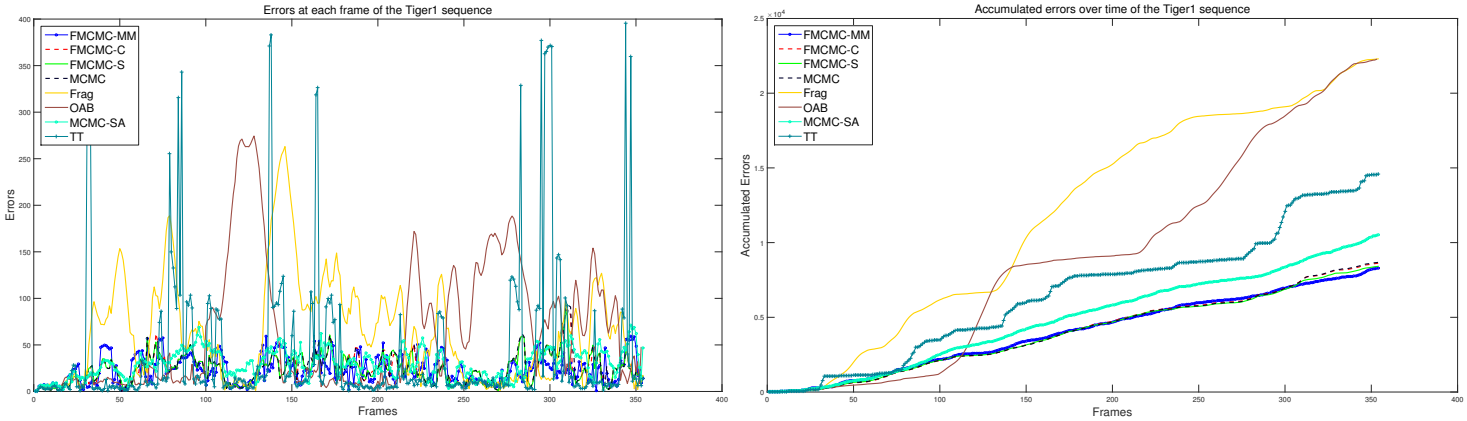
(a) (b)

Figure 5.20: Errors at each frame and accumulated errors over time of trackers for the Cup sequence.*(Note: IVT, OAB, SB, TT were removed because they drifted off the target).*



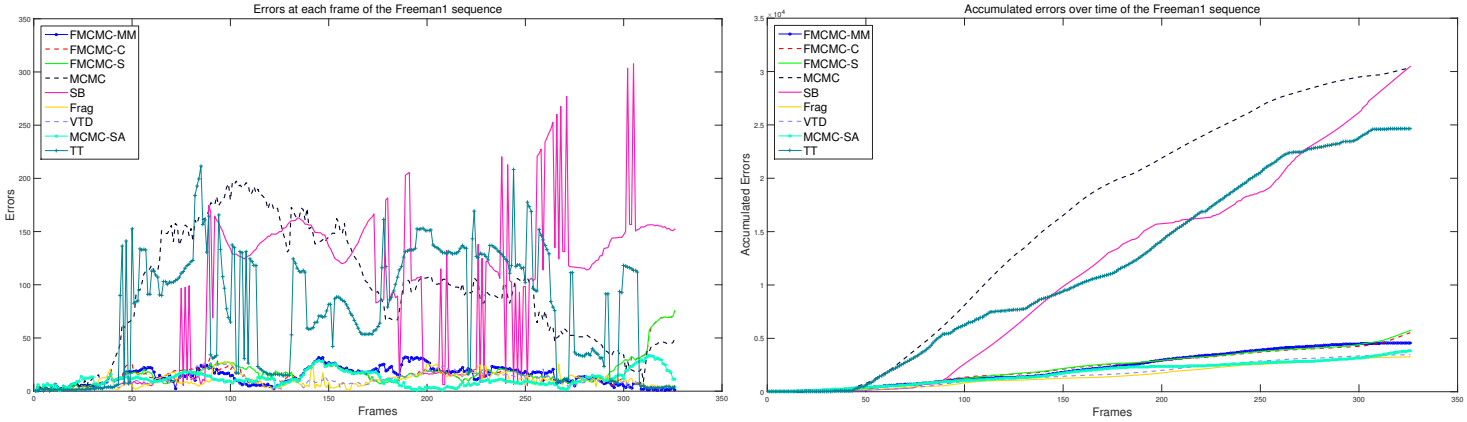
(a) (b)

Figure 5.21: Errors at each frame and accumulated errors over time of trackers for the Hand sequence.*(Note: SB, OAB, IVT, TT were removed because they drifted off the target).*



(a) (b)

Figure 5.22: Errors at each frame and accumulated errors over time of trackers for the Tiger1 sequence.*(Note: SB, IVT, VTD were removed because they drifted off the target).*



(a) (b)

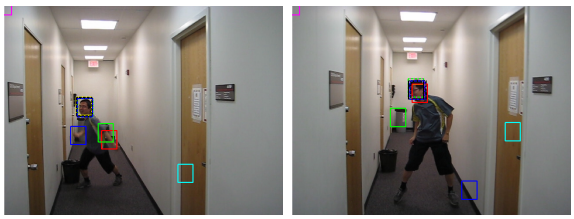
Figure 5.23: Errors at each frame and accumulated errors over time of trackers for the Freeman1 sequence.*(Note: IVT, OAB were removed because they drifted off the target).*

5.4 Discussion

5.4.1 Handling Changes in Appearance

FMCMC-MM's behaviour in handling appearance change is similar to MCMC-SA's, as their appearance models are alike. FMCMC-MM, however, located the target more accurately. This aspect is discussed more detail in the next section.

Compared to OAB, FMCMC-MM could not quickly adapt to appearance changes in the Boy (Figure Figure D.19) (e.g. Frame #475 of Figure 5.24), Girl (Figure D.22) (e.g. Frame #386 of Figure 5.25), and Bird2 (Figure D.23) (e.g. Frame #18 of Figure 5.26)sequences because the appearance model in OAB is updated blindly. Whilst FMCMC-MM only updates the appearance model in a supervised manner, i.e. FMCMC-MM updates or adds a appearance model when this model is different from appearance models stored in the appearance pool. FMCMC-MM, however, worked better than OAB in the rest of the video sequences and always followed the target as demonstrated in Table 5.2. OAB needs precise locations to extract good features to represent the target.

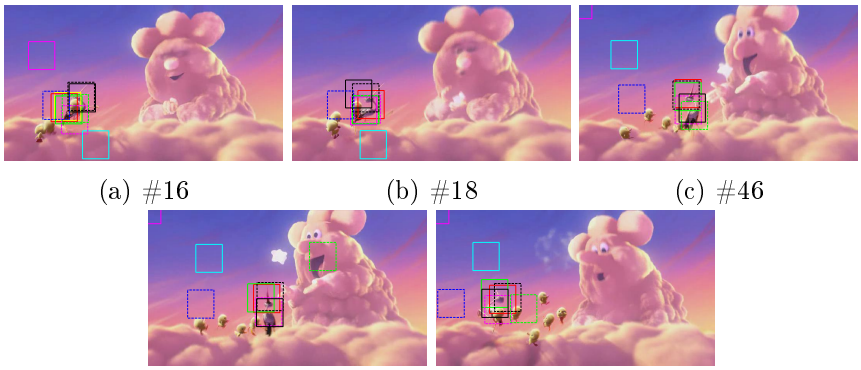


(a) #339 (b) #475

Figure 5.24: Tracking results in selected frames of the Boy sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

(a) #386 (b) #404 (c) #490

Figure 5.25: Tracking results in selected frames of the Girl sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

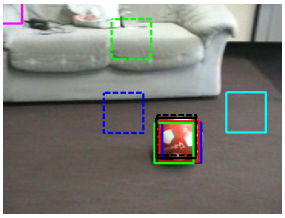


(a) #16 (b) #18 (c) #46

(d) #48 (e) #91

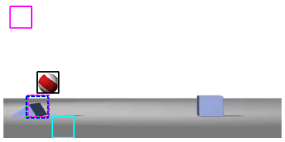
Figure 5.26: Tracking results in selected frames of the Bird2 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

VTD worked better than FMCMC-MM in four video sequences (Data11 (Figure D.1), Data12 (Figure D.2), PETS2009 (Figure D.12) and Freeman1 (Figure D.29)). In those videos, the target slightly changed its appearance. Its multiple SPCA based appearance models allows VTD locate the target slightly better than FMCMC-MM. VTD, however, did not work better than FMCMC-MM in the other sequences because when VTD drifts off the target, as it did in several frames, it selects features which do not represent the target. VTD lost the target in Frame #193 (Figure 5.27) of the Rolling Ball sequence (Figure D.14); Frame #2 of the Bouncing2 (Figure D.13) and Frame #16 (Figure 5.26) of the Bird2 sequence (Figure (Figure 5.28) D.23).



(a) #194

Figure 5.27: Tracking results in the selected frame of the Rolling Ball sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



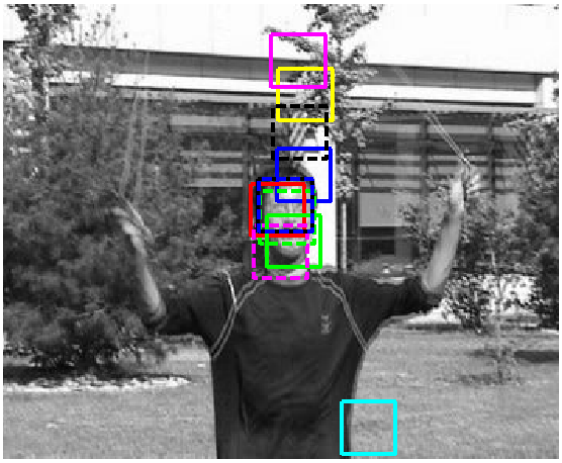
(a) #2

Figure 5.28: Tracking results in the selected frame of the Bouncing2 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

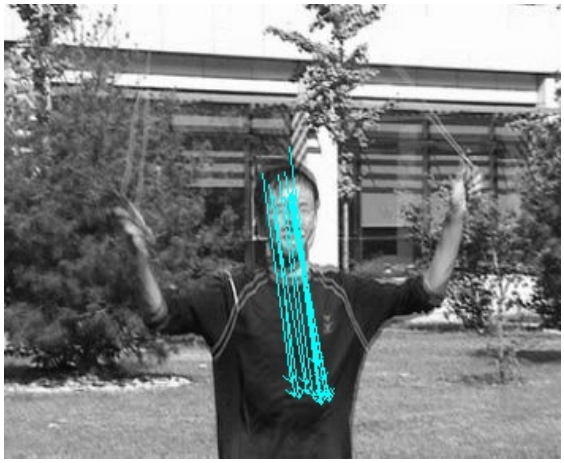
Updating the appearance model when the target does not change or only slightly changes its appearance can reduce tracking performance, especially in cases of incorrect target location and when target scale decreases. Such that MCMC-SA and FMCMC-MM could not track better than FMCMC-C, FMCMC-S in the Data11 (Figure D.1), Data12 (Figure D.2) and Cup (Figure D.24) sequences because the the target's size slightly increases/decreases when the target comes close or moves away from the camera. Knowing precisely when to update is still an open issue, though using thresholds to decide updating appearance model is one solution. It is better if these thresholds are adaptive during tracking. Thresholds need in FMCMC-MM because the appearance model is updated in supervised manner. This is the reason why it does not quick update to the changes of target appearance as OAB does. However, it allows the tracker not update the appearance model when there is uncertainty in the tracking result.

5.4.2 Target Location Improvement

FMCMC-MM improved tracking performance significantly when compared to MCMC-SA in the Hand (Figures D.25, D.26, D.27), Emilio (Figures D.6, D.7), Doll (Figures D.15, D.16), David2 (Figure D.17), Girl (Figure D.22), Bird2 (Figure D.23), and Jumping (Figure D.21) sequences. MCMC-SA predicted the target location where it has the highest score returned by NCC when comparing each template to regions in the image. This can work well, e.g. in Bouncing2 (Figure D.13), Emilio and Football (Figures D.10, D.11) sequences. It, however, failed to track the target in Jumping (e.g. Frame #31) (Figure D.21) because the region with the highest score does not belong to the target. This can happen because a template model is a generative model and it easily gets distractors. Feature detection helps FMCMC-SA located the target correctly. Figure 5.29 shows features detected and matched at Frame #31 of the Jumping sequence. The cyan arrows show the target movement between two consecutive frames. These motions help FMCMC-SA find the correct location (e.g. the search moves down the image to find the target instead of going up like MCMC-SA).



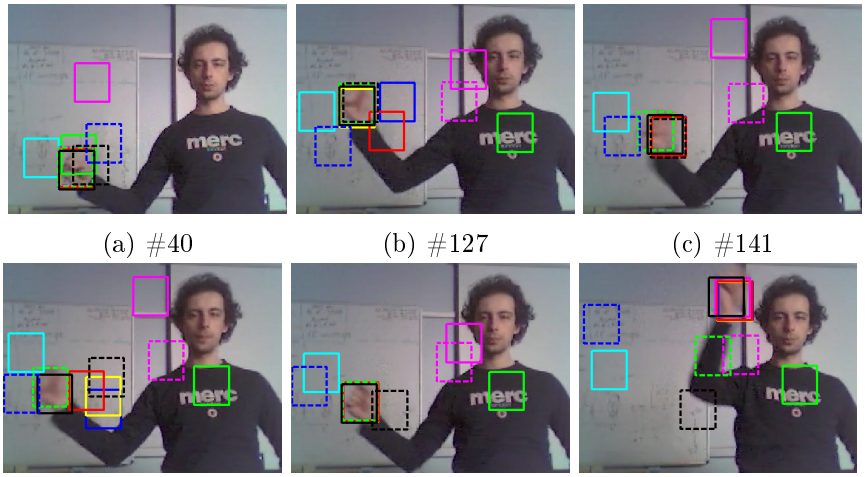
(a) Tracking results (MCMC-SA (dashed) black, FMCMC-MM (black)



(b) Feature Motion at #31

Figure 5.29: (Enlarged) Feature movement at the Frame #31 the Jumping sequence.

In the Hand (Figure D.25 and Figure D.26) sequence (e.g. Frames #181, #196 of Figure 5.30), when the target (i.e. the hand) moved from the right toward the left at Frame #135, MCMC-SA lost the target. FMCMC-MM, however, tracked well. Figure 5.31 shows tracking results and feature motions at Frame #135. In Frame #141, MCMC-SA could track the target again when the target returned the place where MCMC-SA lost the target because the tracker still maintained the correct target appearance model in the appearance pool.

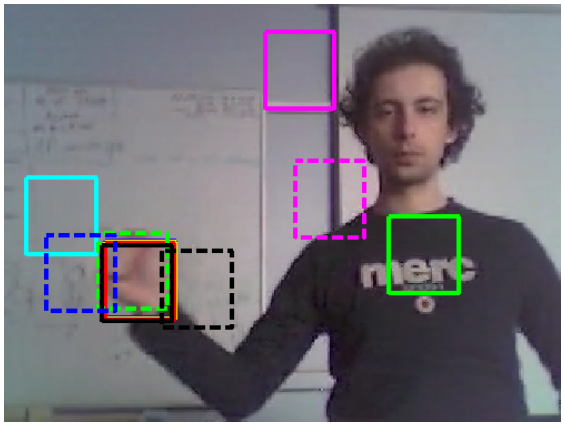


(a) #40 (b) #127 (c) #141

(d) #157 (e) #181 (f) #196

Figure 5.30: Tracking results in selected frames of the Hand sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

In the Tiger1 (Figure D.28) sequence, MCMC-SA incorrectly estimated the target location at Frames #123, #135, #166, #329 (Figure 5.32) when the target moved fast from the right to the left. Whereas FMCMC-MM captured the target correctly.



(a) Tracking results (MCMC-SA (dashed) black, FMCMC-MM (black)

(b) Frame #135

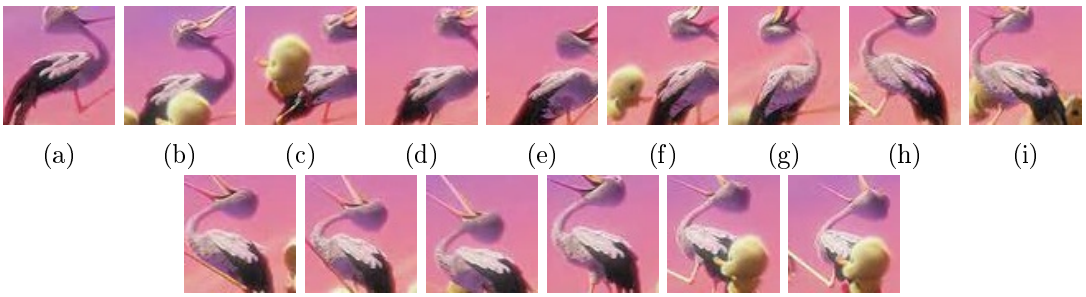
Figure 5.31: Feature motion at the Frame #135 of the Hand sequence.

(a) #66 (b) #105 (c) #123

(d) #135 (e) #166 (f) #329

Figure 5.32: Tracking results in selected frames of the Tiger1 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

In the Bird2 sequence (e.g. Frames #46, #48, #91 (Figure 5.26)) (Figure D.23), without the support of features, MCMC-SA located the target incorrectly. Therefore, the templates extracted by MCMC-SA were not useful. MCMC-SA tracked the bird's head more often after Frame #46 than FMCMC-MM did. Figure 5.33 shows templates gathered while tracking the Bird2 sequence by FMCMC-MM. These templates represent the target precisely compared to the templates detected by MCMC-SA (Figure 3.22o of Section 3.3).



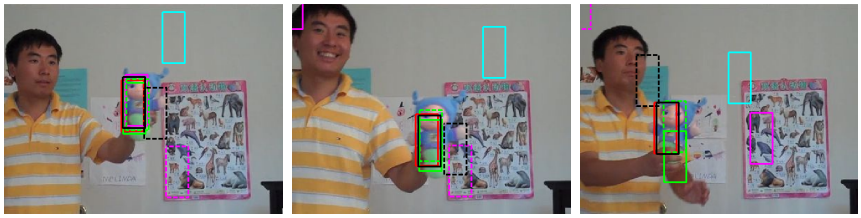
(a) (b) (c) (d) (e) (f) (g) (h) (i)

(j) (k) (l) (m) (n) (o)

Figure 5.33: (Enlarged) Bird templates detected during tracking of FMCMC-MM.

Frame #1689, #1915, #2794 (Figure 5.34) of the Doll sequence (Figures D.15 and D.16) and Frame #404,#490 (Figure 5.25) of the Girl sequence (Figure D.22) also show

FMCMC-MM working better than MCMC-SA.

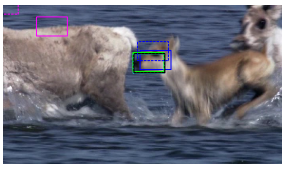


(a) #1689 (b) #1915 (c) #2794

Figure 5.34: Tracking results in selected frames of the Doll sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

5.4.3 Motion Variation Handling

FMCMC-MM with motion sampling approach performed better than MCMC-SA did on the Bouncing1 (Figure D.3 and D.4), Bouncing2 (Figure D.13), Emilio (Figures D.6 and D.7), Animal (Figure D.9) and Hand sequences (Figure D.25). Moreover, FMCMC-MM tracked the target more accurately than FMCMC-S when the target changes its appearance, e.g. in Frame #127, #157 (Figure 5.30) of the Hand sequence (Figures D.25 and D.26), Frame #339 (Figure 5.24) of the Boy sequence (Figure D.19).

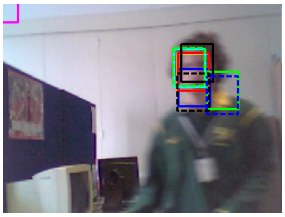


(a) #5

Figure 5.35: Tracking results in the selected frame of the Animal sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

VTD had multiple basic motion models. It, however, could only handle smooth movement. VTD could track the target in the Data11 (Figure D.1), Data12 (Figure D.2) and PETS2009 (Figure D.12) sequences. When the target suddenly moved in a different direction or changed its velocity such as at Frame #40 of the Hand sequence (Figure D.25), Frame #5 (Figure 5.35) of the Animal (Figure D.9), Frame #57 (Figure 5.36) of the Emilio sequence (Figure D.6), it lost the target. FMCMC-MM with the motion

direction sampling component, however, predicted target locations correctly.

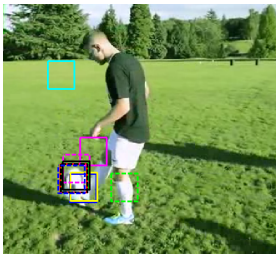


(a) #57

Figure 5.36: Tracking results in the selected frame of the Emilio sequence(Part 1). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

5.4.4 Distractor Handling

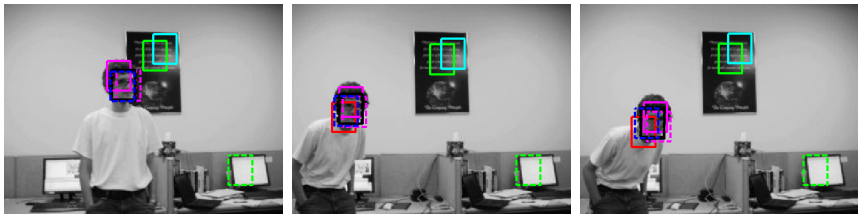
The features used in FMCMC-S help avoid distractors by providing a reasonable search area as an initial estimate of target location. FMCMC-MM is built on top of FMCMC-S, so it can avoid distractors. MCMC-SA could avoid distractors because of the use of template to enhance the target location as shown in Frame #22 (Figure 5.37) of the Football sequence (Figure D.10).



(a) #22

Figure 5.37: Tracking results in the selected frame of the Football sequence(Part 1). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

FMCMC-MM and MCMC-SA could avoid distractors at Frames #224, #334, #340 (Figure 5.38) of the David2 sequence D.17 whilst FragTrack, IVT and TT locked onto the board or the monitor.



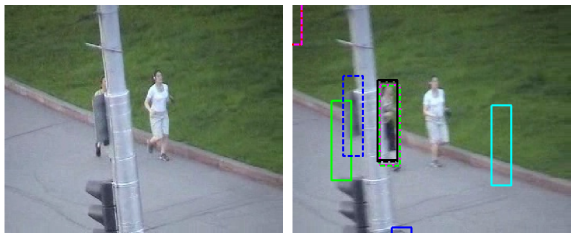
(a) #224 (b) #334 (c) #340

Figure 5.38: Tracking results in selected frames of the David2 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

5.4.5 Occlusion Handling

FMCMC-MM and MCMC-SA have been designed to detect occlusion and re-detect the target following occlusion. They rely on scores returned by simple similarity functions (NCC and Bhatacharya distance). A more sophisticated occlusion could, however, be embedded into these methods to make them more flexible.

FMCMC-C and FMCMC-S did not work well after occlusion at Frame #69 (Figure 5.39) of the Jogging sequence (Figure D.20) because feature based motions after the occlusion did not reflect motions of the true target. VTD could also not track the target after the occlusion because features selected from past frames did not represent the target appearance, and the target moved beyond the search area covered by multiple basic motion models of VTD. Note that in VTD, there is no mechanism to estimate the target's velocity. Also, the target's velocity could help if it does not change dramatically before and after occlusion occurs.



(a) #69 (b) #79

Figure 5.39: Tracking results in selected frames of the Jogging sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

Similarly, VTD and OAB tracked the background object (leaf) after the target was occluded at Frames #66, #105 (Figure 5.32) of the Tiger1 sequence (Figure D.28). FMCMC-MM and MCMC-SA could track the target after occlusion because of using templates learnt.

5.4.6 Scale Change Handling

A scale sampling approach is used in many approaches employing a rectangle to define the target boundary (e.g. IVT, VTD). FMCMC-MM and MCMC-SA could employ the scale sampling approach to handle scale change. Size among templates should be, however, equivalent when using NCC to compare them. This limitation results it difficult for FMCMC-MM and MCMC-SA to apply scale handling. A flexible similarity function should be considered in this case.

5.5 Summary

We have proposed a single tracking algorithm (i.e. without fusing multiple trackers) applicable to both rigid and deformable targets. The appearance model combines two popular generative models, utilising their complementary advantages to improve tracking performance. The tracker uses a pool of template-histogram pairs to provide the best fit appearance model, switching among them using a sampling mechanism. Appearance changes are automatically detected and new, corresponding templates are extracted. These templates are checked in a supervised manner for similarity to other templates maintained in the pool before adding them to it. The MCMC-based search uses the distribution of motion directions of local image features from the feature pool to enhance target prediction. These local motion directions are extracted directly from two consecutive frames. The algorithm can also handle variation in the motion of a target without using any prior knowledge of movement.

Experiments showed the FMCMC-MM tracker to have performance advantages over other trackers and significantly improve tracking accuracy compared to MCMC-SA, FMCMC-C and FMCMC-S. It could capture the target appearance changes in targets displaying complex movements. MCMC-SA could only handle appearance changes when the target had smooth movement. The explicit occlusion detection step of FMCMC-MM reduced the risk of updating the pool with features not belonging to the target. FMCMC-MM could avoid distractors by searching for the target along motion directions close to the true target motion. A robust similarity function could be employed to replace NCC

when dealing with scale changes.

Chapter 6

Conclusion and Future Work

6.1 Contributions

The work reported in this thesis has made the following contributions:

- A new approach has been proposed which handles appearance changes by maintaining and selecting from a pool of template-histogram pairs. Templates provide solid landmarks, supporting accurate prediction of target location and allowing the tracker to decide when to learn a new appearance model, alleviating the drifting problem. Histograms are used to handle appearance changes and model target appearance during search. This approach implicitly updates the appearance model by maintaining multiple appearance models and adding a new appearance model when necessary. Each appearance model presents a change of the target appearance. During tracking, the tracker automatically selects and switches among appearance models maintained in the appearance pool to find a suitable appearance model to describe the target.

- A novel bottom-up approach to motion modelling and location prediction in which likely target movement is captured implicitly by a set of local feature-based motion vectors. Features are detected and matched between consecutive frames to form a motion direction distribution. This method can handle target motion variations and unexpected movements without embedding target motions in advance, which are difficult to model precisely. The motion model then supports the target search via multiple linear searches by sampling a motion direction from the motion direction distribution.

- These components have been combined to produce a tracking algorithm integrating

the appearance model sampling approach and multiple linear searches. This tracker can not only handle target motion variations but also deal with appearance changes. The unified tracker has been evaluated on a variety of challenging image sequences.

6.2 Research Outcomes

Tracking targets in real world situations is a challenging problem due to dynamic and complex backgrounds, target appearance changes and unpredictable motion of the objects of interest. Many approaches (as discussed in Chapter 2) have addressed the problem of varying appearance by building a rich appearance model using one or fusing multiple features. These approaches (e.g. online learning methods) can quickly adapt to appearance changes. They, however, face a key issue: model drift. Regardless of the appearance modelling approach (e.g. use of generative or discriminative models) adopted, these methods rely on an anchor or a prior (e.g. a simple linear update of the reference model, a fixed adaptation speed, semi online learning, co-training). These, however, cannot adapt quickly to appearance changes.

In Chapter 3, a new approach has been proposed to handle appearance changes by employing multiple appearance models stored in an appearance pool. This technique removes the tracker's reliance on a single appearance model carefully designed and selected at implementation time. As reported in Chapter 3, this method can handle appearance changes well if the target moves smoothly, reducing the likelihood of model drift. When drift does occur, the mechanism increases the likelihood that the tracker will re-locate the target; the appearance pool provides a large set of learnt appearance models from which a more appropriate one is selected.

To reduce the search space and enhance target predictions (i.e. improve correctness of target locations), motion models have been normally used in predictive tracking frameworks. Most of the methods presented in Chapter 3 modelled the target motion explicitly. These motion models are suitable for specific applications, particularly those which deal with smooth movements. A new motion model method in Chapter 4 has been developed to handle motion variations and unexpected movements. Experiments in Chapter 4 have shown this approach can enhance target prediction. Note also that there is no motion learning mechanism in FMCMC-MM. The target motion is derived by detecting and matching sparse features. These matches could be used to enhance learning of target motion.

Several works have used multiple appearance models and multiple motion models, often combining them to formm multiple trackers. The approach presented in Chapter

5 integrates multiple motion and appearance models, both of which are created during tracking, into a single unified tracking algorithm using the sampling techniques described in Chapter 3 and Chapter 4. The proposed technique successfully deals with appearance changes and motion variations and significantly improves tracking performance,

6.3 Future Work

A number of improvements to the methods described in Chapter 3, 4 and 5 could be considered if more flexible trackers are to be produced:

- In the current implementations, simple similarity measures (i.e. NCC and Bhattacharya distance) have been used and thresholds have been employed to decide whether a new appearance model is added. Should the target change its appearance often during a long image sequence, many models may be stored, some of which will become irrelevant. A more robust similarity function and improved selection mechanism are needed to improve the quality and reduce the number of appearance models added. In addition, a mechanism is needed to discard out of date appearance models.

- Occlusion detection is an important step, as the motion modelling and prediction mechanisms proposed here rely on the detection and matching of low-level features detected. A robust occlusion detection mechanism could be employed within these tracking algorithms.

- Scale changes are not handled well in the trackers presented here because the trackers use a rectangle to specify a target and templates. It would be both reasonable and inspiring to address scale changes.

- Interaction among targets in multiple target applications can cause issues for trackers, especially when targets occlude each other. By learning target appearances before occlusions occur, a tracker can re-acquire targets afterwards. The trackers described in this thesis have been built upon the MCMC algorithm, providing a natural extension to multiple target tracking.

Appendix A

Algorithms

A.1 Kernel Mean-shift tracking

Algorithm 15 The mean-shift algorithm (adapted from Comaniciu et al. [2003]).

Given the target model $\{\hat{q}\}_{u=1\ldots m}$ and its location $\hat{y}_0$

1. Initialise the location of the target in the current frame with $\hat{y}_0$, compute $\{\hat{p}_u(\hat{y}_0)\}_{u=1\ldots m}$, and evaluate $\rho[\hat{p}(\hat{y}_0), \hat{q}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{y}_0)\hat{q}_u}$

2. Derive the weights $\{w\}_{i=1\ldots n}$ according to Equation A.2.

3. Find the next location of the target candidate

$$\hat{y} = \left[\frac{\sum_{i=1}^{n} x_i^* w_i g(\| \frac{\hat{y}_0 - x_i^*}{h} \|^2)}{\sum_{i=1}^{n} w_i g(\| \frac{\hat{y}_0 - x_i^*}{h} \|^2)} \right] \tag{A.1}$$

$$w_i = \sum_{u=1}^{m} \delta\left(b(x_i) - u\right) \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0)}} \tag{A.2}$$

   where $g(x) = k^{'}(x)$ is the derivative with respect to x of tracking kernel profile k, $\hat{y}_0$ is the current position of the target, $\hat{y}$ is the new location and w_i is the weight of the ith pixel.

4. Compute $\{\hat{p}_u(\hat{y}_1)\}_{u=1\ldots m}$, and evaluate $\rho[\hat{p}(\hat{y}_1), \hat{q}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{y}_1)\hat{q}_u}$.

5. While $\rho[\hat{p}(\hat{y}_1), \hat{q}] < \rho[\hat{p}(\hat{y}_0), \hat{q}]$
   Do $\hat{y}_1 \leftarrow \frac{1}{2}(\hat{y}_0 + \hat{y}_1)$.
   Evaluate $\rho[\hat{p}(\hat{y}_1), \hat{q}]$.

6. If $\|\hat{y}_1 - \hat{y}_0\| < \epsilon$ Stop. (ϵ is a small number to stop the iteration.)
   Otherwise Set $\hat{y}_0 \leftarrow \hat{y}_1$ and go to the Step 2.

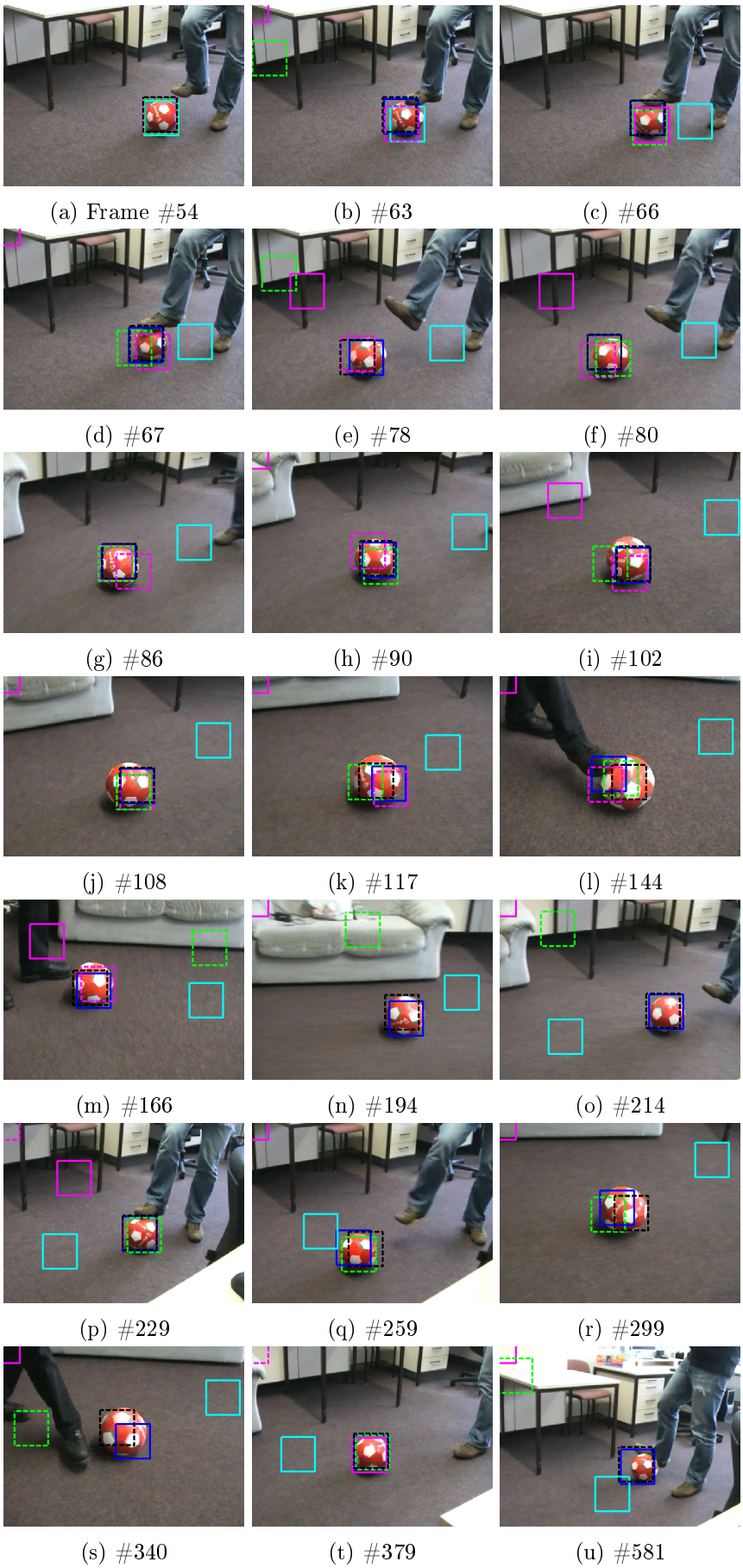
A.2 Kalman filter

Time Update ("Predict")	**Measurement Update ("Correct")**
1. Project the state ahead. $\hat{x}_t^- = A\hat{x}_{t-1} + Bu_{t-1}.$ (A.3) The $\hat{x}^-$ a prior state estimate or the predicted state.	1. Compute the Kalman gain. $K_t = P_t^- H^T (HP_t^- H^T + R)^{-1}.$ (A.5) R is the noise associated with measurement process
2. Project the error covariance ahead. $\hat{P}_t^- = AP_{t-1}A^T + Q.$ (A.4) $\hat{P}_t^-$ is the covariance estimate, Q is the covariance of the noise associated this state prediction process.	2. Update estimate with measurement. $\hat{x}_t = \hat{x}_t^- + K(z_t - H\hat{x}_t^-).$ (A.6) The $\hat{x}$ is a posterior estimate 3. Update the error covariance. $P_t = (I - K_t H)P_t^-.$ (A.7)

Table A.1: Kalman filter algorithm (Welch and Bishop [1995]).

Appendix B

Tracking Results for Chapter 3



(a) Frame #54 (b) #63 (c) #66

(d) #67 (e) #78 (f) #80

(g) #86 (h) #90 (i) #102

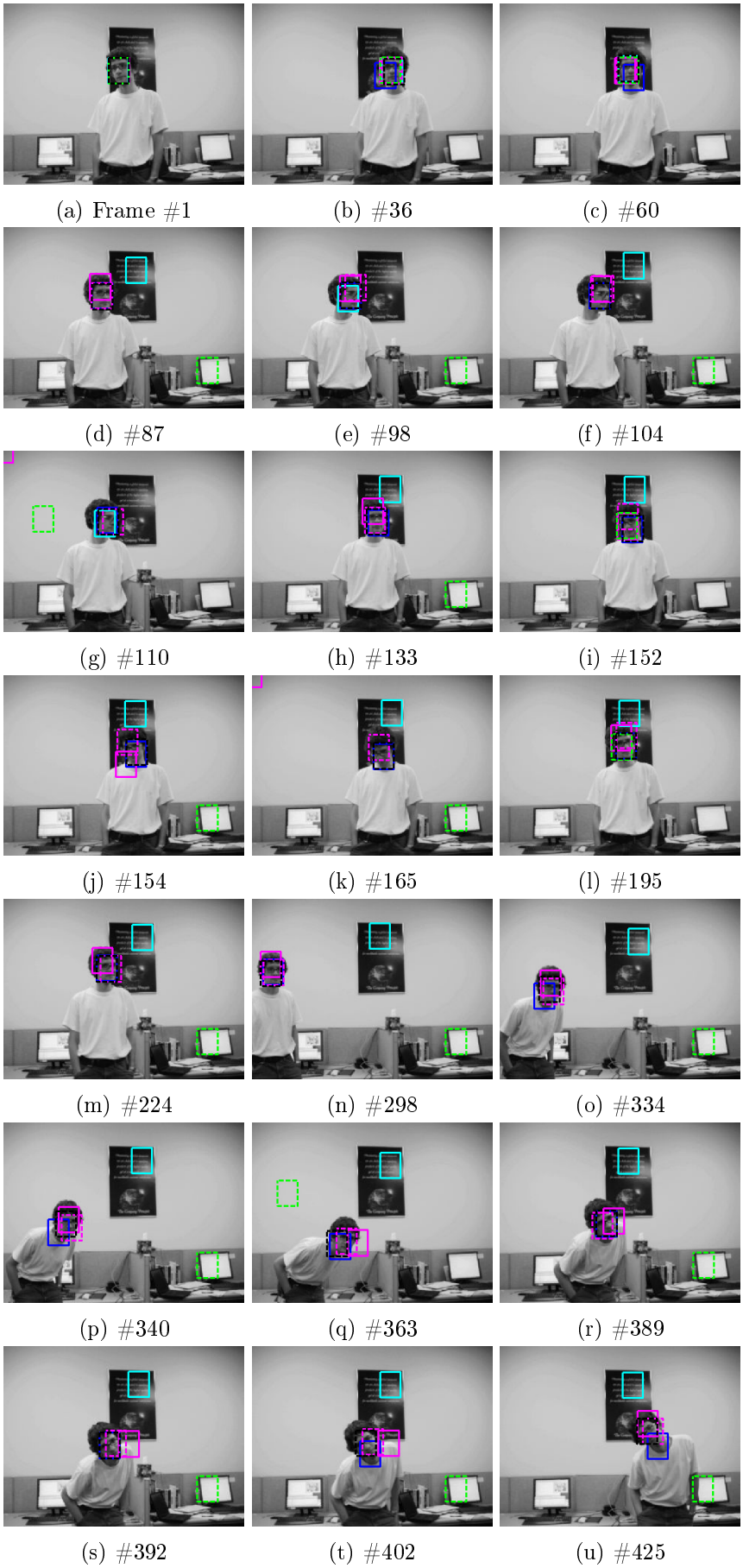
(j) #108 (k) #117 (l) #144

(m) #166 (n) #194 (o) #214

(p) #229 (q) #259 (r) #299

(s) #340 (t) #379 (u) #581

Figure B.1: Tracking results of the Rolling Ball sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #1 (b) #36 (c) #60

(d) #87 (e) #98 (f) #104

(g) #110 (h) #133 (i) #152

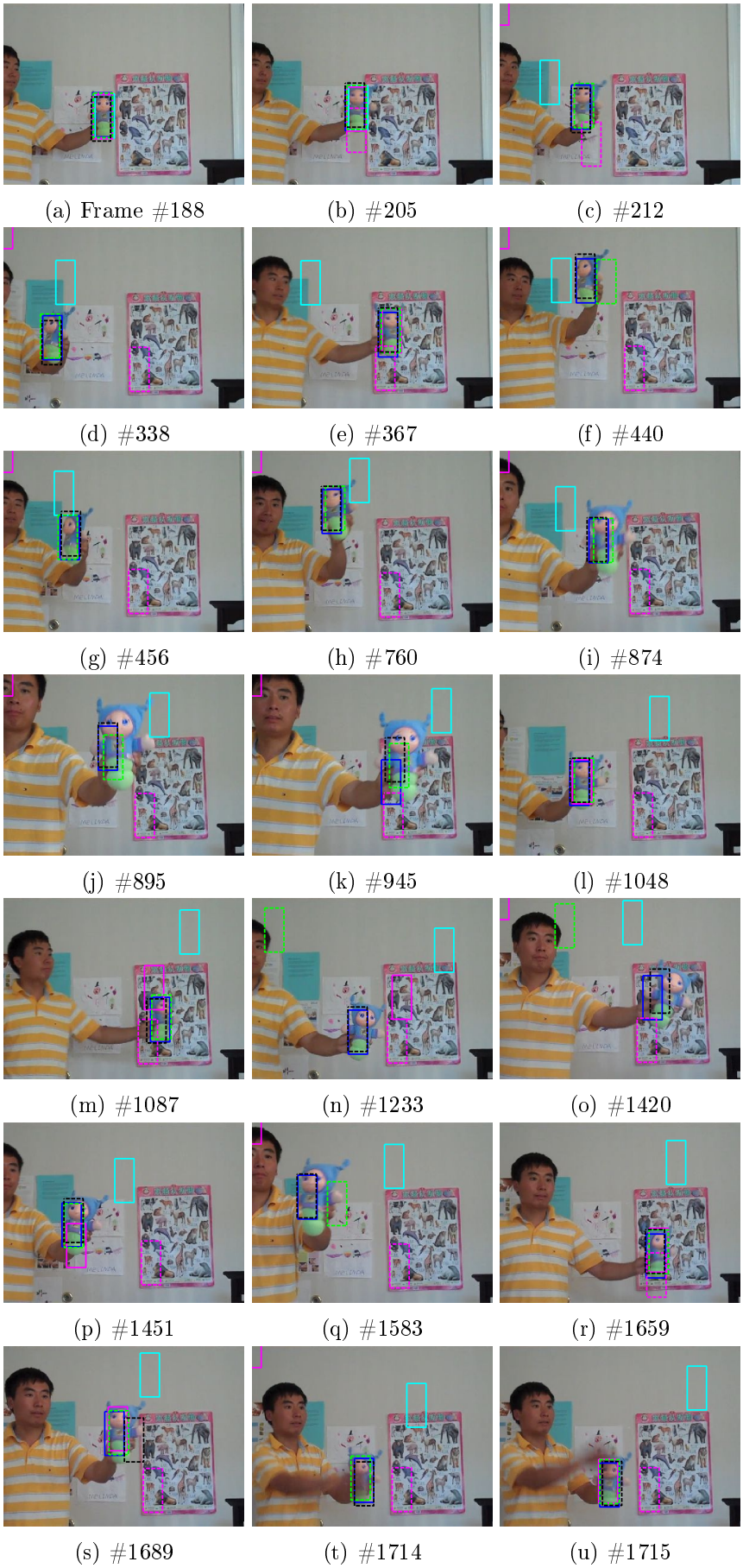
(j) #154 (k) #165 (l) #195

(m) #224 (n) #298 (o) #334

(p) #340 (q) #363 (r) #389

(s) #392 (t) #402 (u) #425

Figure B.2: Tracking results of the David2 sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #188 (b) #205 (c) #212

(d) #338 (e) #367 (f) #440

(g) #456 (h) #760 (i) #874

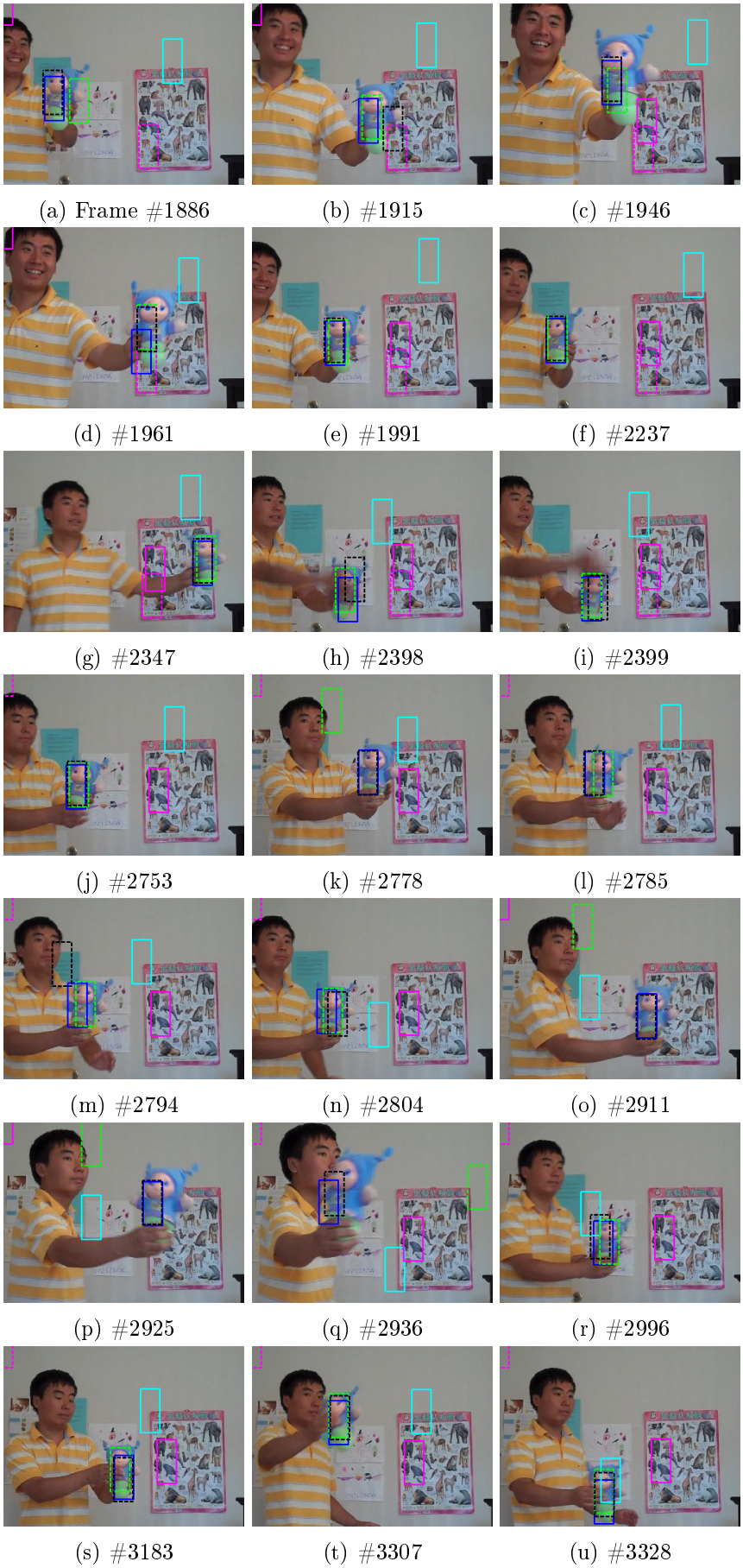
(j) #895 (k) #945 (l) #1048

(m) #1087 (n) #1233 (o) #1420

(p) #1451 (q) #1583 (r) #1659

(s) #1689 (t) #1714 (u) #1715

Figure B.3: Tracking results of the Doll sequence (Part 1). MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #1886 (b) #1915 (c) #1946

(d) #1961 (e) #1991 (f) #2237

(g) #2347 (h) #2398 (i) #2399

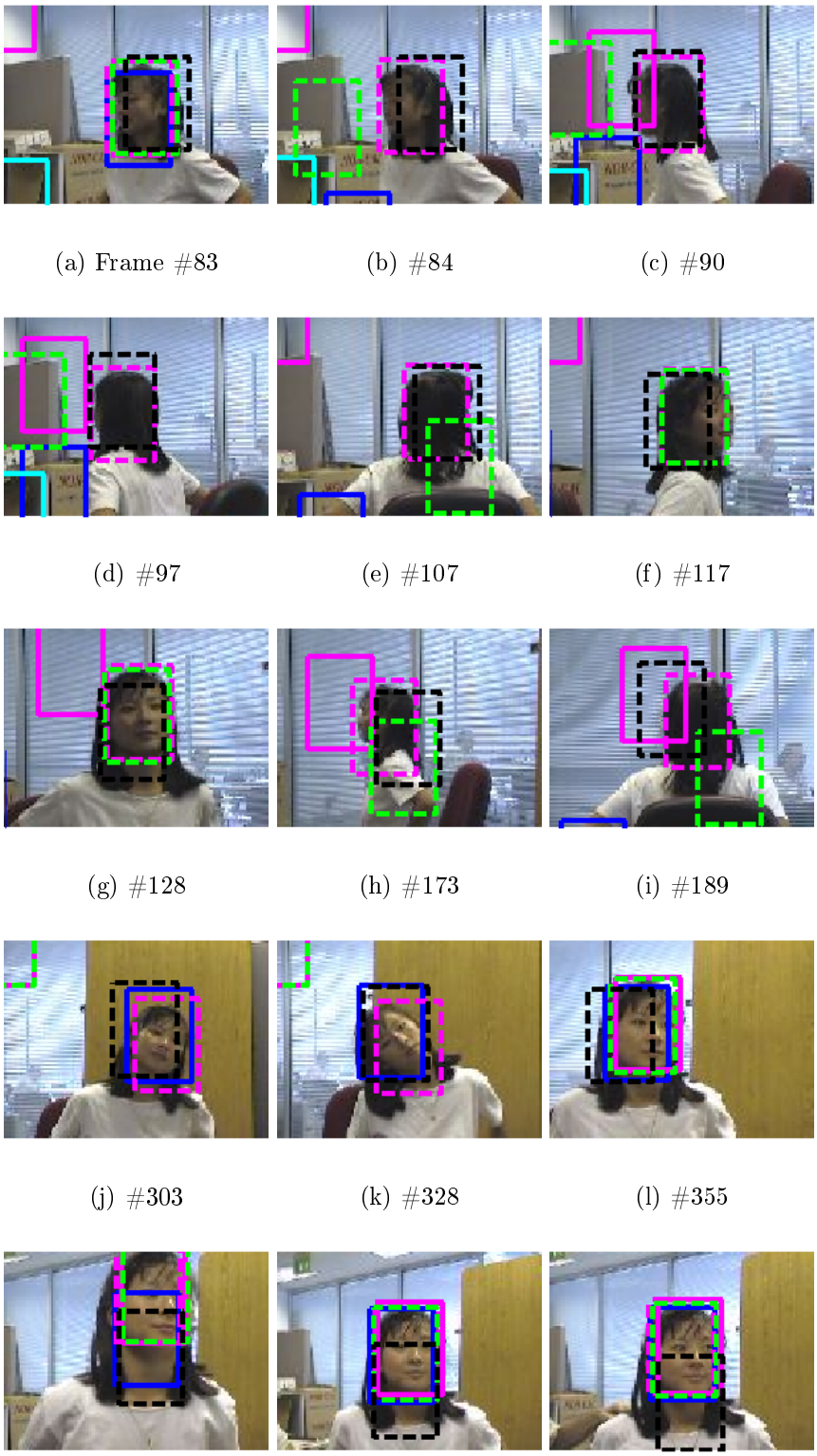
(j) #2753 (k) #2778 (l) #2785

(m) #2794 (n) #2804 (o) #2911

(p) #2925 (q) #2936 (r) #2996

(s) #3183 (t) #3307 (u) #3328

Figure B.4: Tracking results of the Doll sequence (Part 2). MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #83 (b) #84 (c) #90

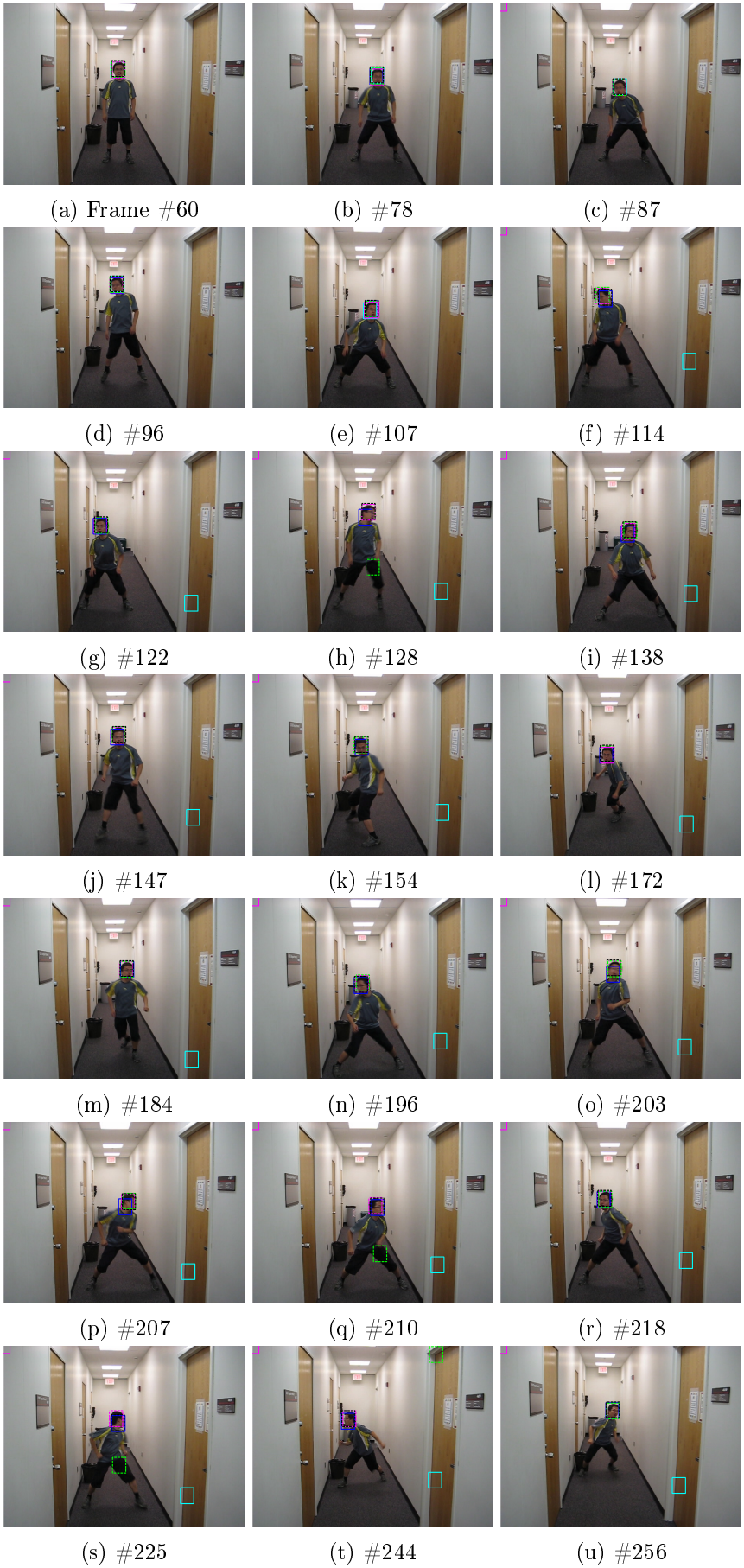
(d) #97 (e) #107 (f) #117

(g) #128 (h) #173 (i) #189

(j) #303 (k) #328 (l) #355

(m) #386 (n) #404 (o) #490

Figure B.5: Tracking results of the Girl sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #60 (b) #78 (c) #87

(d) #96 (e) #107 (f) #114

(g) #122 (h) #128 (i) #138

(j) #147 (k) #154 (l) #172

(m) #184 (n) #196 (o) #203

(p) #207 (q) #210 (r) #218

(s) #225 (t) #244 (u) #256

Figure B.6: Tracking results of the Boy sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

(a) Frame #272 (b) #284 (c) #300

(d) #320 (e) #339 (f) #357

(g) #363 (h) #386 (i) #403

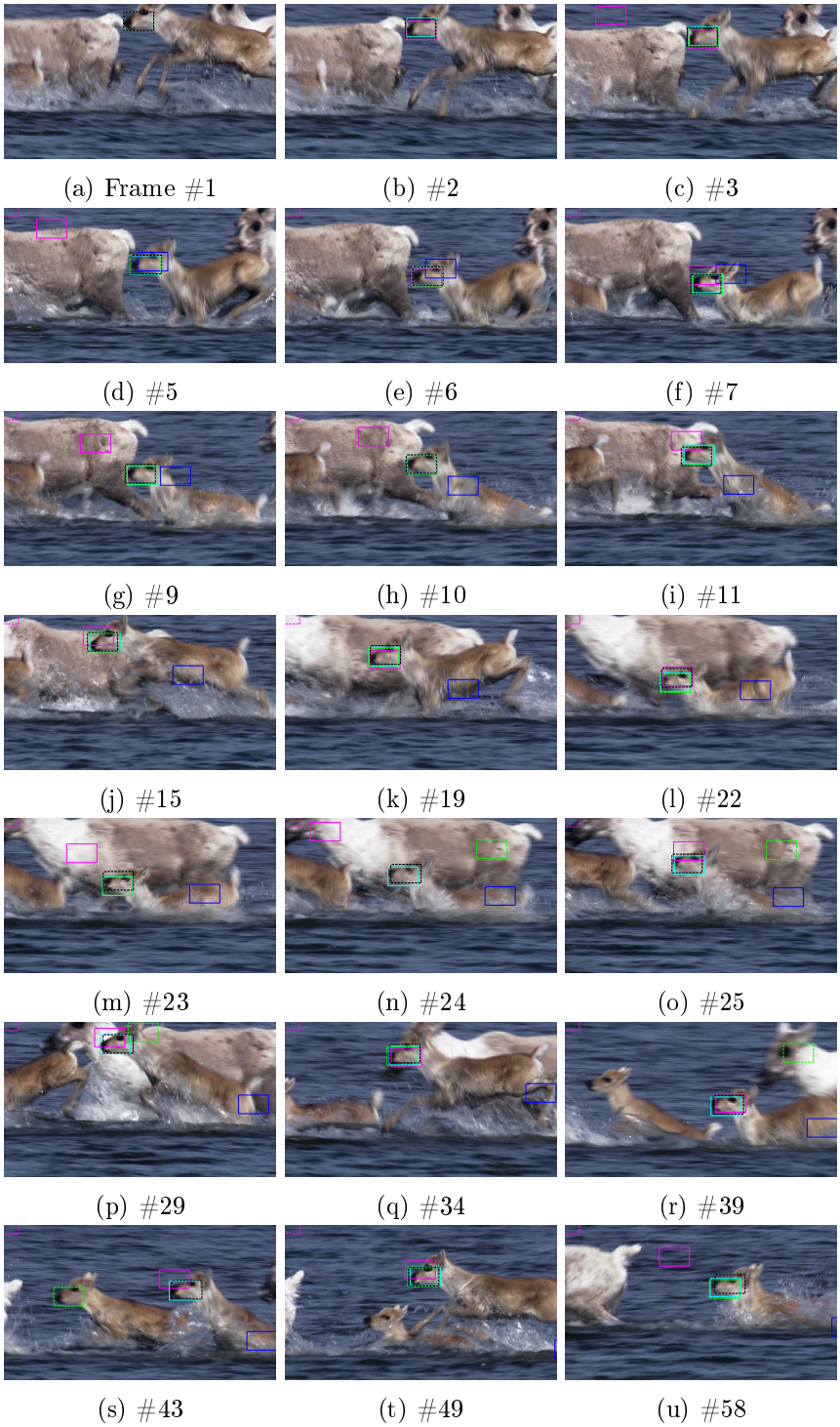
(j) #421 (k) #431 (l) #463

(m) #475 (n) #491 (o) #505

(p) #523 (q) #546 (r) #564

(s) #581 (t) #593 (u) #599

Figure B.7: Tracking results of the Boy sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #1 (b) #2 (c) #3

(d) #5 (e) #6 (f) #7

(g) #9 (h) #10 (i) #11

(j) #15 (k) #19 (l) #22

(m) #23 (n) #24 (o) #25

(p) #29 (q) #34 (r) #39

(s) #43 (t) #49 (u) #58

Figure B.8: Tracking results of the Animal sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

(a) Frame #1 (b) #6 (c) #9

(d) #65 (e) #70 (f) #78

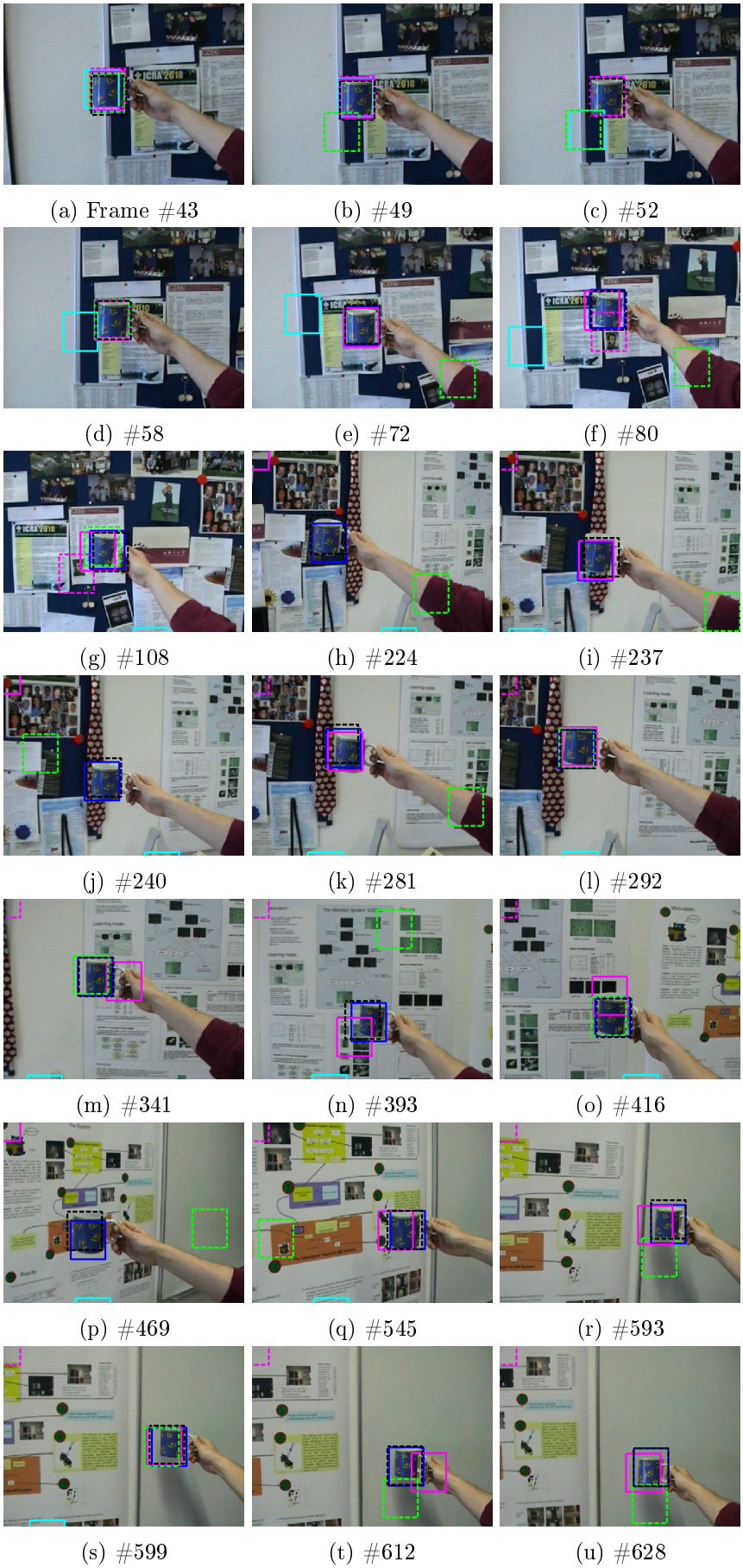
(g) #79 (h) #87 (i) #93

(j) #104 (k) #113 (l) #118

(m) #162 (n) #198 (o) #279

(p) #285 (q) #288 (r) #302

Figure B.9: Tracking results of the Jogging sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #43 (b) #49 (c) #52

(d) #58 (e) #72 (f) #80

(g) #108 (h) #224 (i) #237

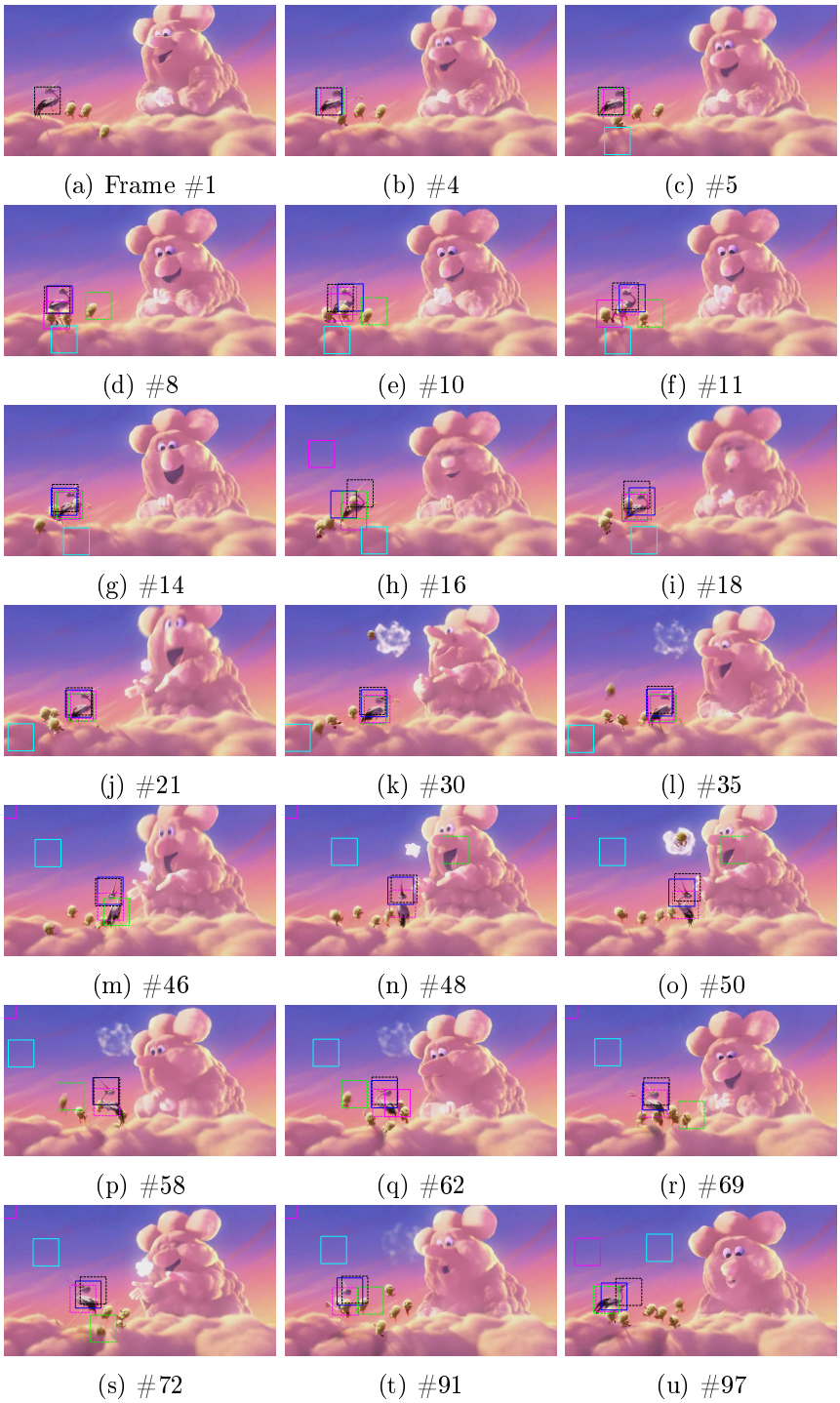
(j) #240 (k) #281 (l) #292

(m) #341 (n) #393 (o) #416

(p) #469 (q) #545 (r) #593

(s) #599 (t) #612 (u) #628

Figure B.10: Tracking results of the Cup sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #1 (b) #4 (c) #5

(d) #8 (e) #10 (f) #11

(g) #14 (h) #16 (i) #18

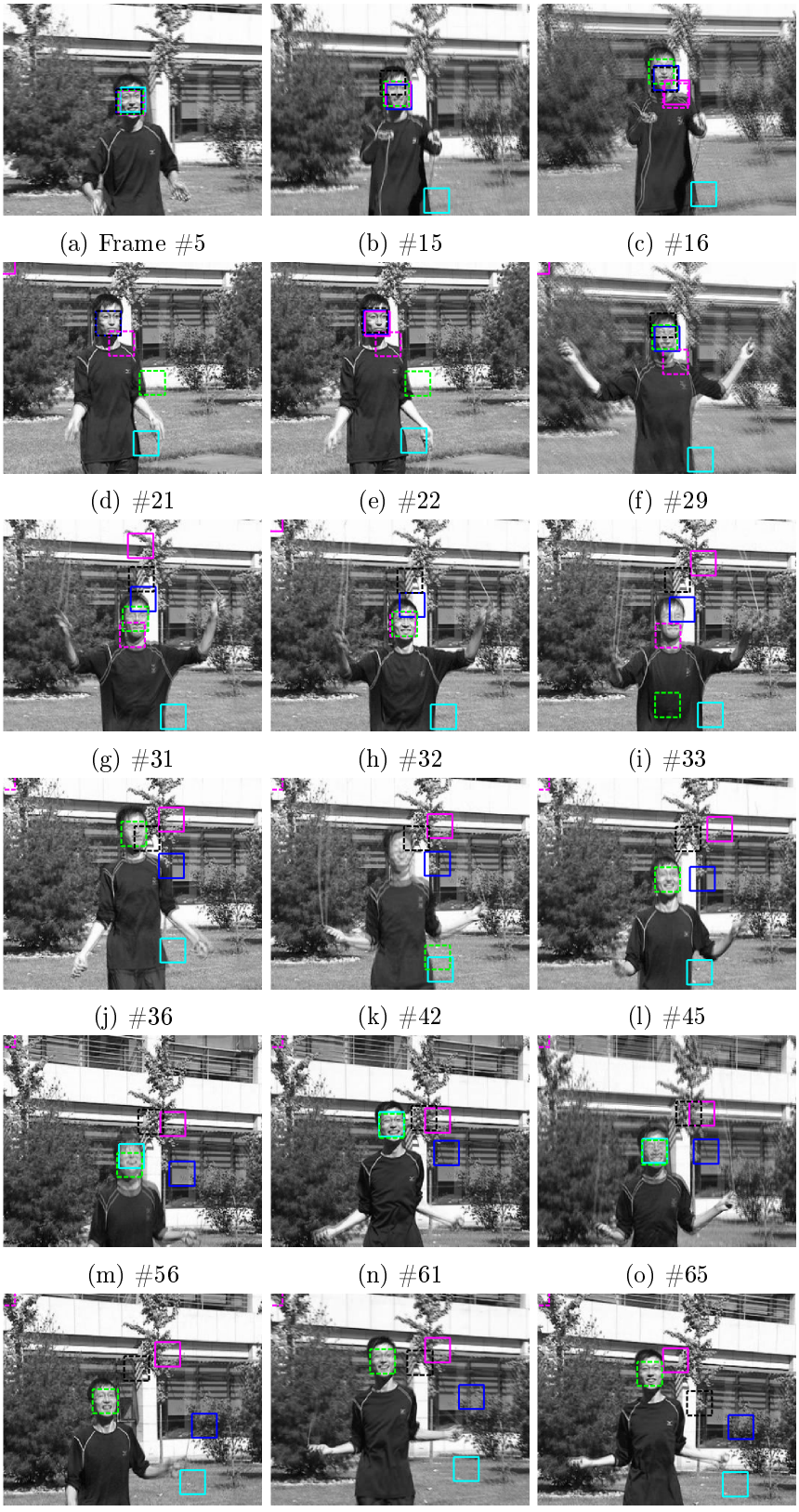
(j) #21 (k) #30 (l) #35

(m) #46 (n) #48 (o) #50

(p) #58 (q) #62 (r) #69

(s) #72 (t) #91 (u) #97

Figure B.11: Tracking results of the Bird2 sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).



(a) Frame #5 (b) #15 (c) #16

(d) #21 (e) #22 (f) #29

(g) #31 (h) #32 (i) #33

(j) #36 (k) #42 (l) #45

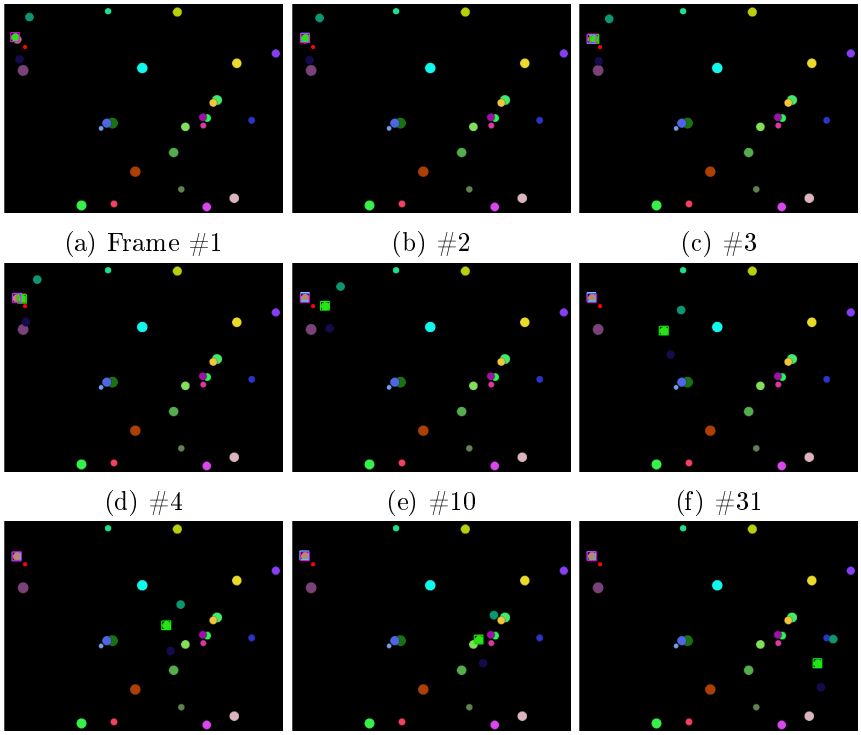
(m) #56 (n) #61 (o) #65

(p) #116 (q) #181 (r) #230

Figure B.12: Tracking results of the Jumping sequence. MCMC-SA ((dashed)black), MCMC (blue), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta).

Appendix C

Tracking Results for Chapter 4

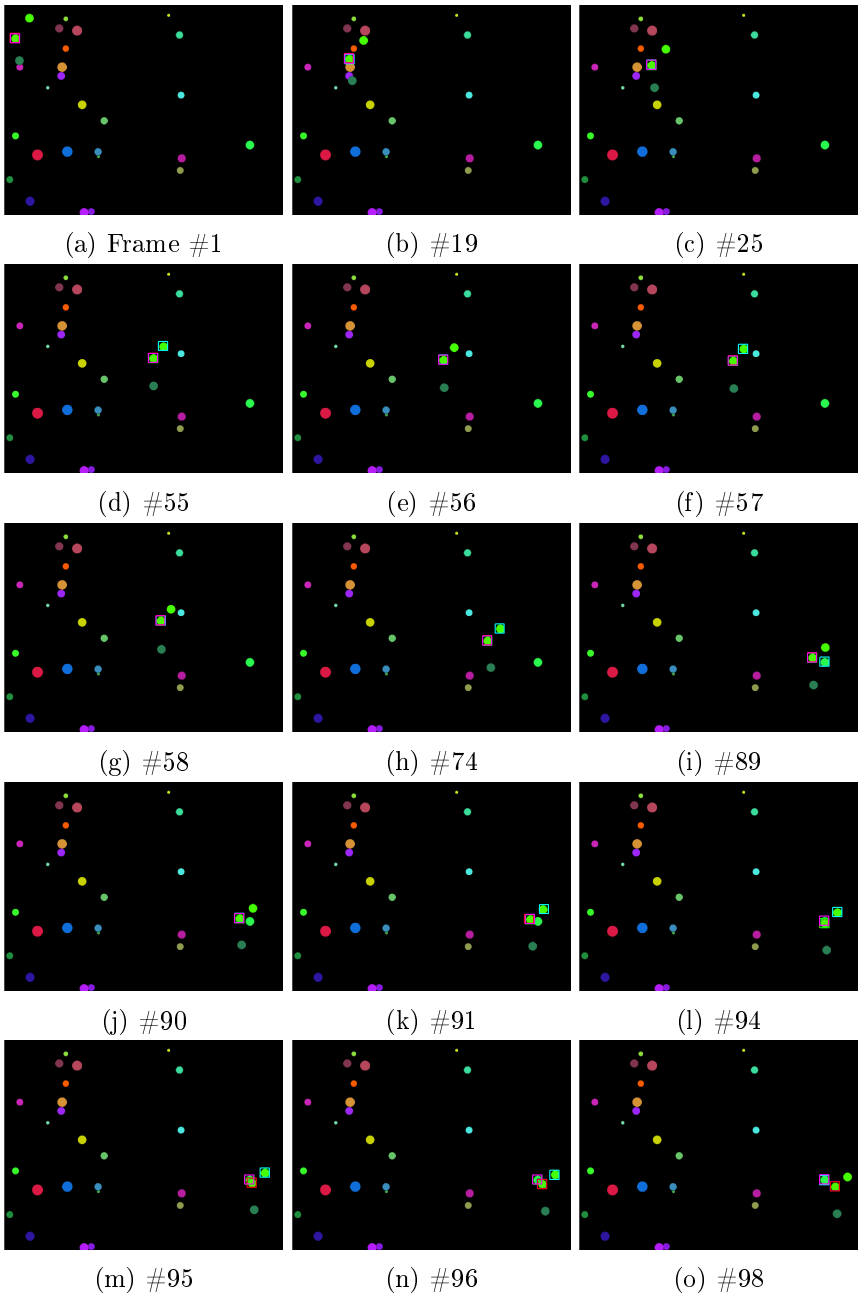


(a) Frame #1 (b) #2 (c) #3

(d) #4 (e) #10 (f) #31

(g) #61 (h) #71 (i) #91

Figure C.1: Tracking results of the Data11 sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



(a) Frame #1 (b) #19 (c) #25

(d) #55 (e) #56 (f) #57

(g) #58 (h) #74 (i) #89

(j) #90 (k) #91 (l) #94

(m) #95 (n) #96 (o) #98

Figure C.2: Tracking results of the Data12 sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

(a) Frame #7 (b) #20 (c) #28

(d) #45 (e) #546 (f) #551

(g) #552 (h) #583 (i) #584

(j) #585 (k) #586 (l) #587

(m) #588 (n) #609 (o) #610

(p) #611 (q) #612 (r) #614

(s) #615 (t) #623 (u) #624

Figure C.3: Tracking results of the Bouncing1 sequence (Part 1). MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

(a) Frame #625 (b) #626 (c) #627

(d) #628 (e) #629 (f) #630

(g) #631 (h) #632 (i) #638

(j) #639 (k) #640 (l) #641

(m) #642 (n) #643 (o) #644

(p) #645 (q) #646 (r) #647

(s) #652 (t) #653 (u) #654

Figure C.4: Tracking results of the Bouncing1 sequence (Part 2). MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

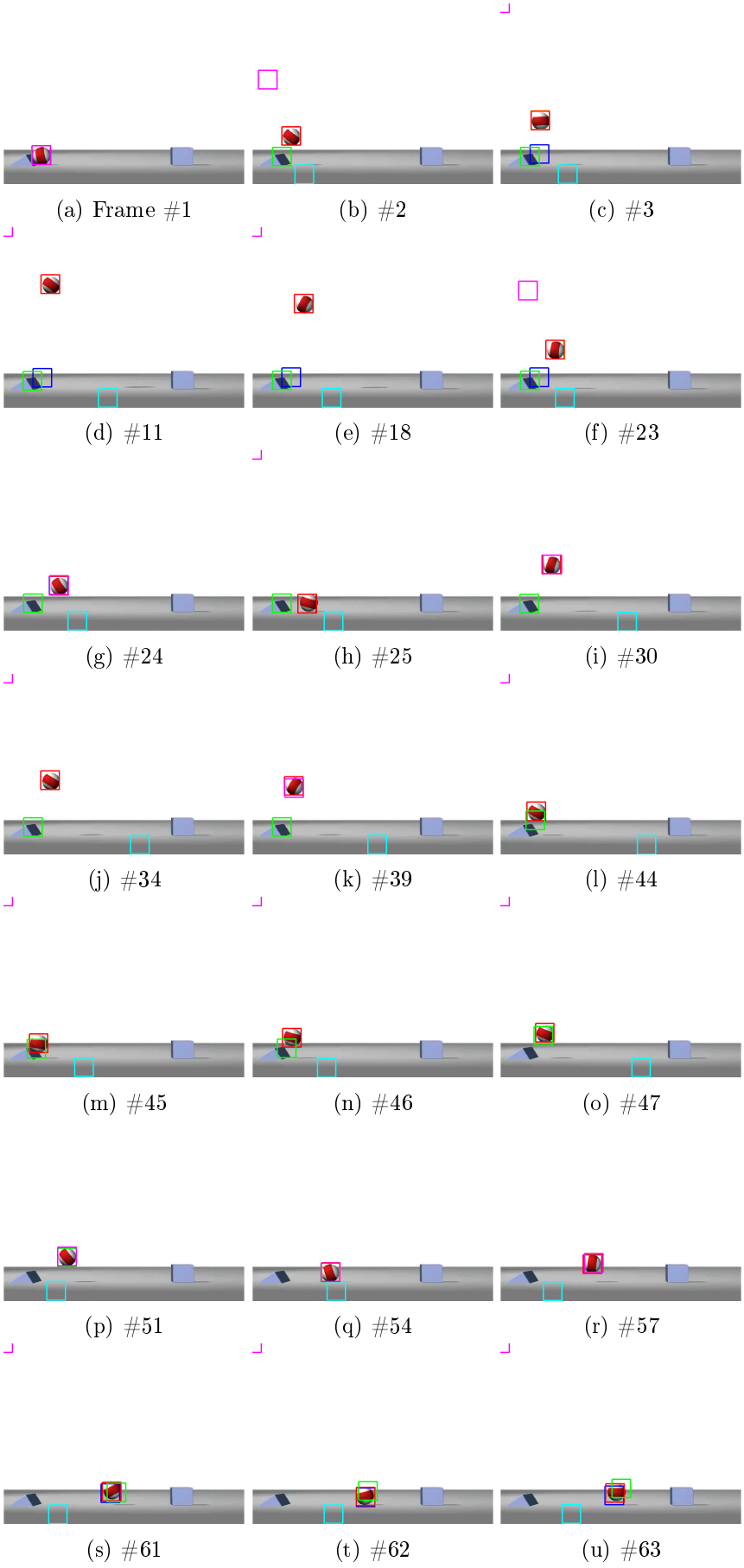
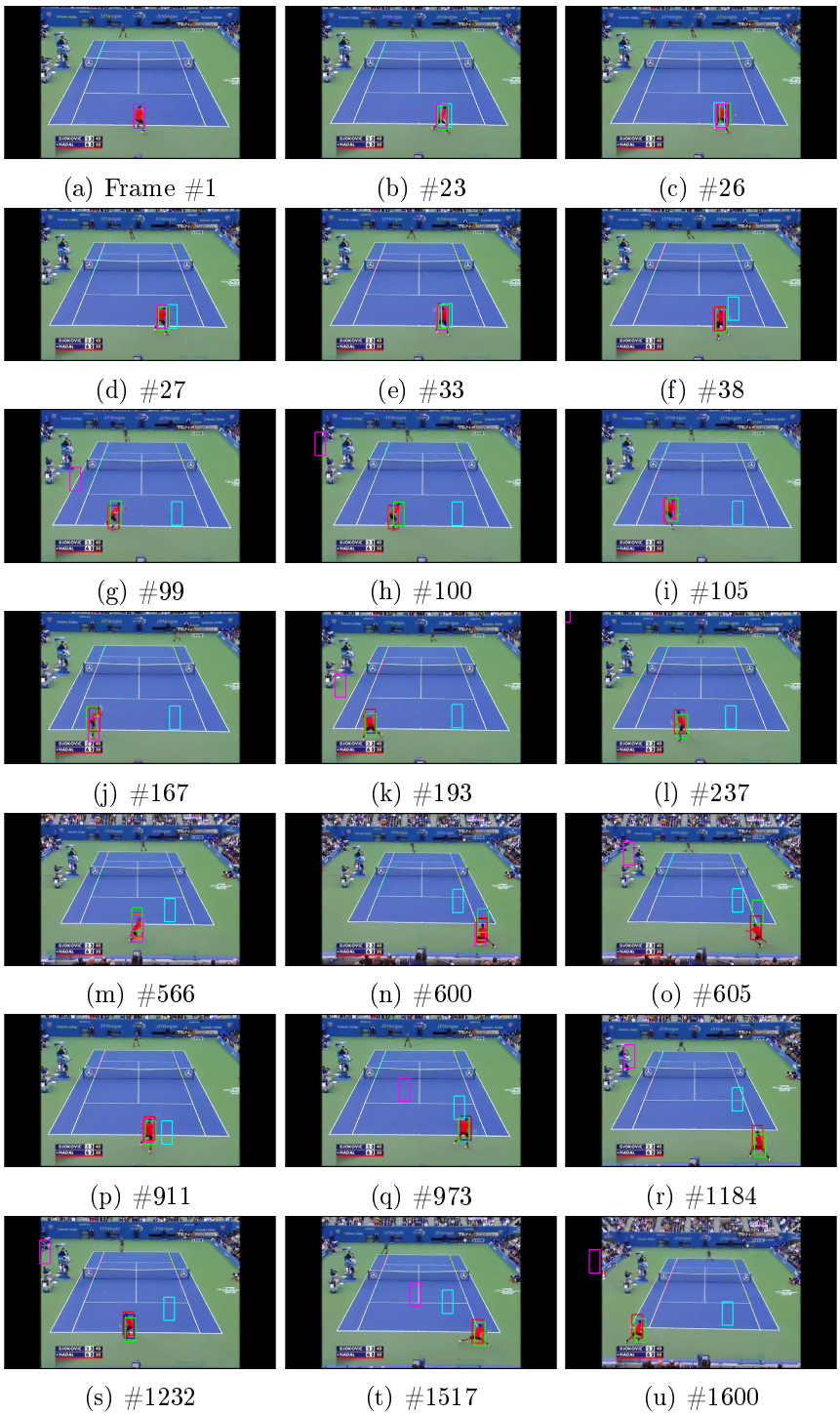


Figure C.5: Tracking results of the Bouncing2 sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



(a) Frame #1 (b) #23 (c) #26

(d) #27 (e) #33 (f) #38

(g) #99 (h) #100 (i) #105

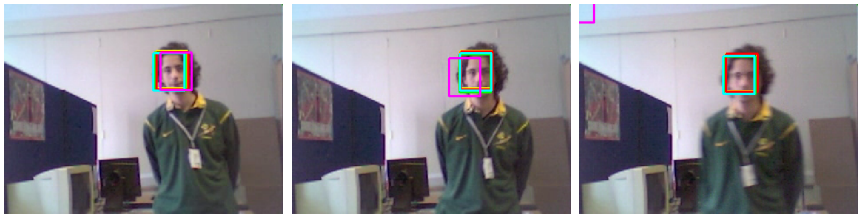
(j) #167 (k) #193 (l) #237

(m) #566 (n) #600 (o) #605

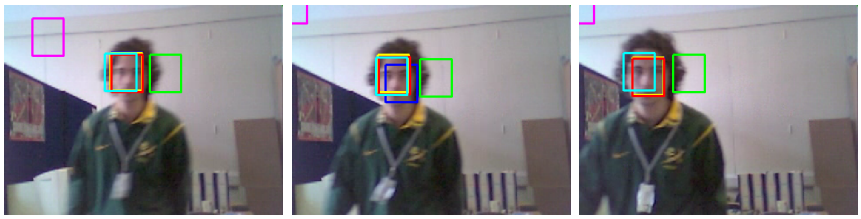
(p) #911 (q) #973 (r) #1184

(s) #1232 (t) #1517 (u) #1600

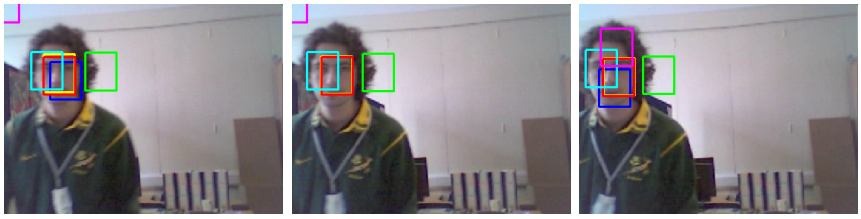
Figure C.6: Tracking results of the Tennis match sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



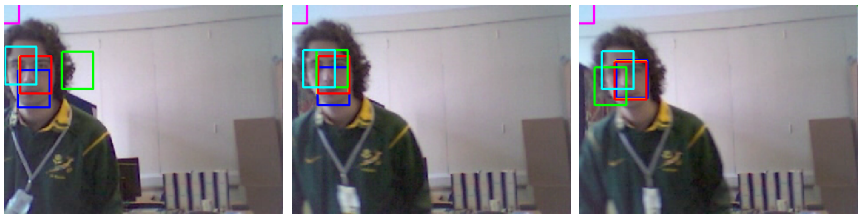
(a) Frame #17 (b) #22 (c) #24



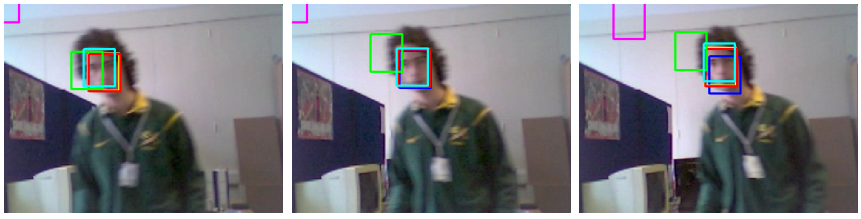
(d) #26 (e) #27 (f) #29



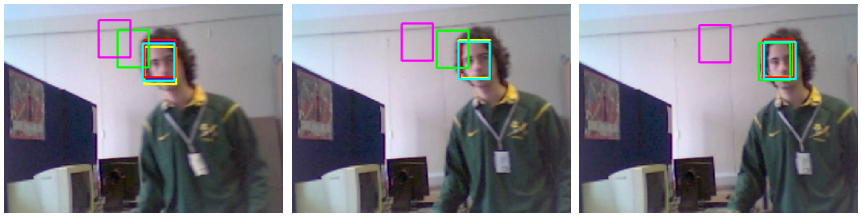
(g) #30 (h) #31 (i) #32



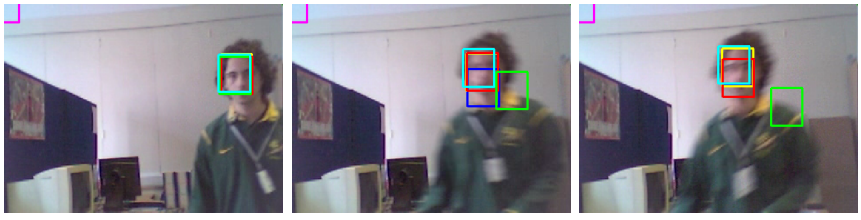
(j) #33 (k) #36 (l) #37



(m) #40 (n) #41 (o) #42



(p) #43 (q) #45 (r) #47



(s) #55 (t) #57 (u) #58

Figure C.7: Tracking results of the Emilio sequence (Part 1). MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

(a) Frame #59 (b) #60 (c) #62

(d) #63 (e) #65 (f) #67

(g) #68 (h) #69 (i) #70

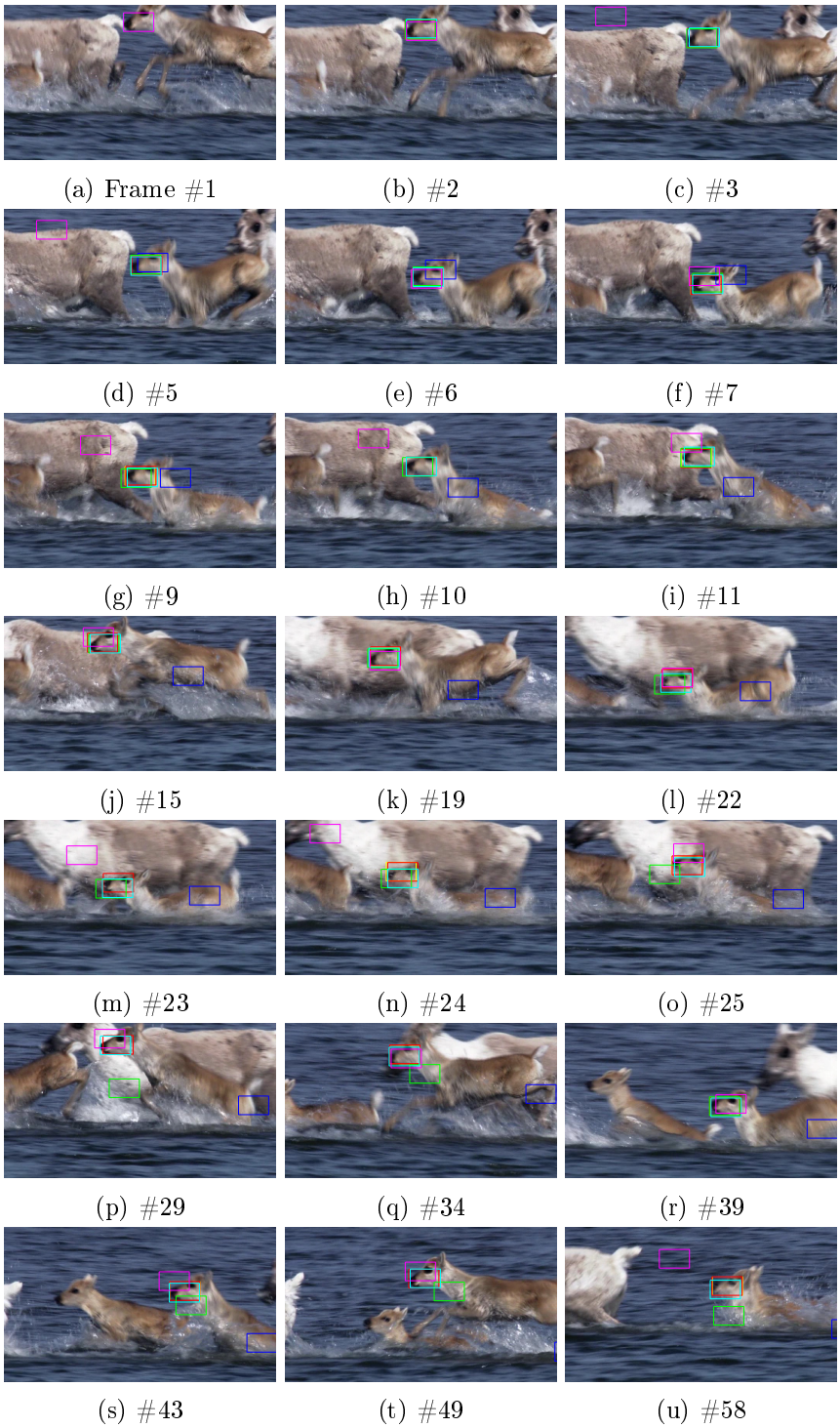
(j) #71 (k) #72 (l) #74

(m) #81 (n) #97 (o) #102

(p) #116 (q) #129 (r) #145

(s) #176 (t) #220 (u) #236

Figure C.8: Tracking results of the Emilio sequence (Part 2). MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



(a) Frame #1 (b) #2 (c) #3

(d) #5 (e) #6 (f) #7

(g) #9 (h) #10 (i) #11

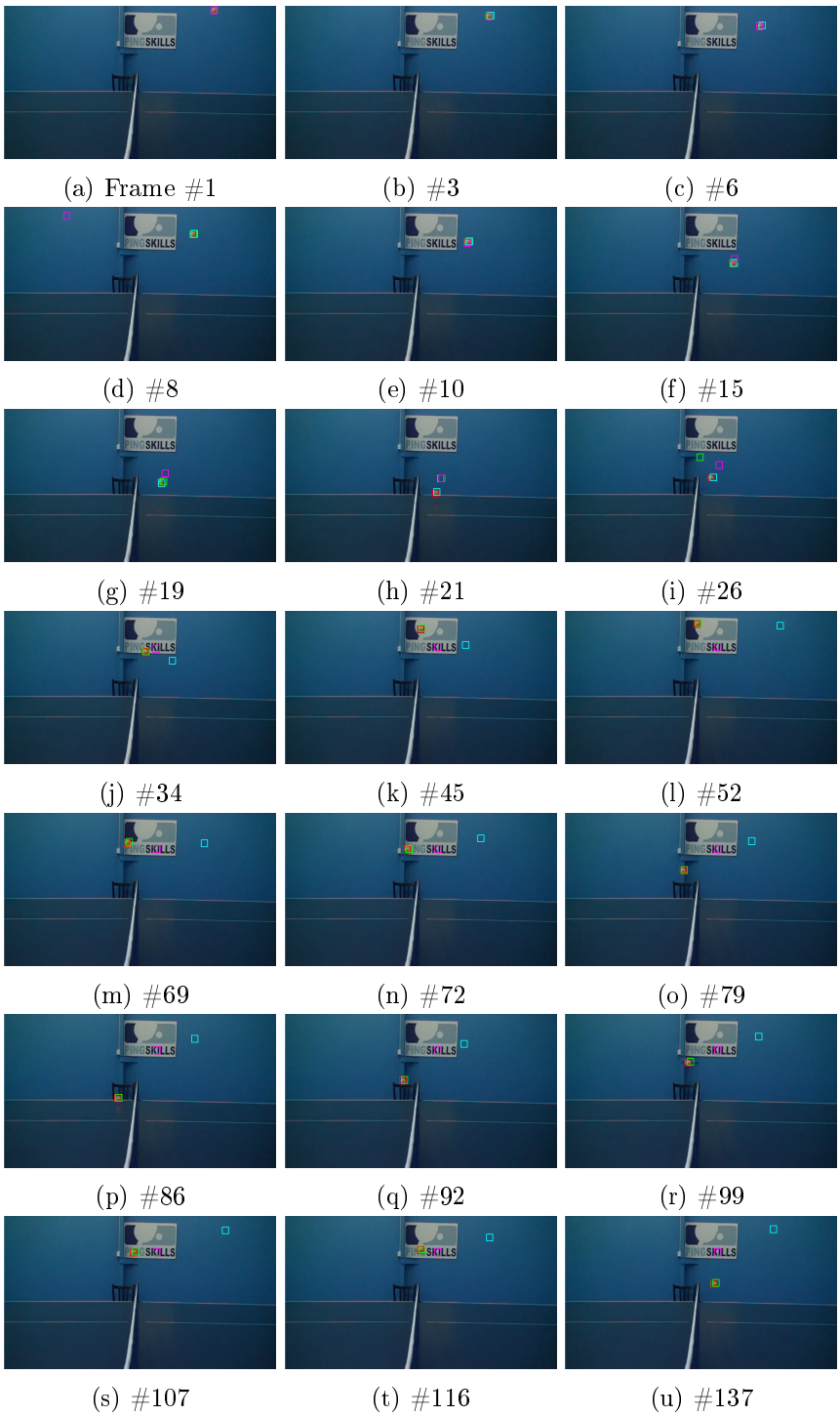
(j) #15 (k) #19 (l) #22

(m) #23 (n) #24 (o) #25

(p) #29 (q) #34 (r) #39

(s) #43 (t) #49 (u) #58

Figure C.9: Tracking results of the Animal sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



(a) Frame #1 (b) #3 (c) #6

(d) #8 (e) #10 (f) #15

(g) #19 (h) #21 (i) #26

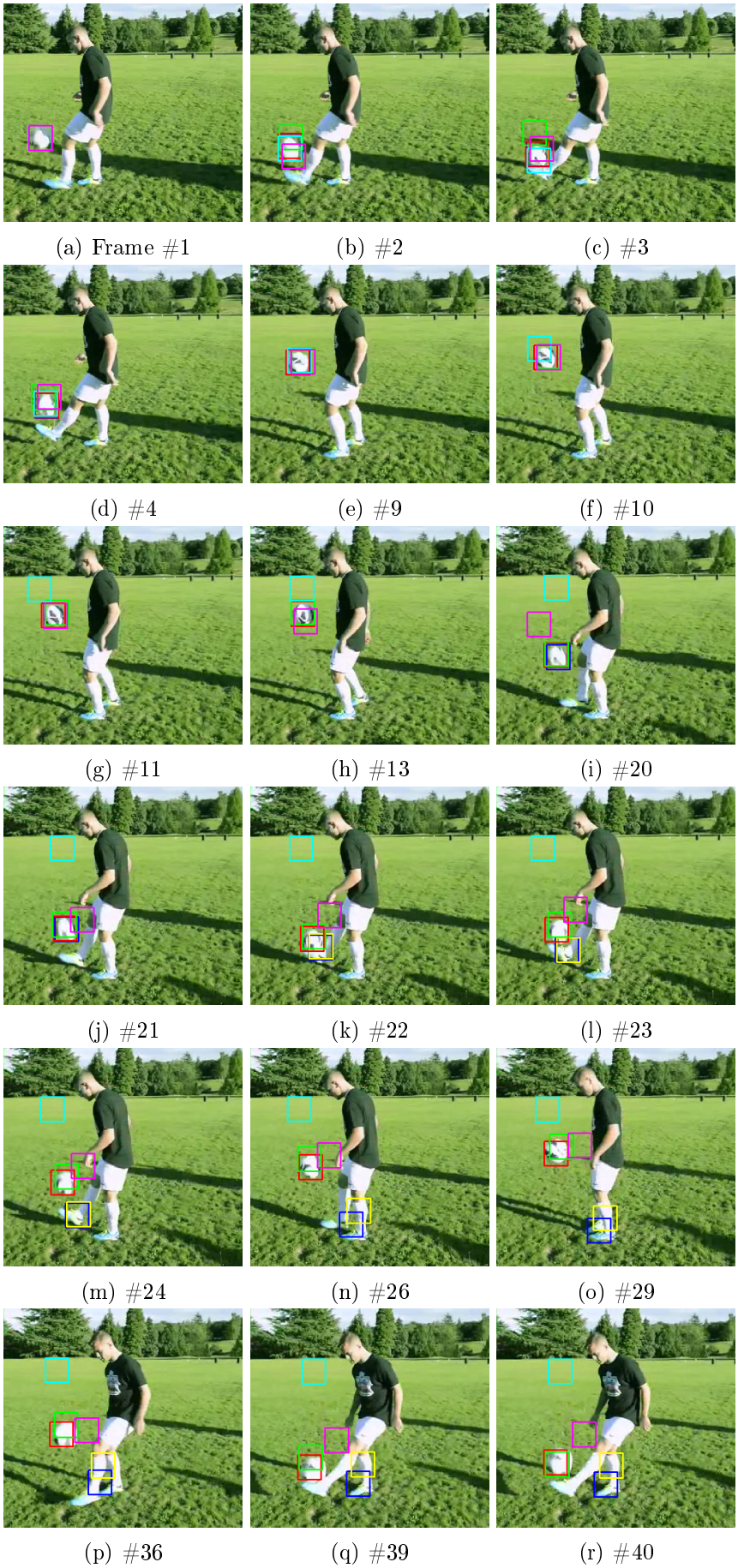
(j) #34 (k) #45 (l) #52

(m) #69 (n) #72 (o) #79

(p) #86 (q) #92 (r) #99

(s) #107 (t) #116 (u) #137

Figure C.10: Tracking results of the Table tennis sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



(a) Frame #1 (b) #2 (c) #3

(d) #4 (e) #9 (f) #10

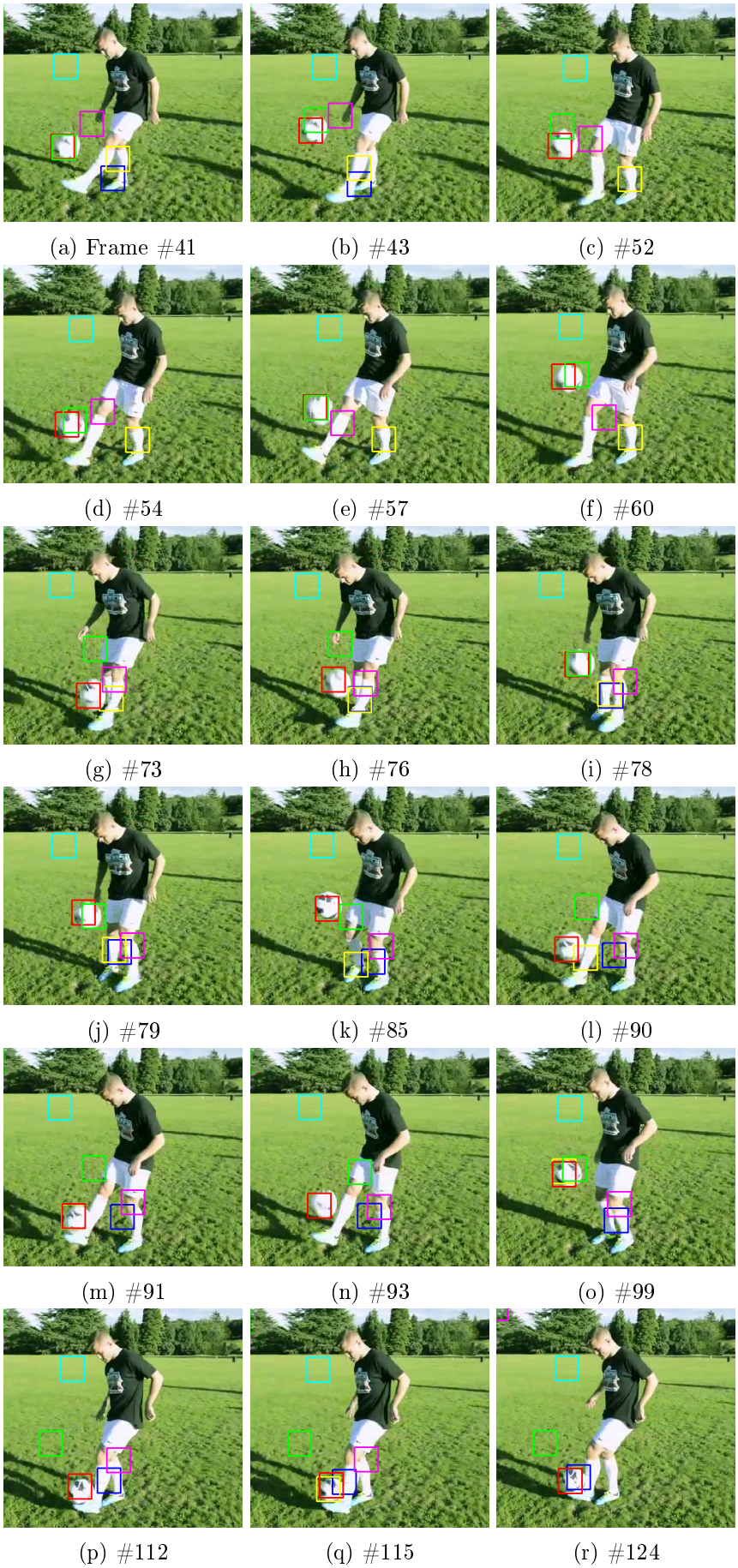
(g) #11 (h) #13 (i) #20

(j) #21 (k) #22 (l) #23

(m) #24 (n) #26 (o) #29

(p) #36 (q) #39 (r) #40

Figure C.11: Tracking results of the Football sequence (Part 1). MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



(a) Frame #41 (b) #43 (c) #52

(d) #54 (e) #57 (f) #60

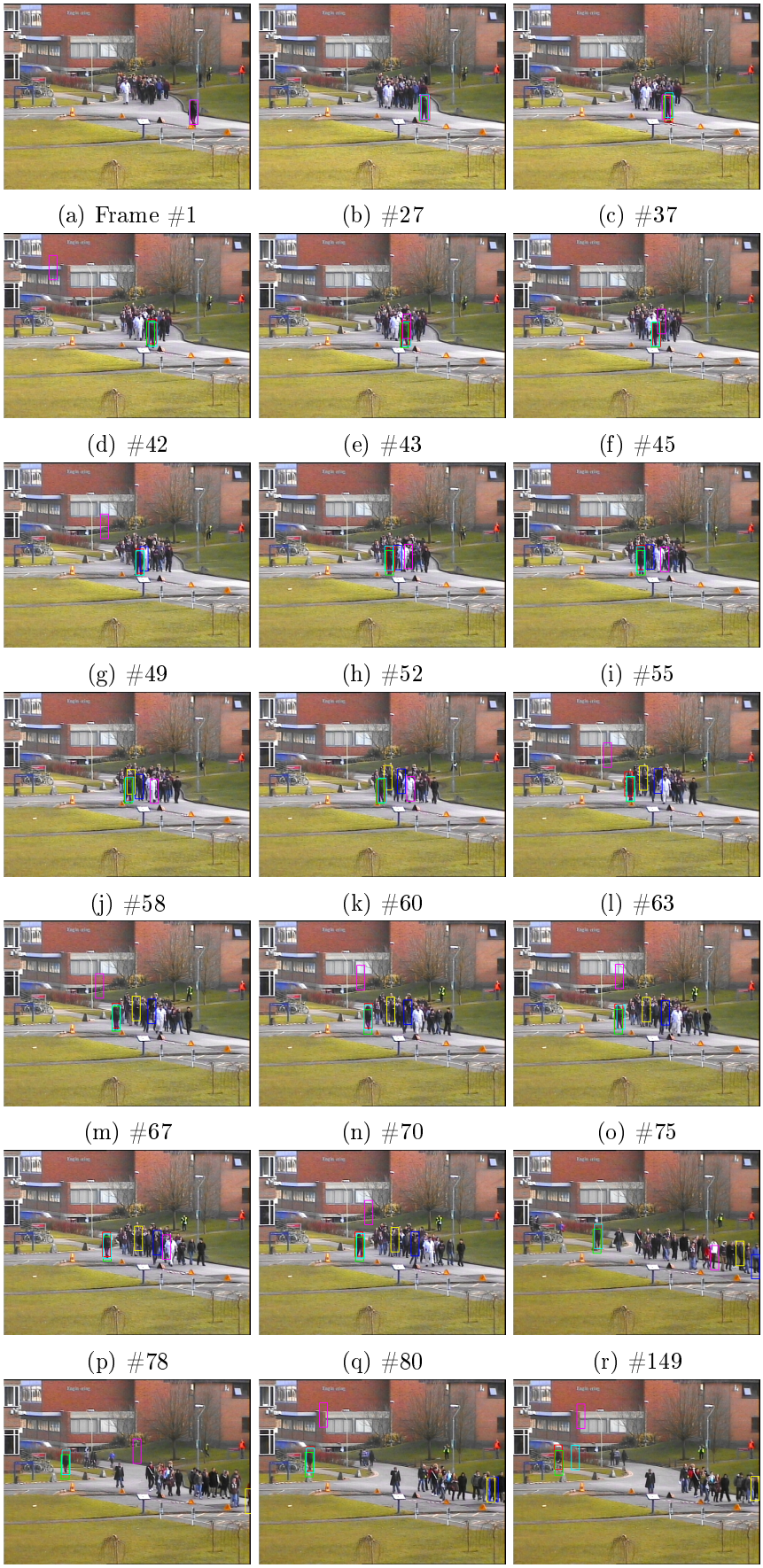
(g) #73 (h) #76 (i) #78

(j) #79 (k) #85 (l) #90

(m) #91 (n) #93 (o) #99

(p) #112 (q) #115 (r) #124

Figure C.12: Tracking results of the Football sequence (Part 2). MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



(a) Frame #1 (b) #27 (c) #37

(d) #42 (e) #43 (f) #45

(g) #49 (h) #52 (i) #55

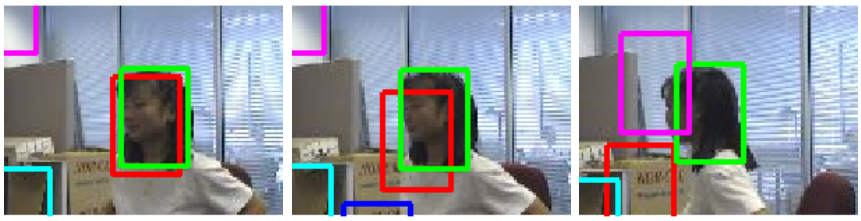
(j) #58 (k) #60 (l) #63

(m) #67 (n) #70 (o) #75

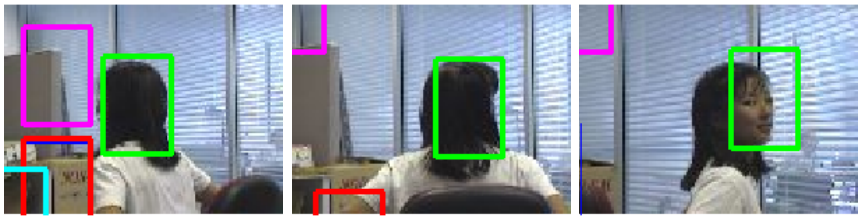
(p) #78 (q) #80 (r) #149

(s) #186 (t) #208 (u) #216

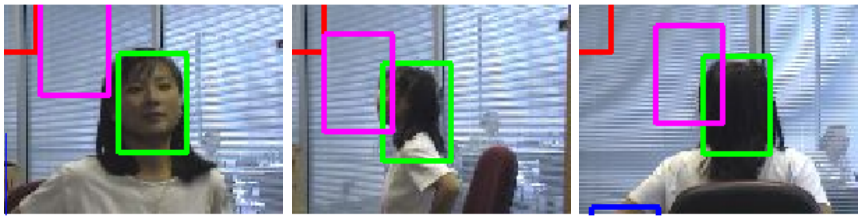
Figure C.13: Tracking results of the PETS09 sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).



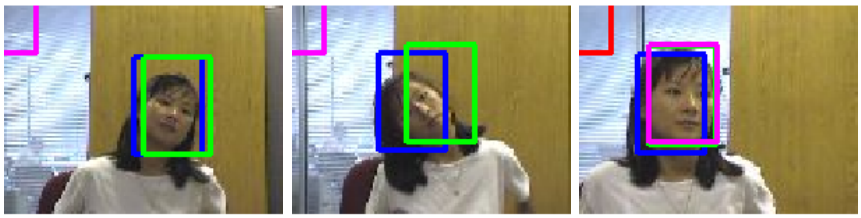
(a) Frame #83 (b) #84 (c) #90



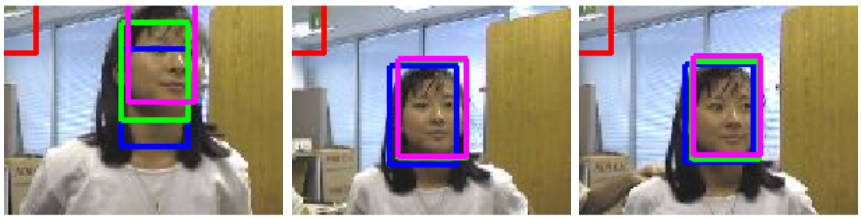
(d) #97 (e) #107 (f) #117



(g) #128 (h) #173 (i) #189



(j) #303 (k) #328 (l) #355

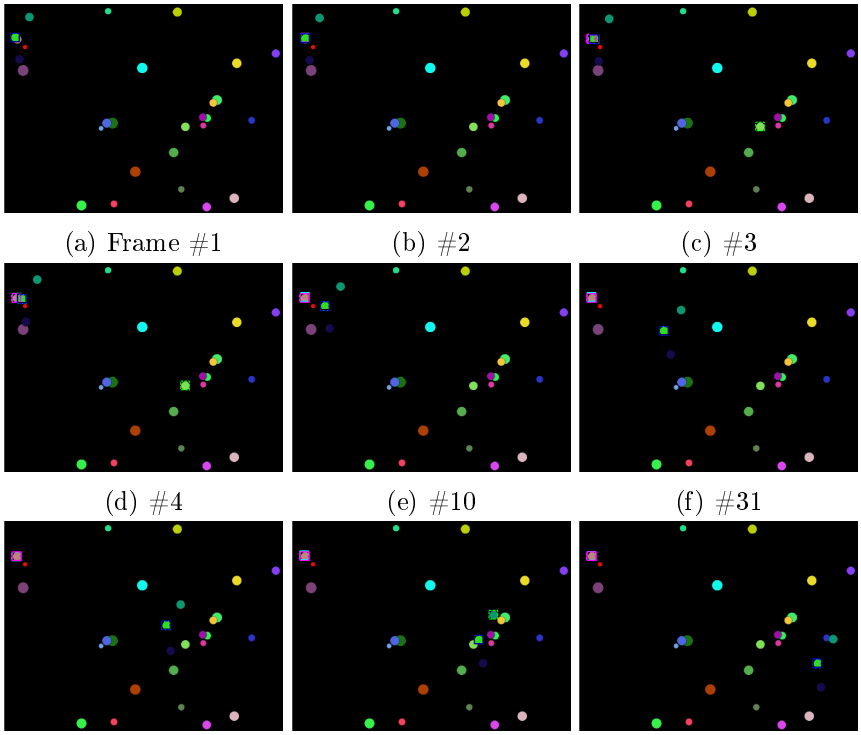


(m) #386 (n) #404 (o) #490

Figure C.14: Tracking results of the Girl sequence. MCMC(blue), FMCMC-C(yellow), FMCMC-S(red), FragTrack(green), IVT(cyan), SB(magenta).

Appendix D

Tracking Results for Chapter 5

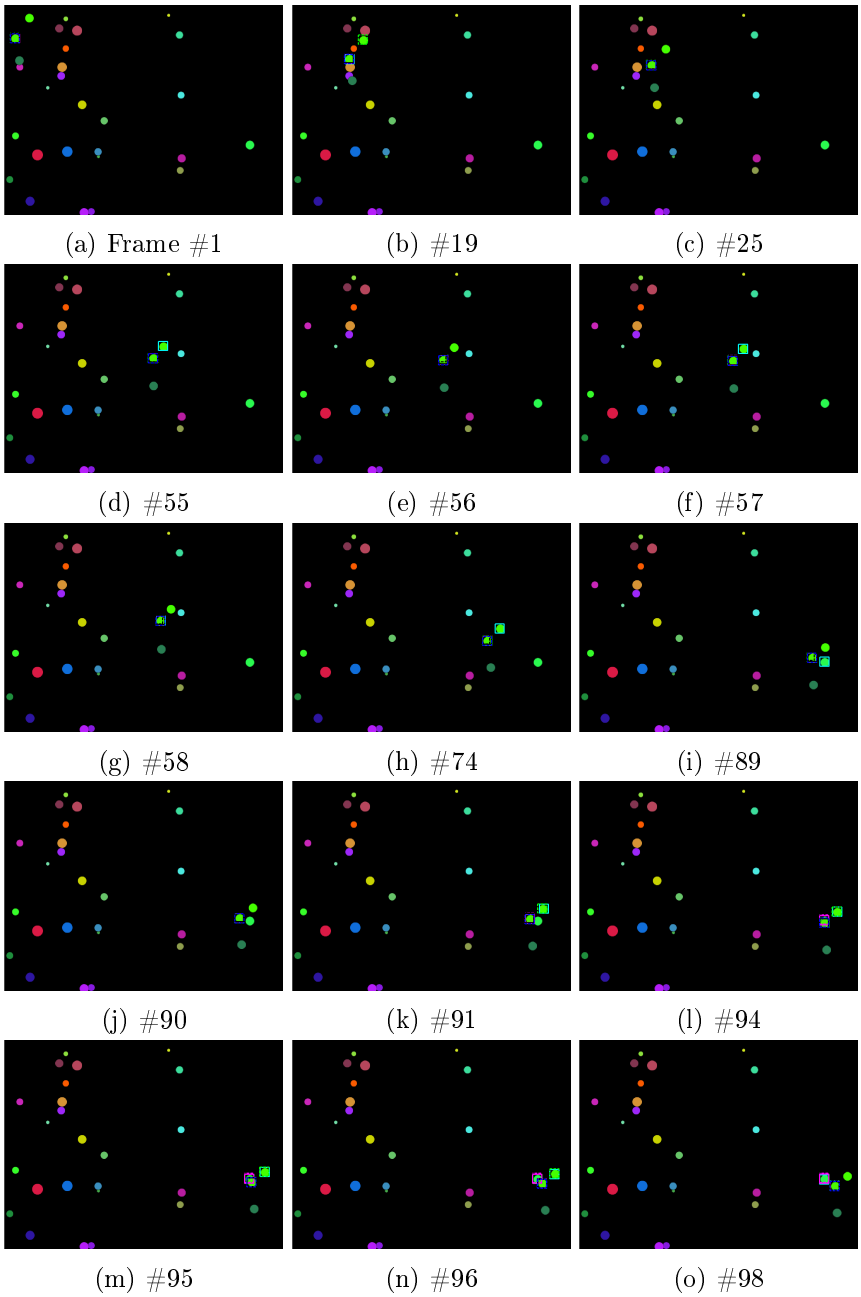


(a) Frame #1 (b) #2 (c) #3

(d) #4 (e) #10 (f) #31

(g) #61 (h) #71 (i) #91

Figure D.1: Tracking results of the Data11 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #19 (c) #25

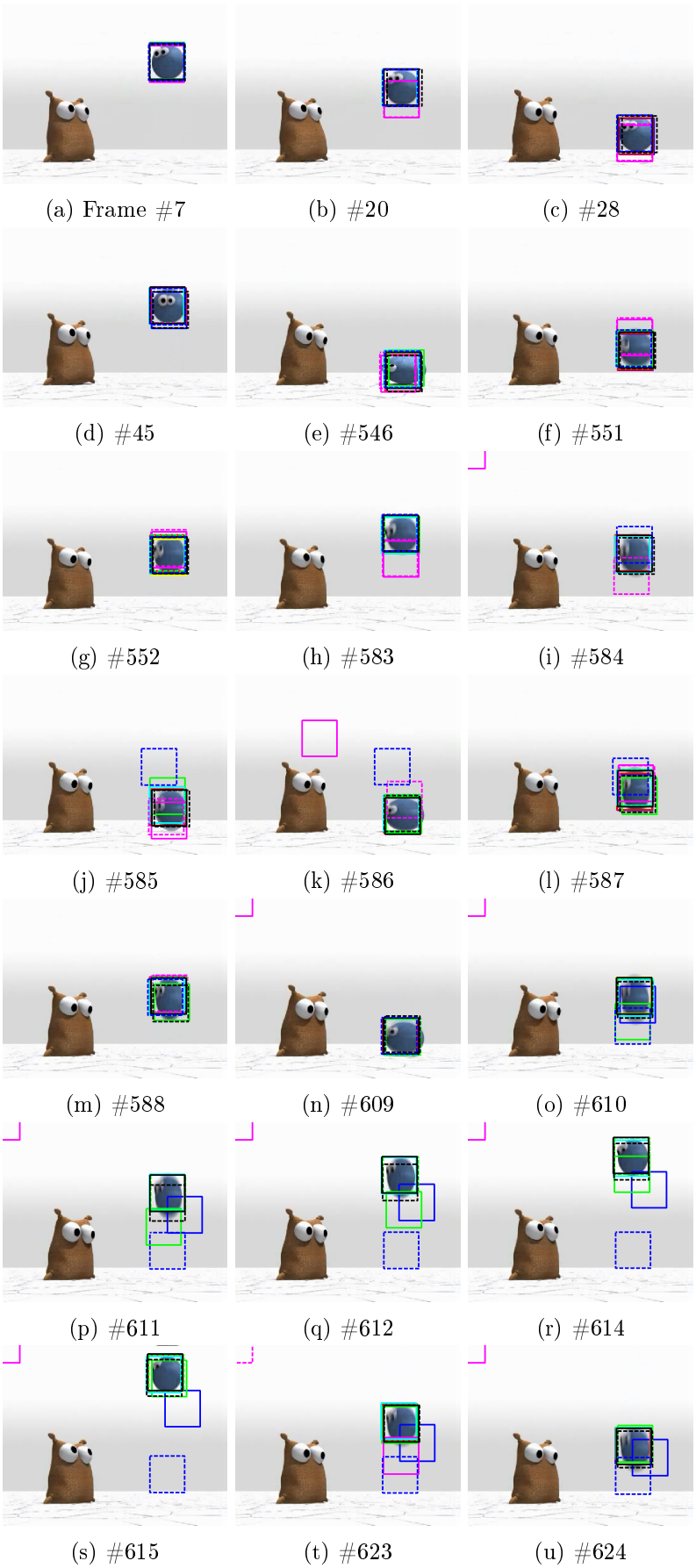
(d) #55 (e) #56 (f) #57

(g) #58 (h) #74 (i) #89

(j) #90 (k) #91 (l) #94

(m) #95 (n) #96 (o) #98

Figure D.2: Tracking results of the Data12 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #7 (b) #20 (c) #28

(d) #45 (e) #546 (f) #551

(g) #552 (h) #583 (i) #584

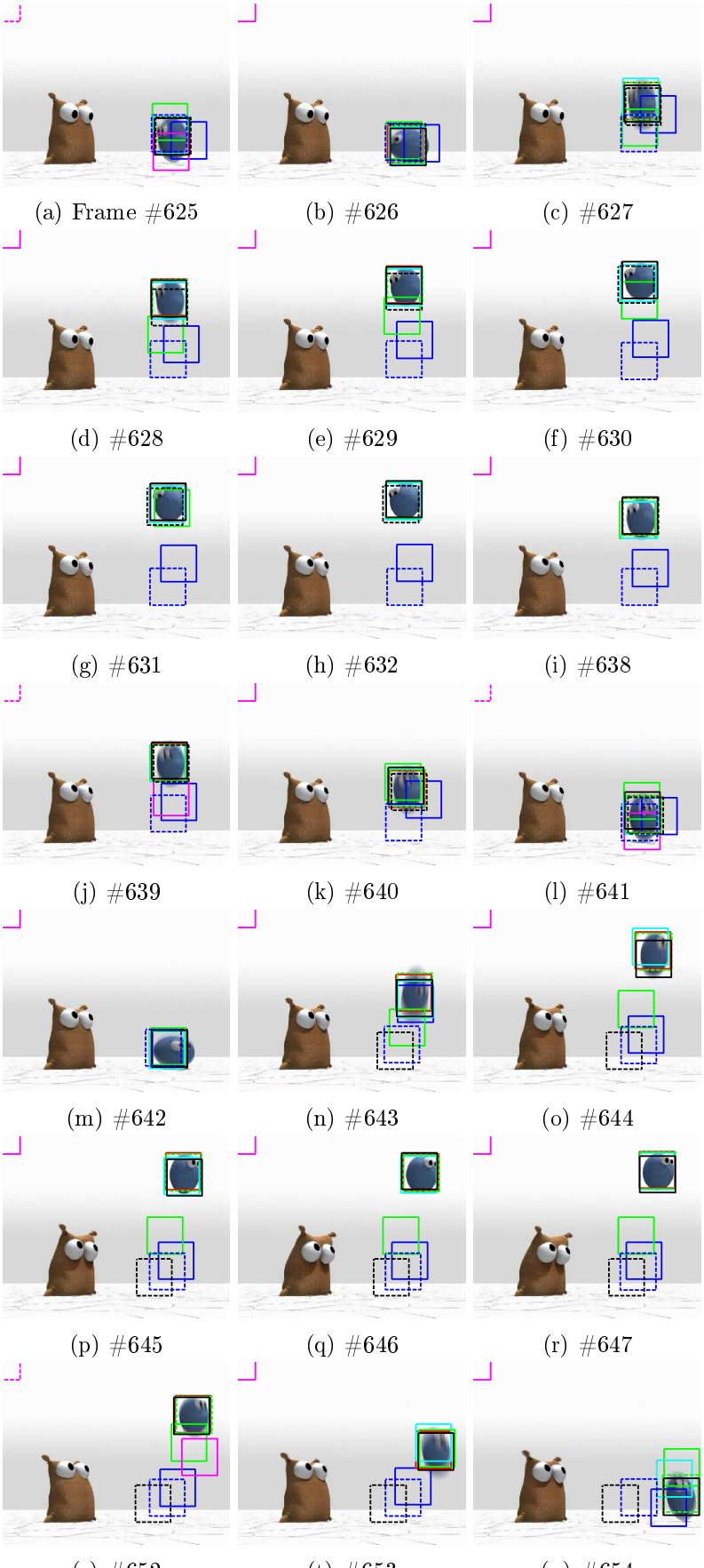
(j) #585 (k) #586 (l) #587

(m) #588 (n) #609 (o) #610

(p) #611 (q) #612 (r) #614

(s) #615 (t) #623 (u) #624

Figure D.3: Tracking results of the Bouncing1 sequence (Part 1). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #625 (b) #626 (c) #627

(d) #628 (e) #629 (f) #630

(g) #631 (h) #632 (i) #638

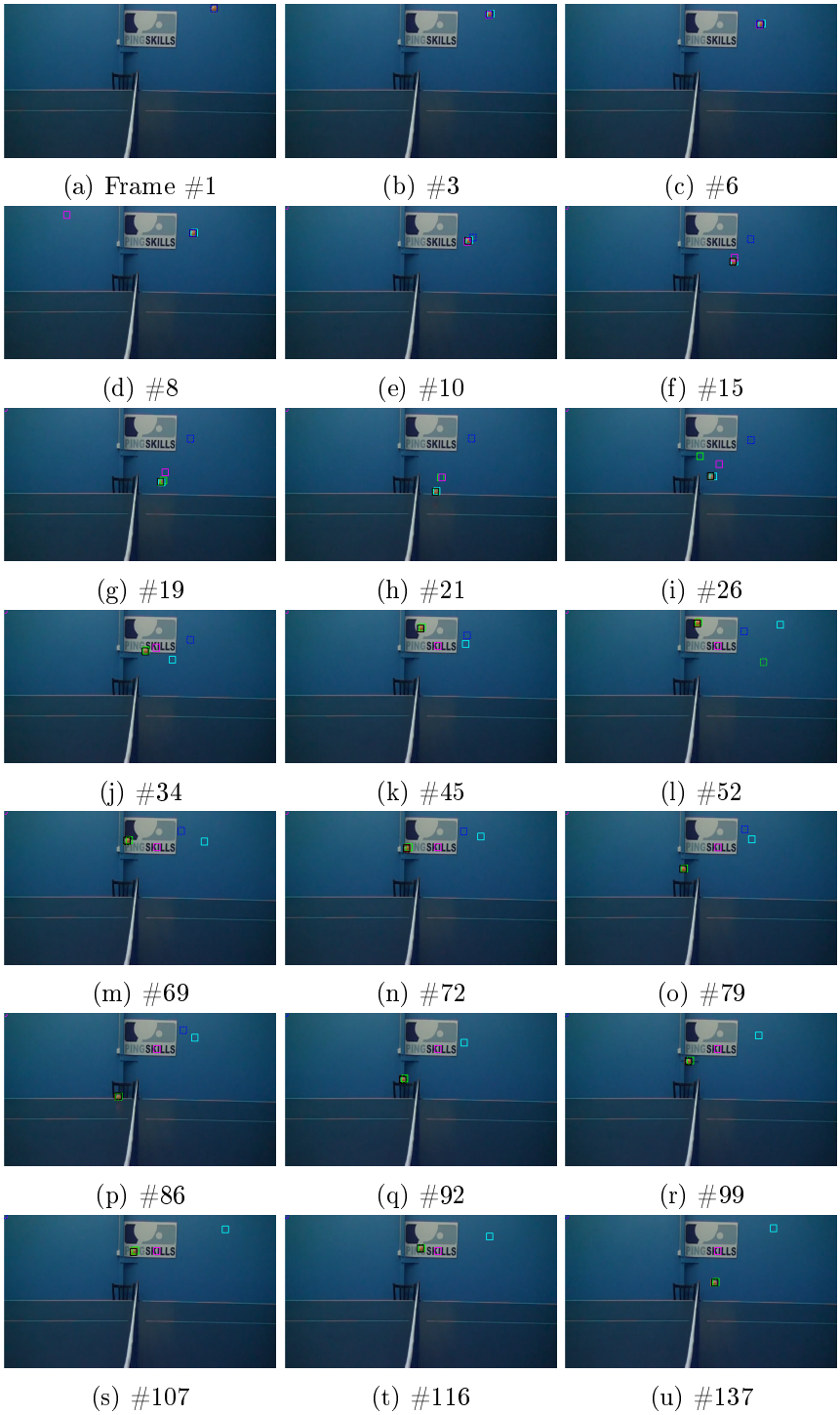
(j) #639 (k) #640 (l) #641

(m) #642 (n) #643 (o) #644

(p) #645 (q) #646 (r) #647

(s) #652 (t) #653 (u) #654

Figure D.4: Tracking results of the Bouncing1 sequence (Part 2). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #3 (c) #6

(d) #8 (e) #10 (f) #15

(g) #19 (h) #21 (i) #26

(j) #34 (k) #45 (l) #52

(m) #69 (n) #72 (o) #79

(p) #86 (q) #92 (r) #99

(s) #107 (t) #116 (u) #137

Figure D.5: Tracking results of the Table tennis sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

(a) Frame #17 (b) #22 (c) #24

(d) #26 (e) #27 (f) #29

(g) #30 (h) #31 (i) #32

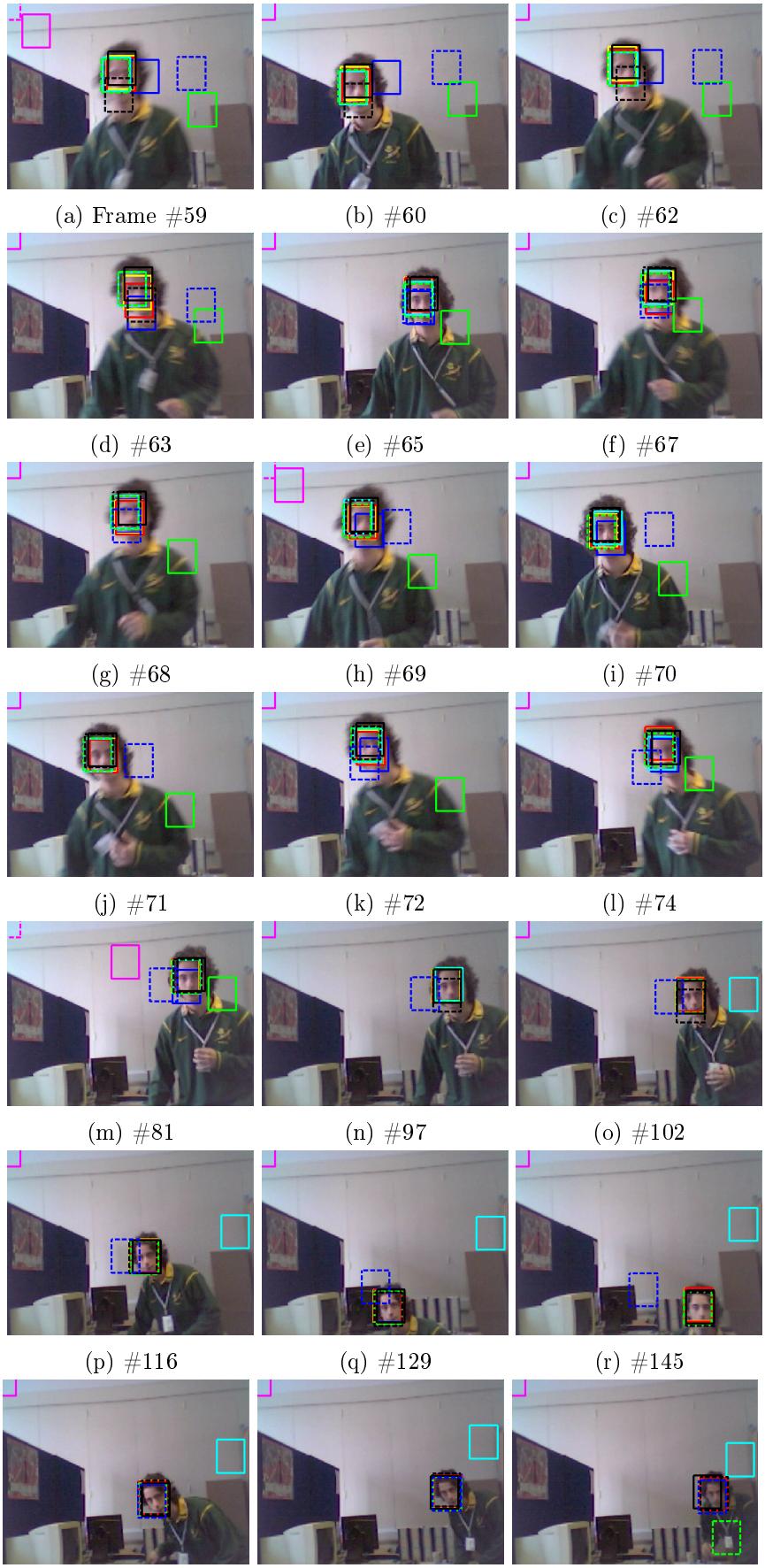
(j) #33 (k) #36 (l) #37

(m) #40 (n) #41 (o) #42

(p) #43 (q) #45 (r) #47

(s) #55 (t) #57 (u) #58

Figure D.6: Tracking results of the Emilio sequence (Part 1). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #59 (b) #60 (c) #62

(d) #63 (e) #65 (f) #67

(g) #68 (h) #69 (i) #70

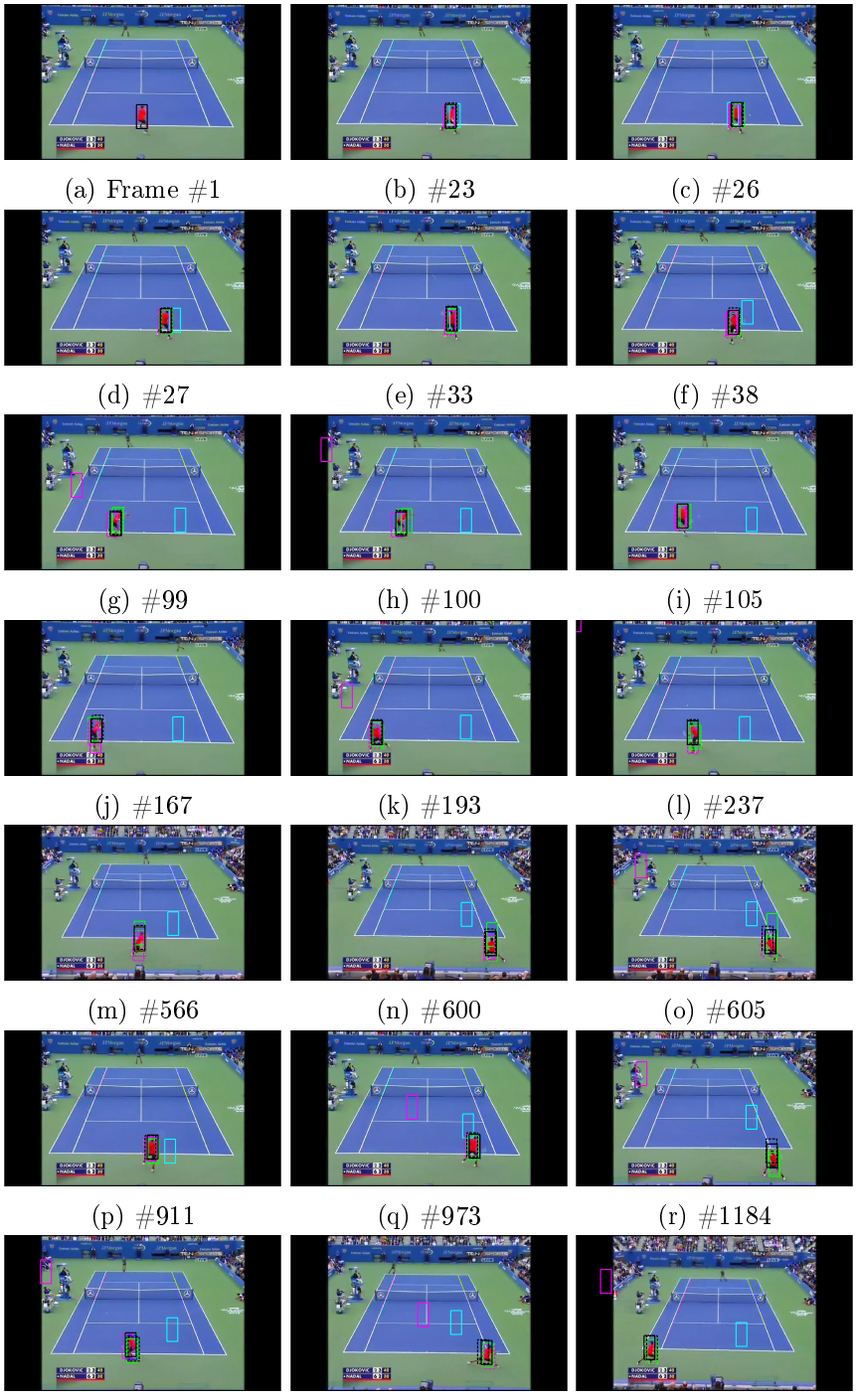
(j) #71 (k) #72 (l) #74

(m) #81 (n) #97 (o) #102

(p) #116 (q) #129 (r) #145

(s) #176 (t) #220 (u) #236

Figure D.7: Tracking results of the Emilio sequence (Part 2). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #23 (c) #26

(d) #27 (e) #33 (f) #38

(g) #99 (h) #100 (i) #105

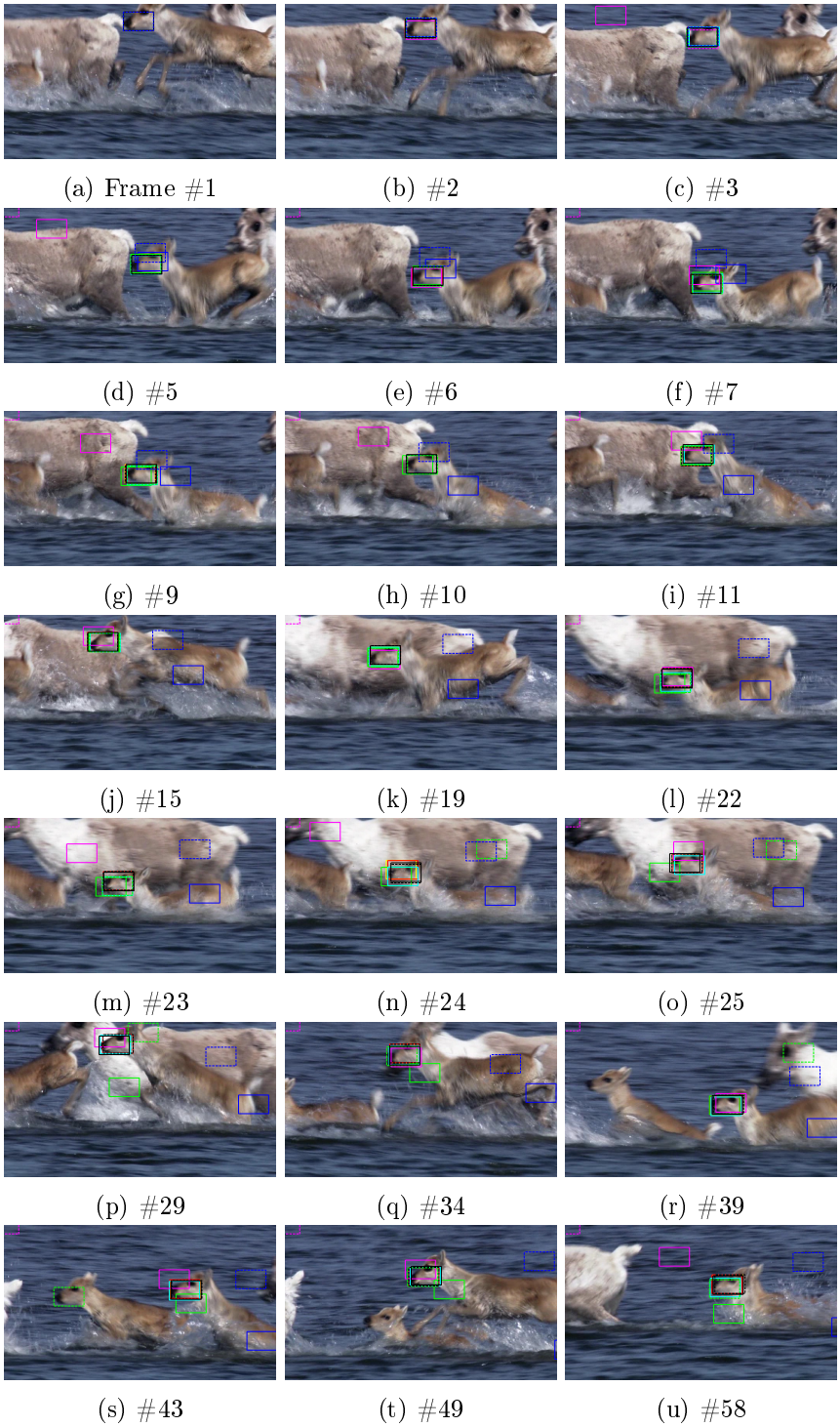
(j) #167 (k) #193 (l) #237

(m) #566 (n) #600 (o) #605

(p) #911 (q) #973 (r) #1184

(s) #1232 (t) #1517 (u) #1600

Figure D.8: Tracking results of the Tennis match sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #2 (c) #3

(d) #5 (e) #6 (f) #7

(g) #9 (h) #10 (i) #11

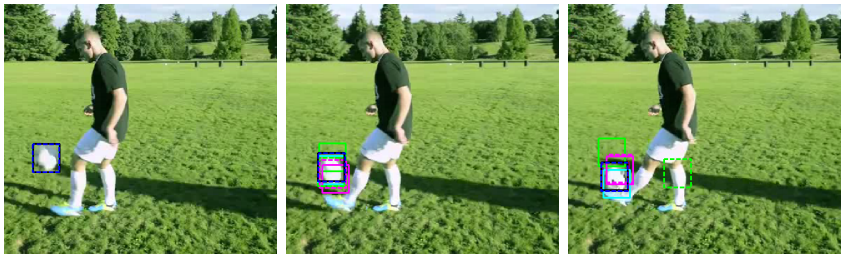
(j) #15 (k) #19 (l) #22

(m) #23 (n) #24 (o) #25

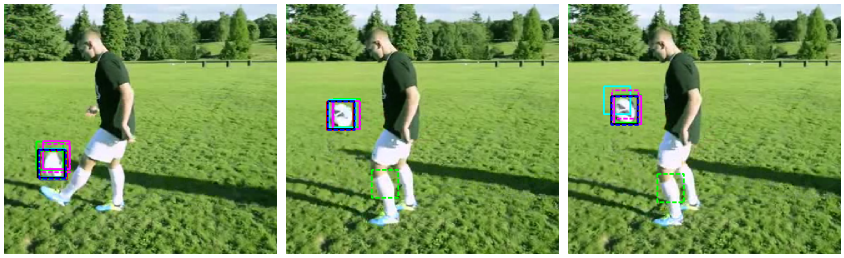
(p) #29 (q) #34 (r) #39

(s) #43 (t) #49 (u) #58

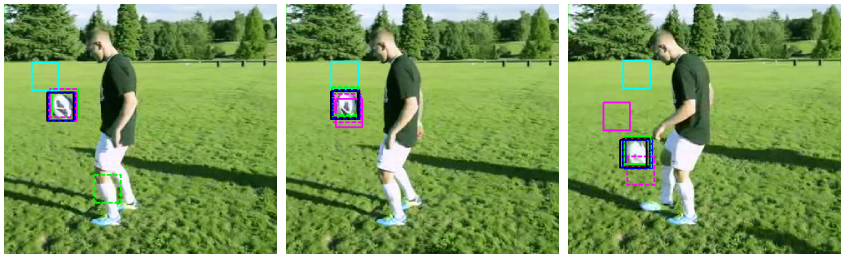
Figure D.9: Tracking results of the Animal sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



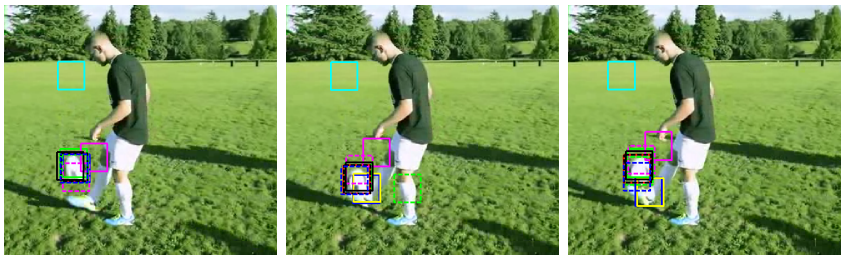
(a) Frame #1 (b) #2 (c) #3



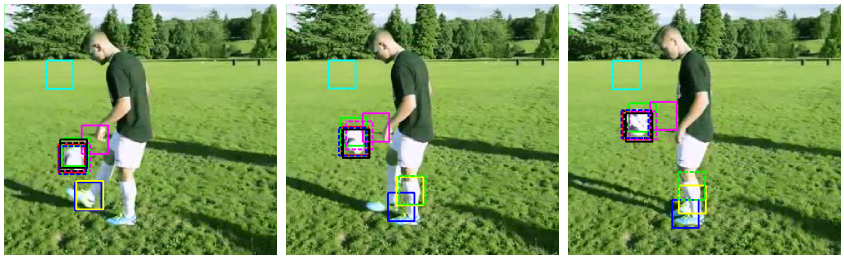
(d) #4 (e) #9 (f) #10



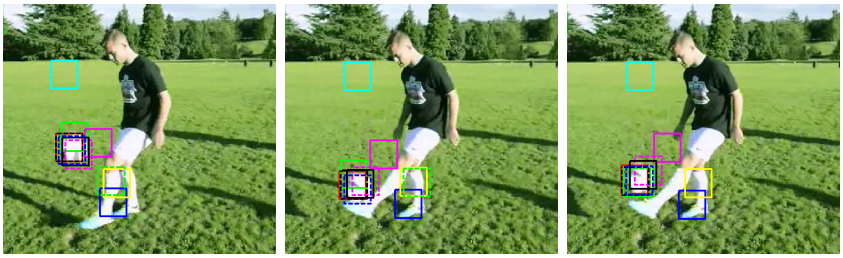
(g) #11 (h) #13 (i) #20



(j) #21 (k) #22 (l) #23

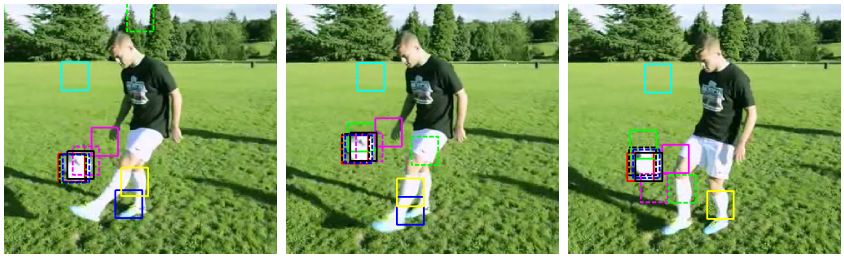


(m) #24 (n) #26 (o) #29

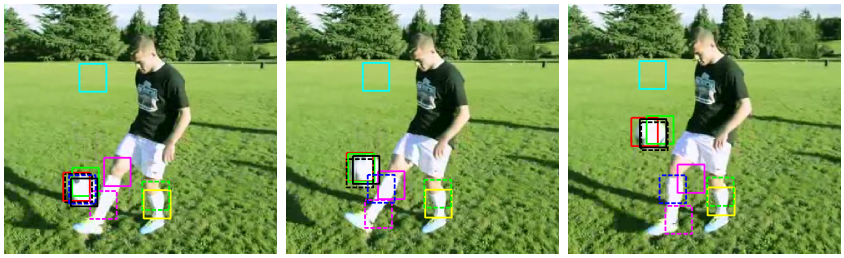


(p) #36 (q) #39 (r) #40

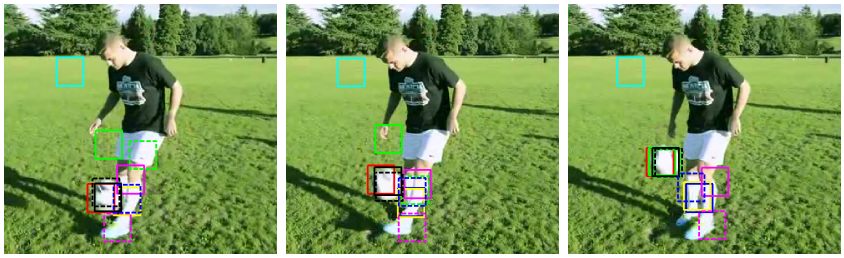
Figure D.10: Tracking results of the Football sequence (Part 1). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



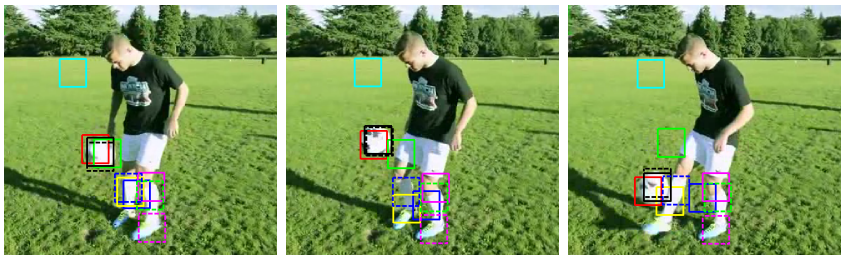
(a) Frame #41 (b) #43 (c) #52



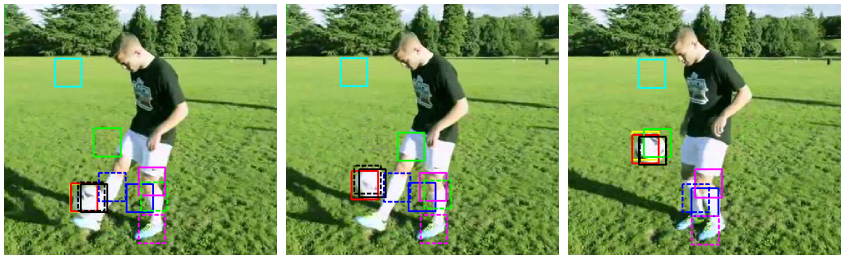
(d) #54 (e) #57 (f) #60



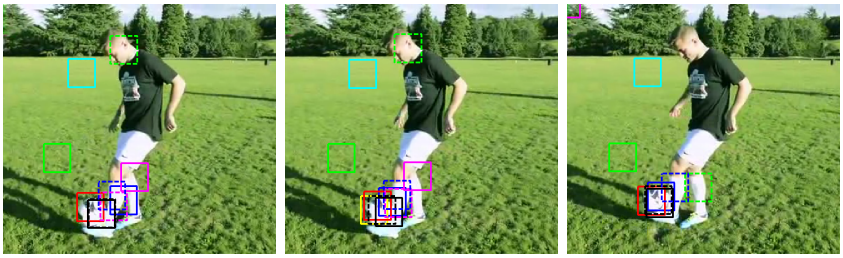
(g) #73 (h) #76 (i) #78



(j) #79 (k) #85 (l) #90

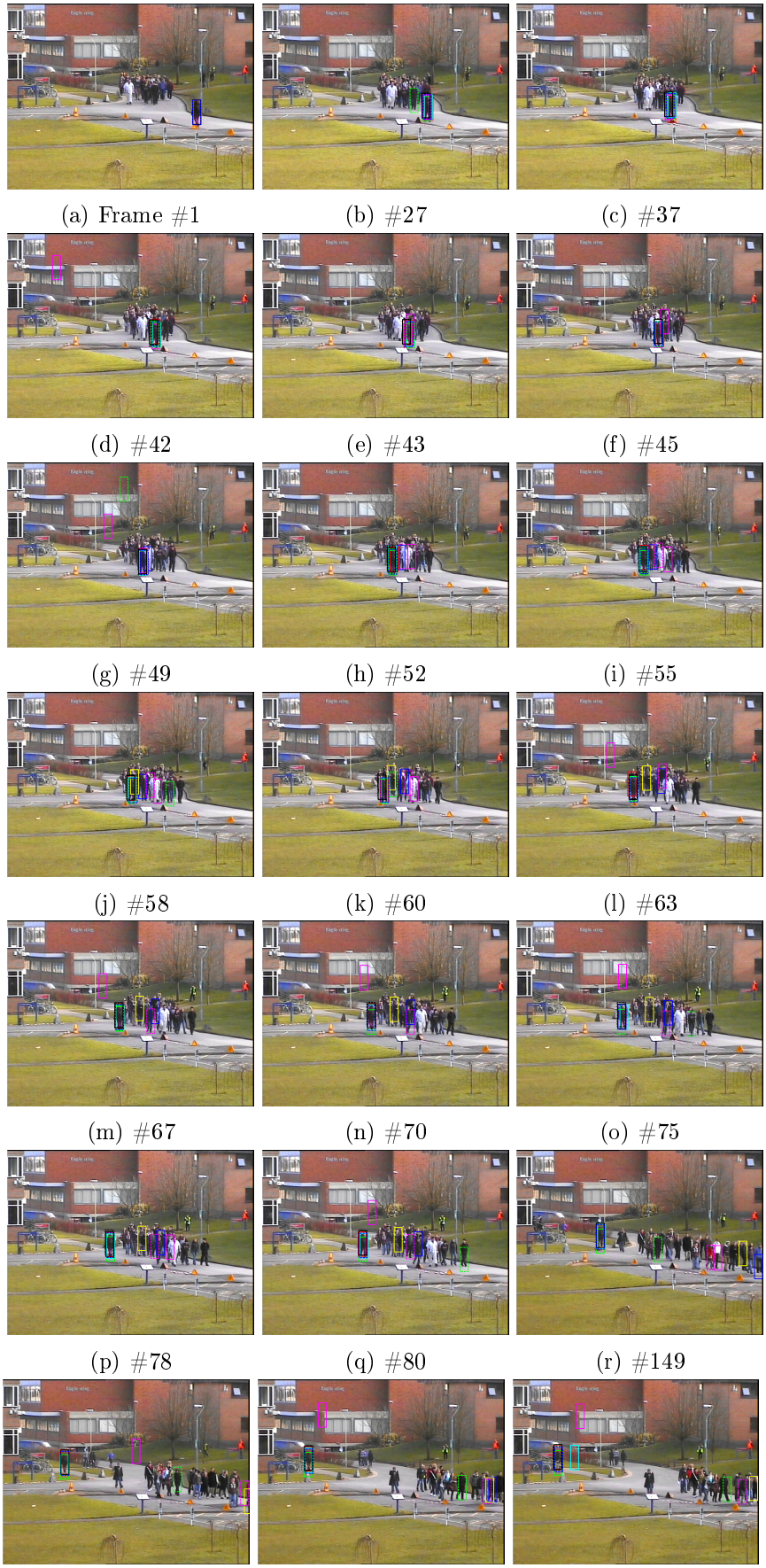


(m) #91 (n) #93 (o) #99



(p) #112 (q) #115 (r) #124

Figure D.11: Tracking results of the Football sequence (Part 2). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #27 (c) #37

(d) #42 (e) #43 (f) #45

(g) #49 (h) #52 (i) #55

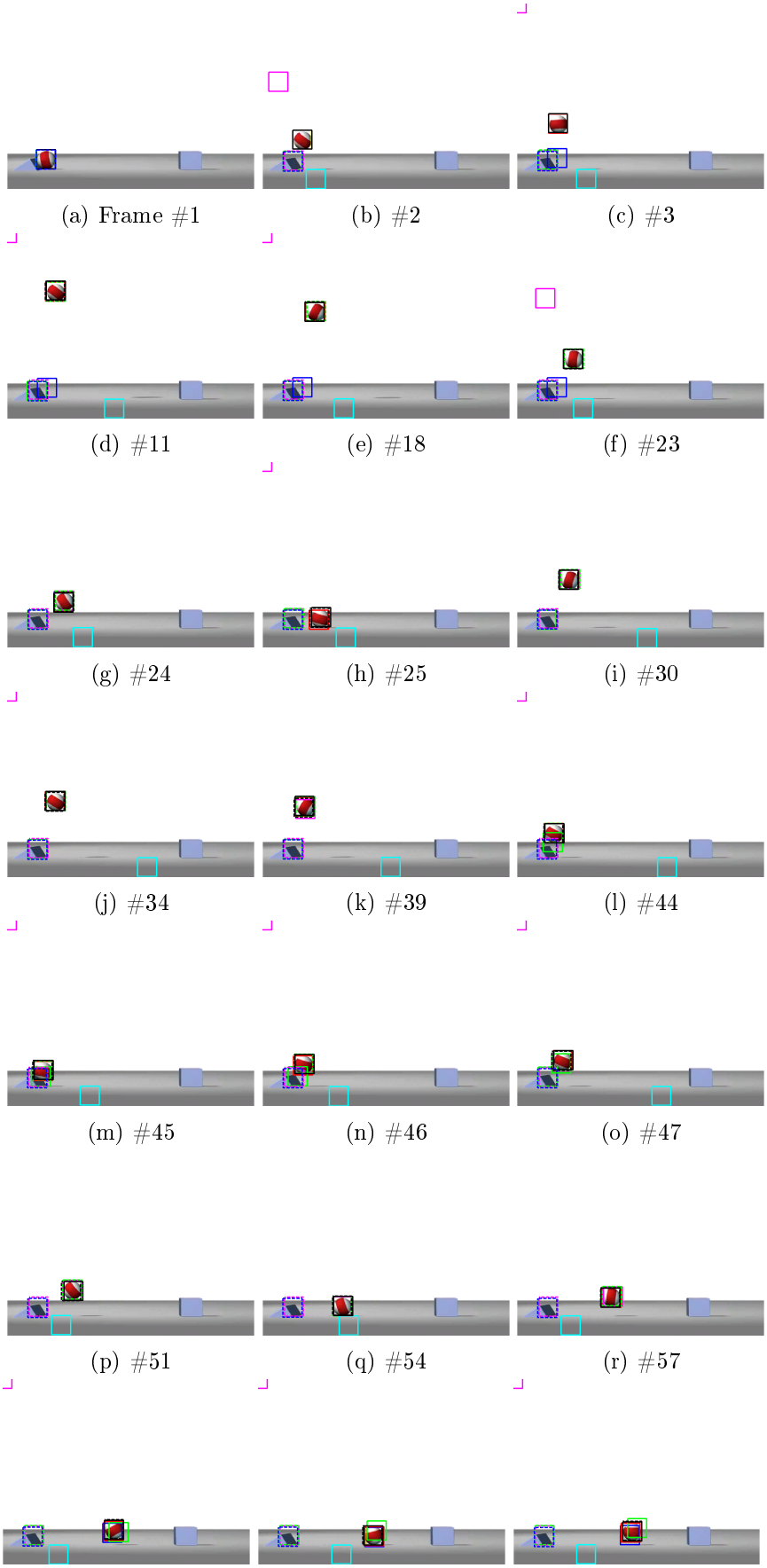
(j) #58 (k) #60 (l) #63

(m) #67 (n) #70 (o) #75

(p) #78 (q) #80 (r) #149

(s) #186 (t) #208 (u) #216

Figure D.12: Tracking results of the PETS09 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #2 (c) #3

(d) #11 (e) #18 (f) #23

(g) #24 (h) #25 (i) #30

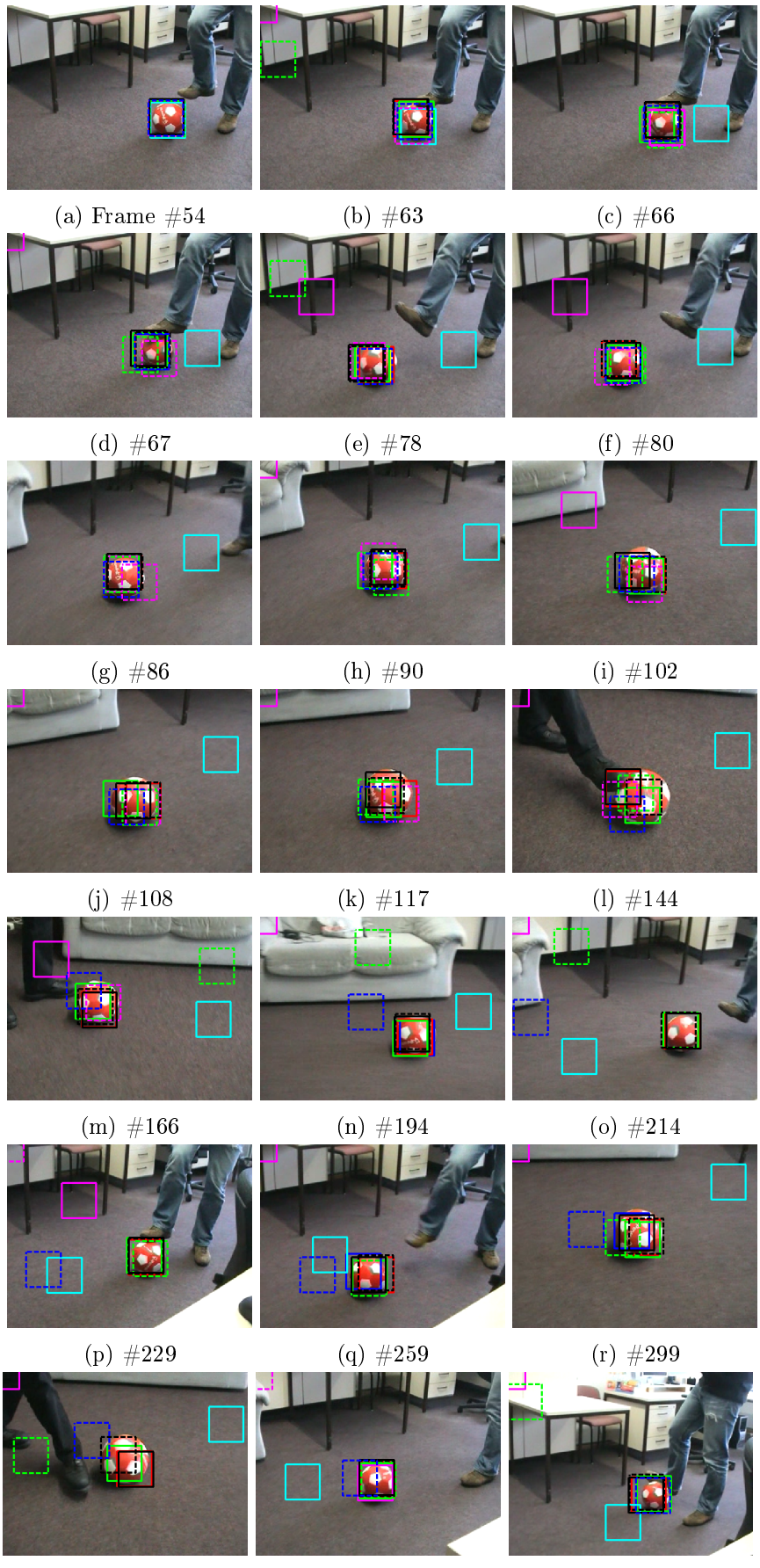
(j) #34 (k) #39 (l) #44

(m) #45 (n) #46 (o) #47

(p) #51 (q) #54 (r) #57

(s) #61 (t) #62 (u) #63

Figure D.13: Tracking results of the Bouncing2 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #54 (b) #63 (c) #66

(d) #67 (e) #78 (f) #80

(g) #86 (h) #90 (i) #102

(j) #108 (k) #117 (l) #144

(m) #166 (n) #194 (o) #214

(p) #229 (q) #259 (r) #299

(s) #340 (t) #379 (u) #581

Figure D.14: Tracking results of the Rolling Ball sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

(a) Frame #188 (b) #205 (c) #212

(d) #338 (e) #367 (f) #440

(g) #456 (h) #760 (i) #874

(j) #895 (k) #945 (l) #1048

(m) #1087 (n) #1233 (o) #1420

(p) #1451 (q) #1583 (r) #1659

(s) #1689 (t) #1714 (u) #1715

Figure D.15: Tracking results of the Doll sequence (Part 1). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

(a) Frame #1886 (b) #1915 (c) #1946

(d) #1961 (e) #1991 (f) #2237

(g) #2347 (h) #2398 (i) #2399

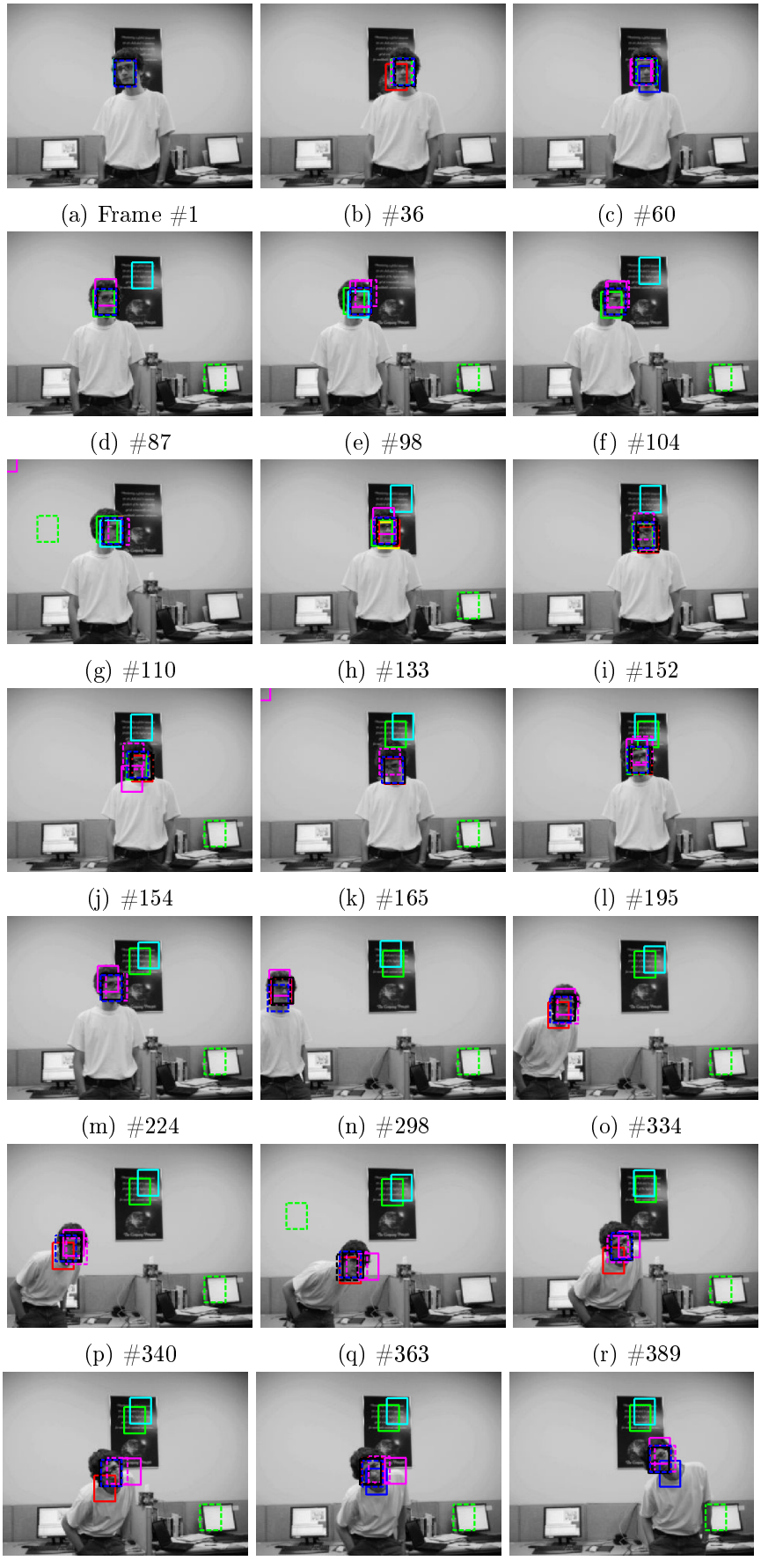
(j) #2753 (k) #2778 (l) #2785

(m) #2794 (n) #2897 (o) #2911

(p) #2925 (q) #2936 (r) #2996

(s) #3183 (t) #3307 (u) #3328

Figure D.16: Tracking results of the Doll sequence (Part 2). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #36 (c) #60

(d) #87 (e) #98 (f) #104

(g) #110 (h) #133 (i) #152

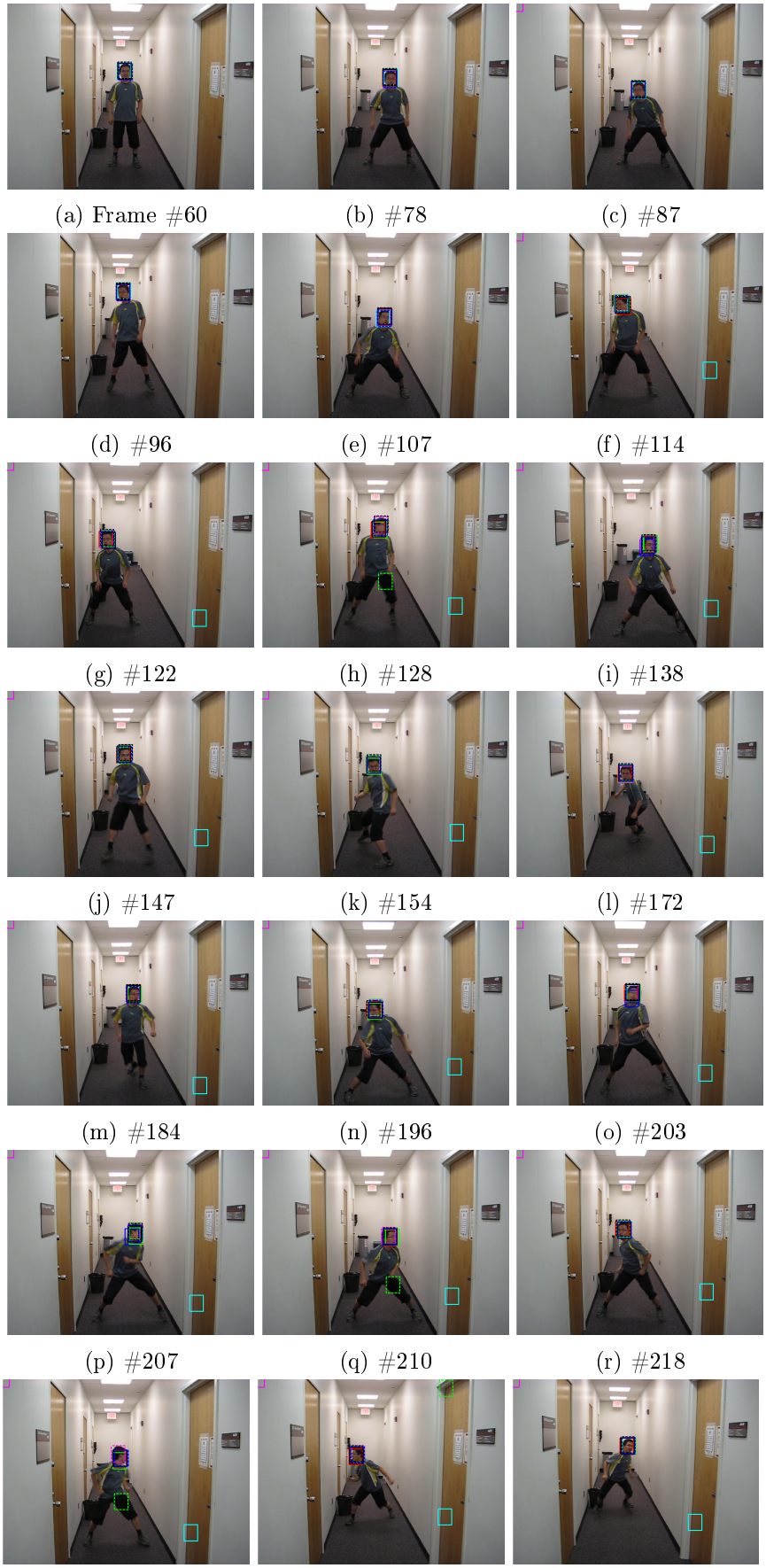
(j) #154 (k) #165 (l) #195

(m) #224 (n) #298 (o) #334

(p) #340 (q) #363 (r) #389

(s) #392 (t) #402 (u) #425

Figure D.17: Tracking results of the David2 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #60 (b) #78 (c) #87

(d) #96 (e) #107 (f) #114

(g) #122 (h) #128 (i) #138

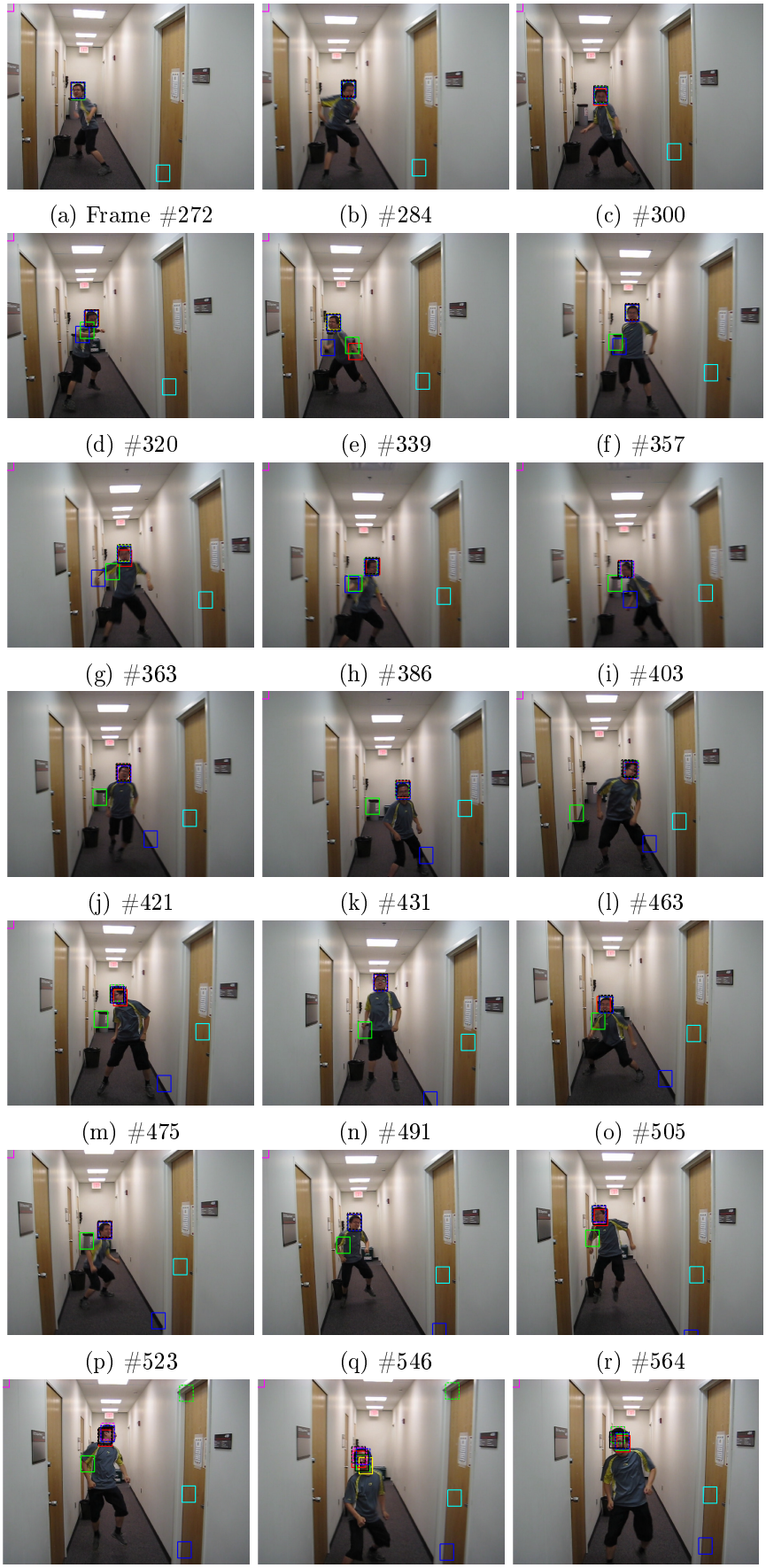
(j) #147 (k) #154 (l) #172

(m) #184 (n) #196 (o) #203

(p) #207 (q) #210 (r) #218

(s) #225 (t) #244 (u) #256

Figure D.18: Tracking results of the Boy sequence (Part 1). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #272 (b) #284 (c) #300

(d) #320 (e) #339 (f) #357

(g) #363 (h) #386 (i) #403

(j) #421 (k) #431 (l) #463

(m) #475 (n) #491 (o) #505

(p) #523 (q) #546 (r) #564

(s) #581 (t) #593 (u) #599

Figure D.19: Tracking results of the Boy sequence (Part 2). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

(a) Frame #1 (b) #6 (c) #9

(d) #65 (e) #70 (f) #78

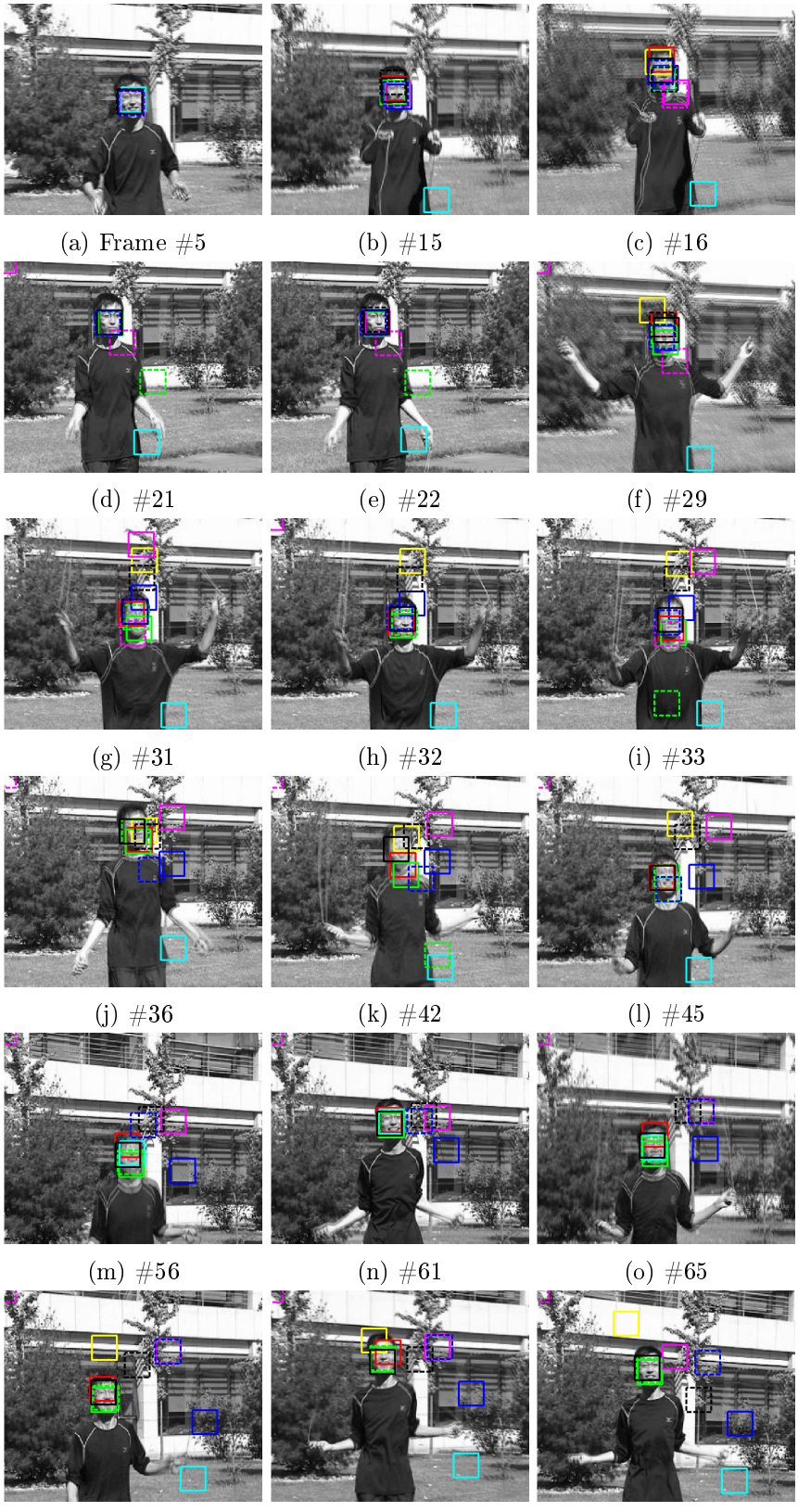
(g) #79 (h) #87 (i) #93

(j) #104 (k) #113 (l) #118

(m) #162 (n) #198 (o) #279

(p) #285 (q) #288 (r) #302

Figure D.20: Tracking results of the Jogging sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #5 (b) #15 (c) #16

(d) #21 (e) #22 (f) #29

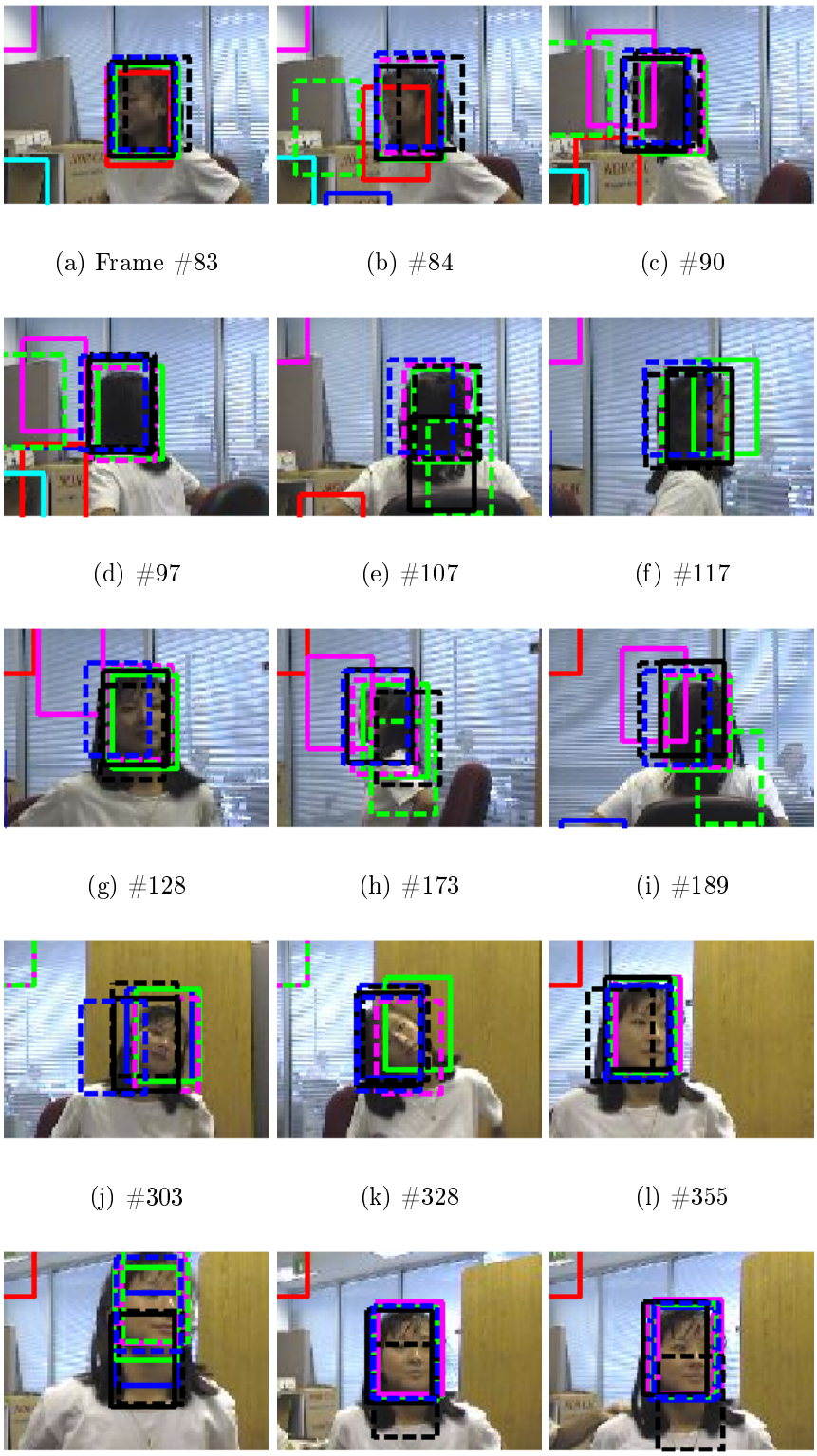
(g) #31 (h) #32 (i) #33

(j) #36 (k) #42 (l) #45

(m) #56 (n) #61 (o) #65

(p) #116 (q) #181 (r) #230

Figure D.21: Tracking results of the Jumping sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #83 (b) #84 (c) #90

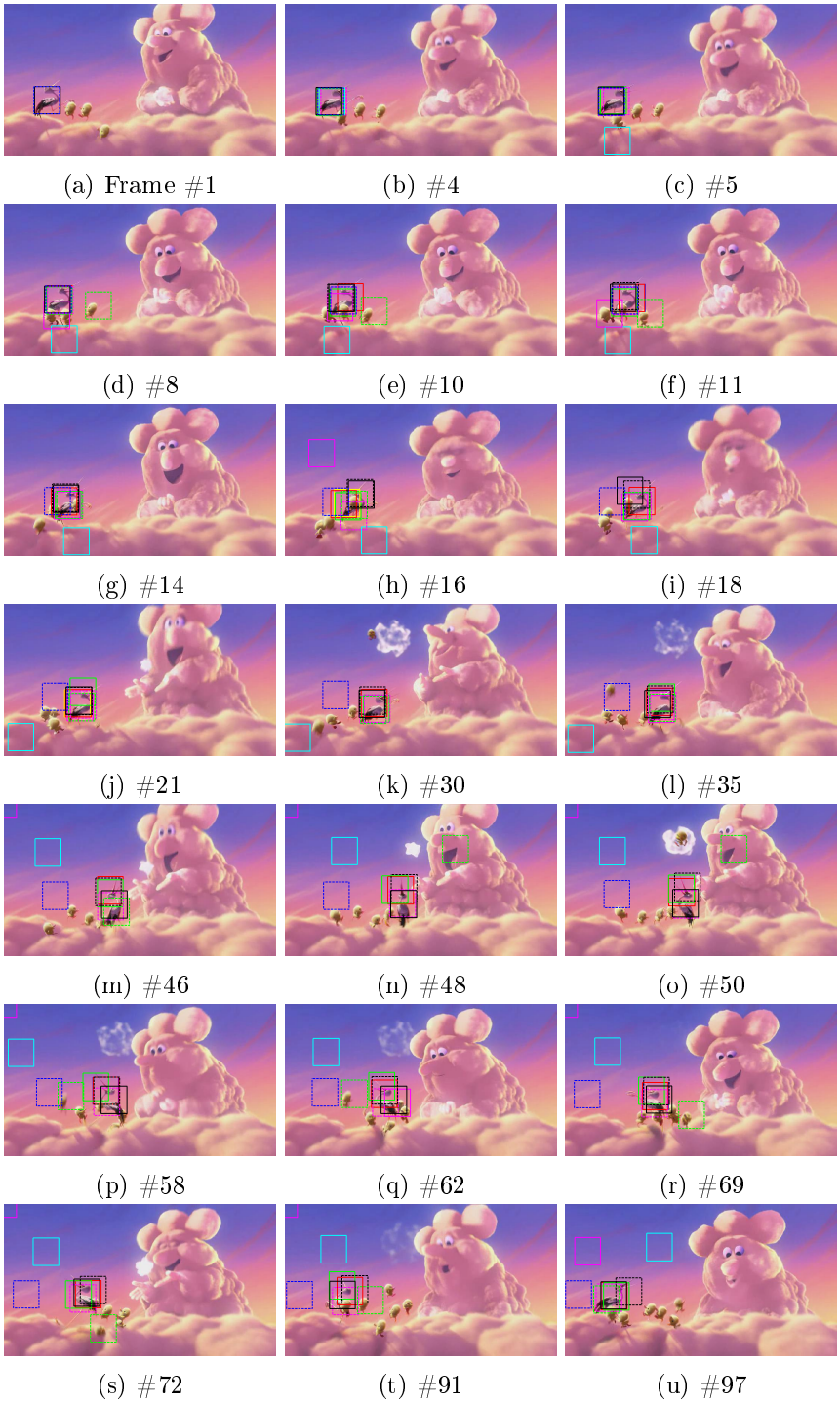
(d) #97 (e) #107 (f) #117

(g) #128 (h) #173 (i) #189

(j) #303 (k) #328 (l) #355

(m) #386 (n) #404 (o) #490

Figure D.22: Tracking results of the Girl sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #4 (c) #5

(d) #8 (e) #10 (f) #11

(g) #14 (h) #16 (i) #18

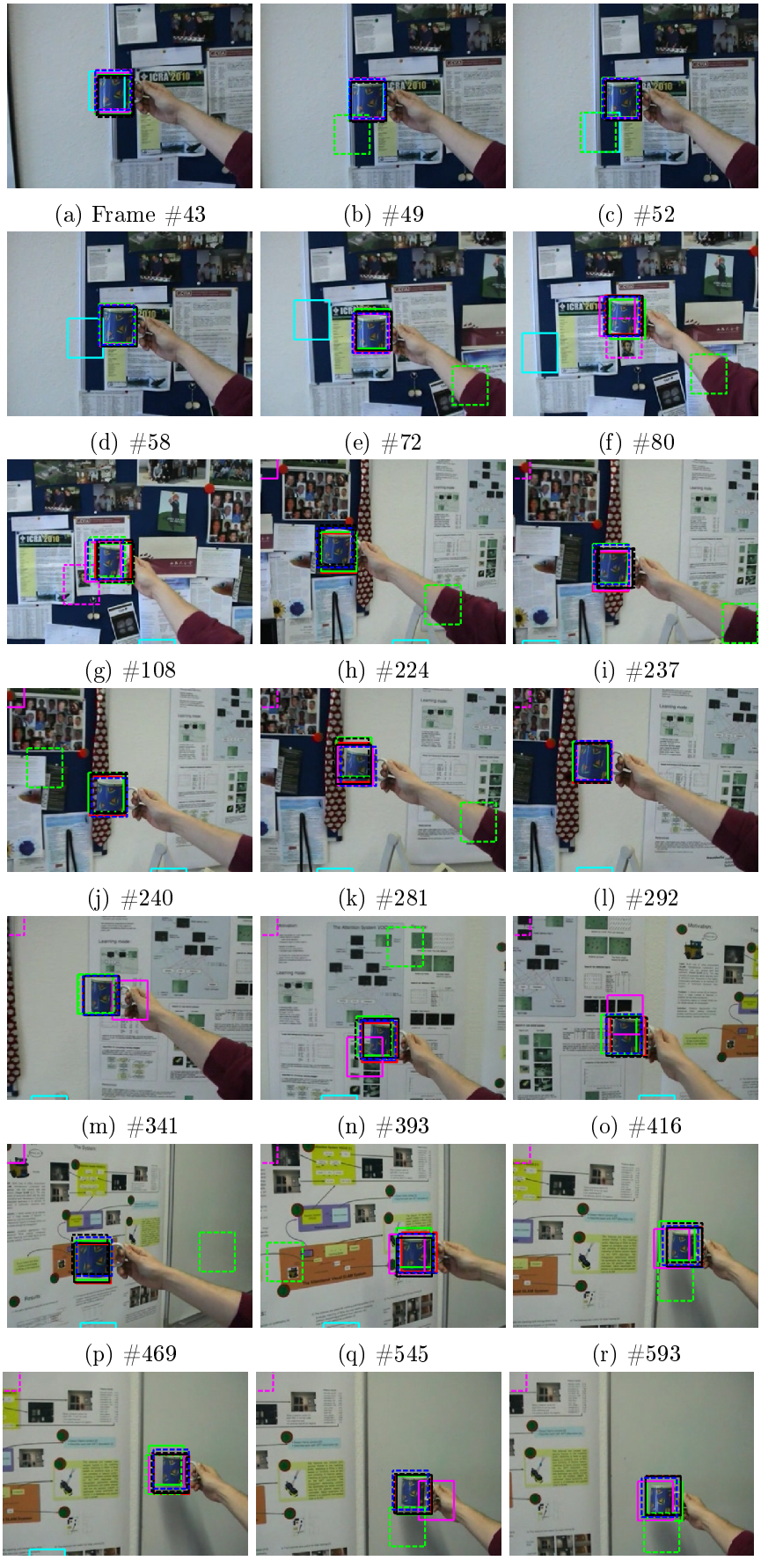
(j) #21 (k) #30 (l) #35

(m) #46 (n) #48 (o) #50

(p) #58 (q) #62 (r) #69

(s) #72 (t) #91 (u) #97

Figure D.23: Tracking results of the Bird2 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #43 (b) #49 (c) #52

(d) #58 (e) #72 (f) #80

(g) #108 (h) #224 (i) #237

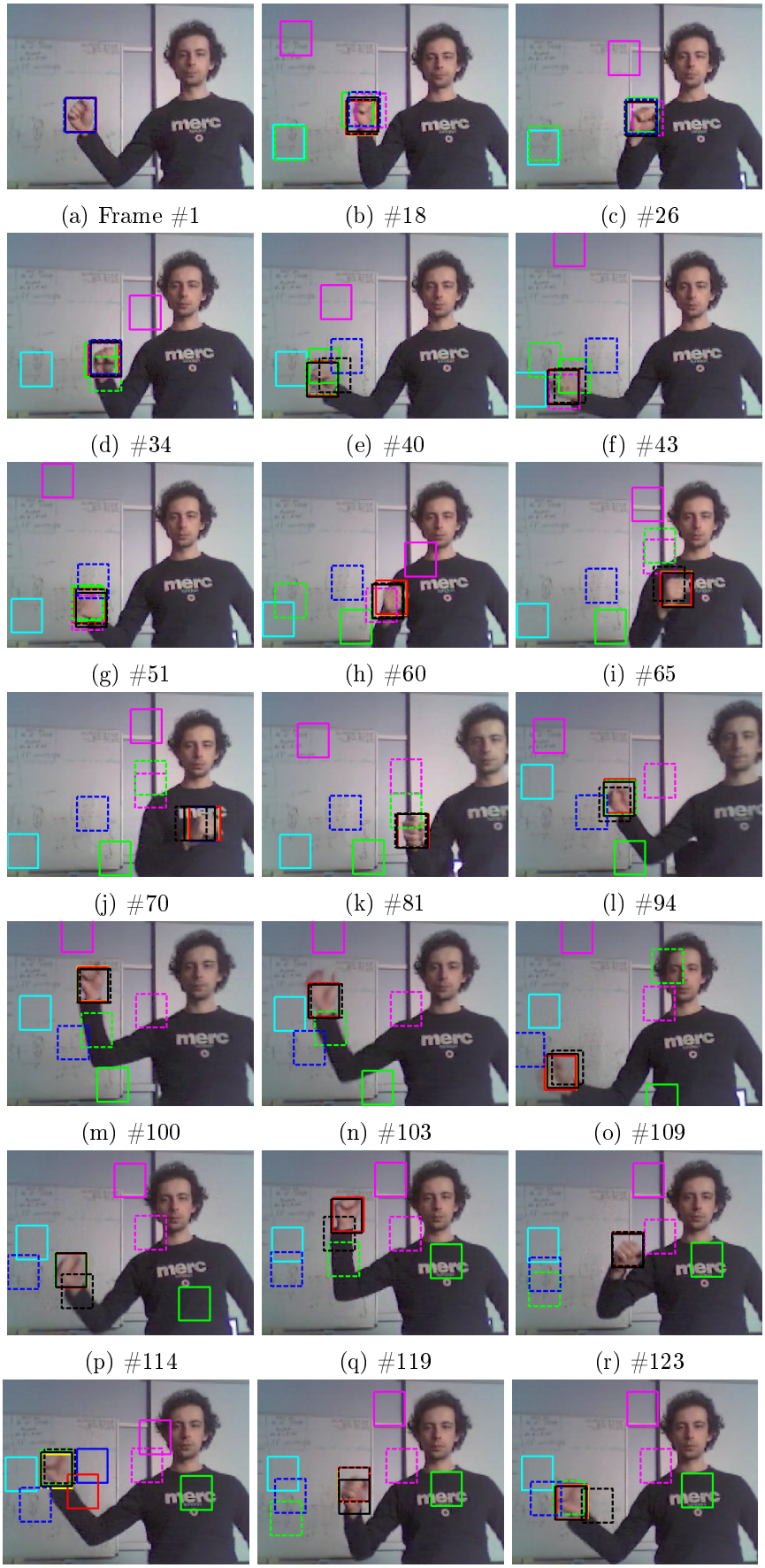
(j) #240 (k) #281 (l) #292

(m) #341 (n) #393 (o) #416

(p) #469 (q) #545 (r) #593

(s) #599 (t) #612 (u) #628

Figure D.24: Tracking results of the Cup sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #18 (c) #26

(d) #34 (e) #40 (f) #43

(g) #51 (h) #60 (i) #65

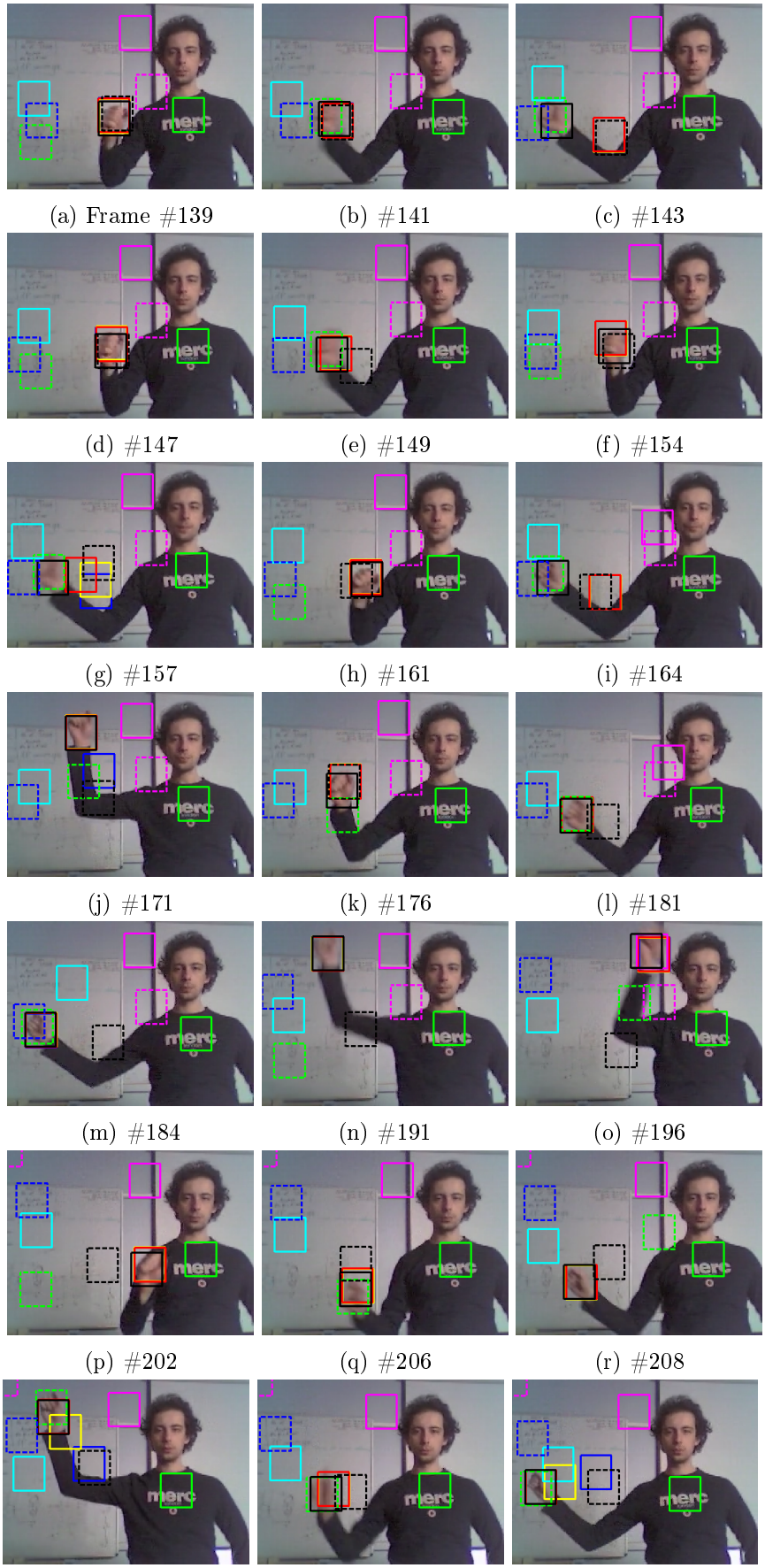
(j) #70 (k) #81 (l) #94

(m) #100 (n) #103 (o) #109

(p) #114 (q) #119 (r) #123

(s) #127 (t) #132 (u) #135

Figure D.25: Tracking results of the Hand sequence (Part 1). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #139 (b) #141 (c) #143

(d) #147 (e) #149 (f) #154

(g) #157 (h) #161 (i) #164

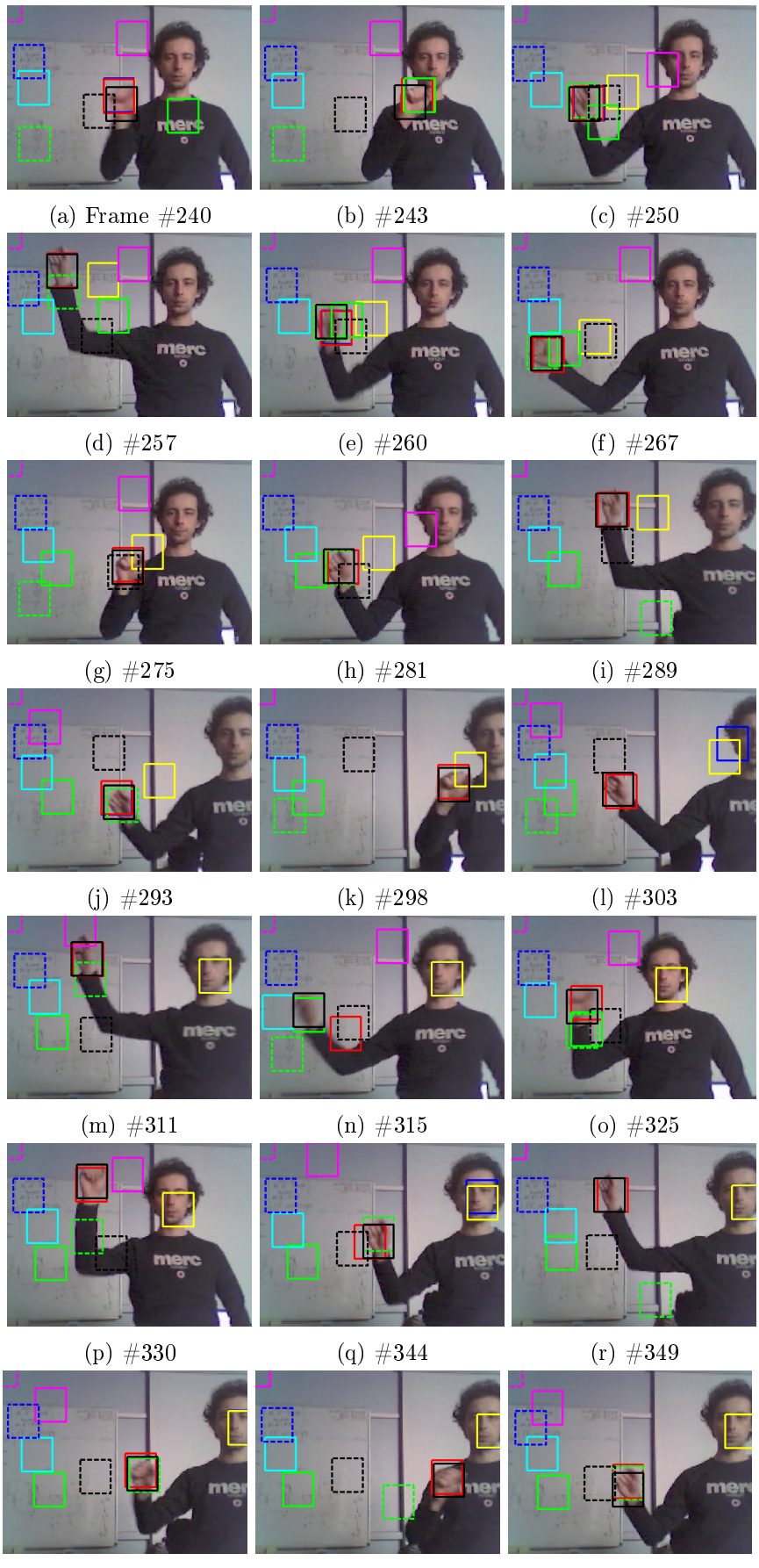
(j) #171 (k) #176 (l) #181

(m) #184 (n) #191 (o) #196

(p) #202 (q) #206 (r) #208

(s) #221 (t) #225 (u) #231

Figure D.26: Tracking results of the Hand sequence (Part 2). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #240 (b) #243 (c) #250

(d) #257 (e) #260 (f) #267

(g) #275 (h) #281 (i) #289

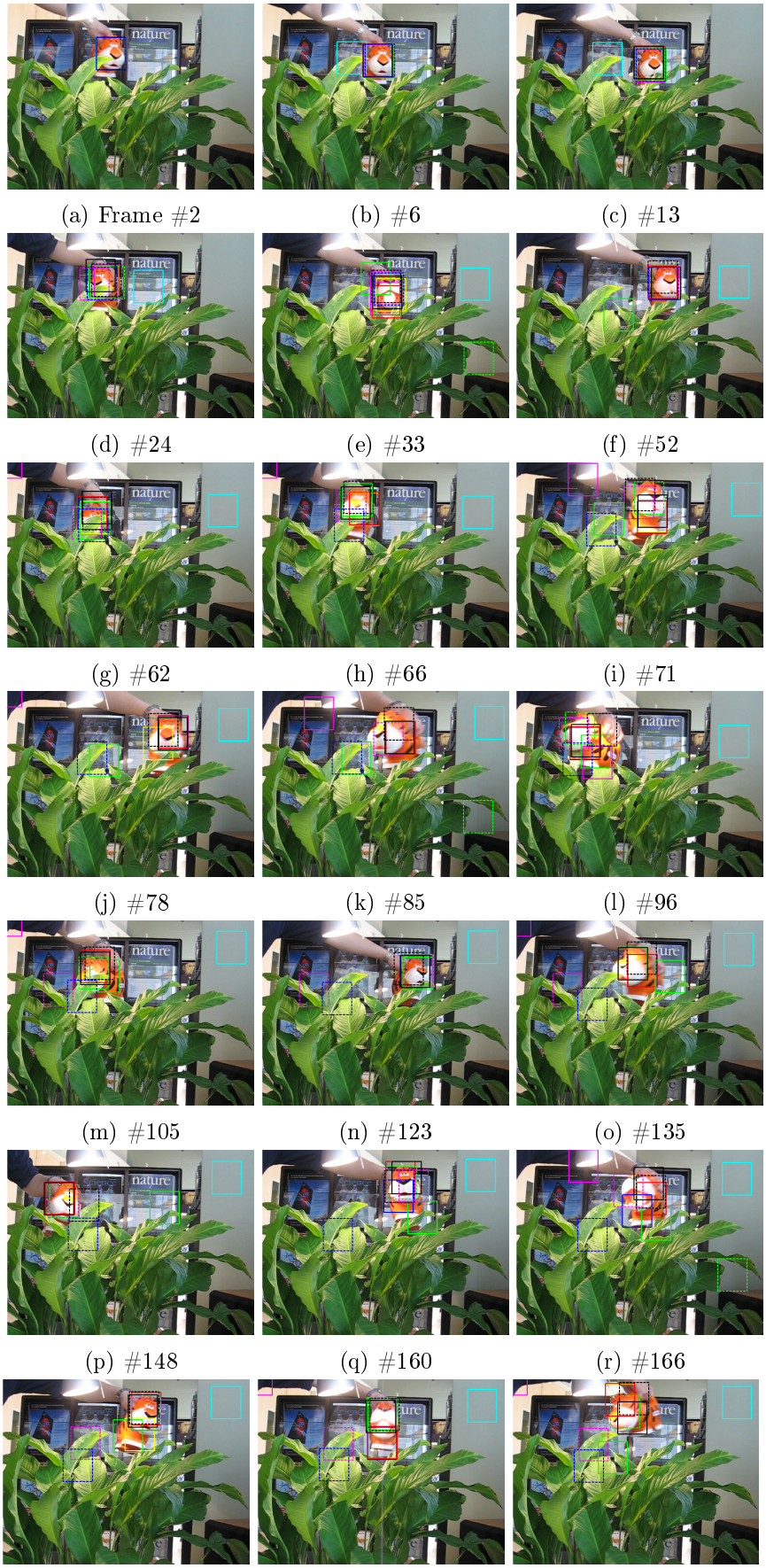
(j) #293 (k) #298 (l) #303

(m) #311 (n) #315 (o) #325

(p) #330 (q) #344 (r) #349

(s) #354 (t) #357 (u) #364

Figure D.27: Tracking results of the Hand sequence (Part 3). FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #2 (b) #6 (c) #13

(d) #24 (e) #33 (f) #52

(g) #62 (h) #66 (i) #71

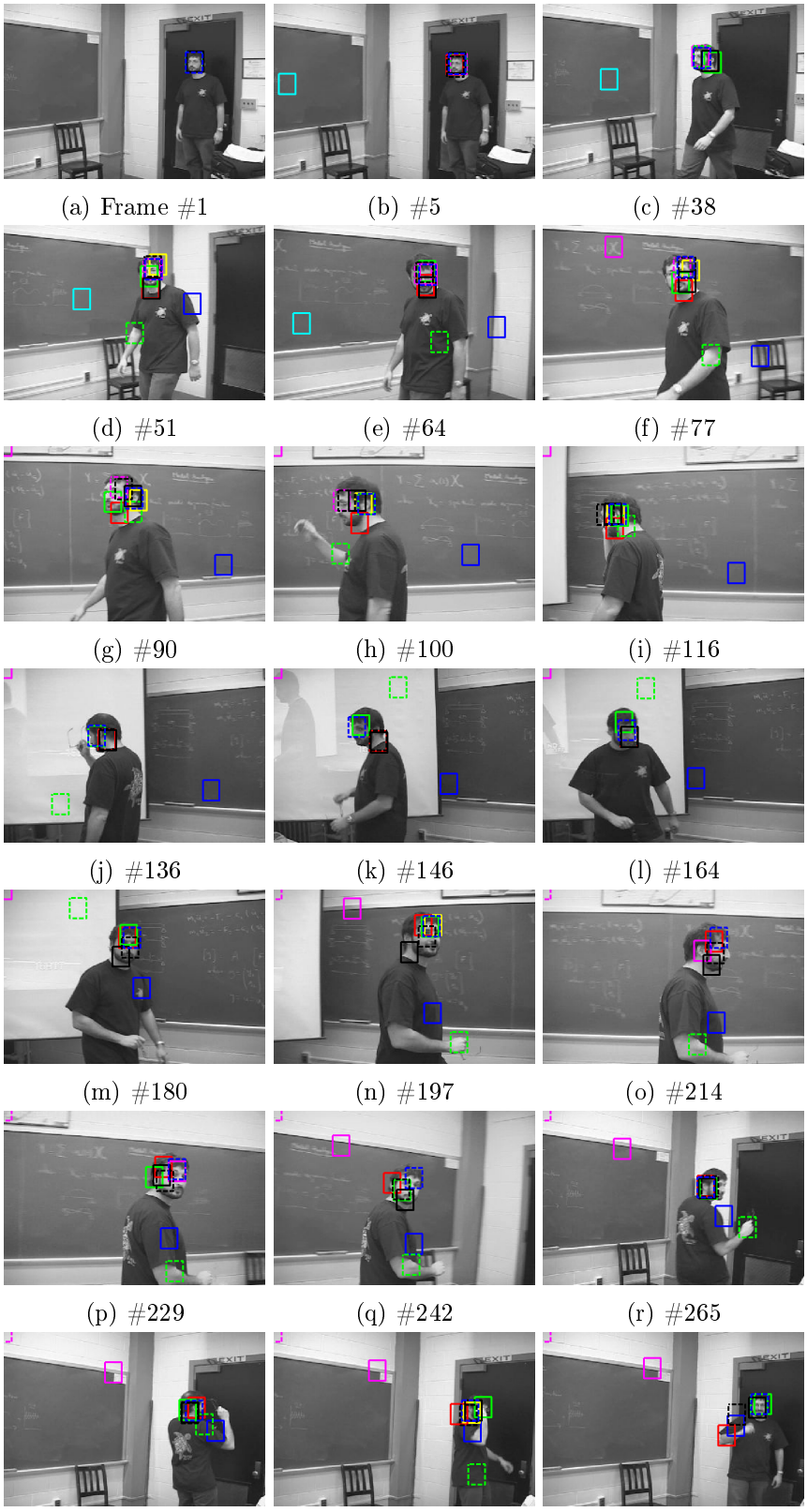
(j) #78 (k) #85 (l) #96

(m) #105 (n) #123 (o) #135

(p) #148 (q) #160 (r) #166

(s) #222 (t) #285 (u) #329

Figure D.28: Tracking results of the Tiger1 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).



(a) Frame #1 (b) #5 (c) #38

(d) #51 (e) #64 (f) #77

(g) #90 (h) #100 (i) #116

(j) #136 (k) #146 (l) #164

(m) #180 (n) #197 (o) #214

(p) #229 (q) #242 (r) #265

(s) #283 (t) #298 (u) #314

Figure D.29: Tracking results of the Freeman1 sequence. FMCMC-MM (black), MCMC-SA ((dashed)black), MCMC (blue), FMCMC-C (yellow), FMCMC-S (red), FragTrack (green), IVT (cyan), SB (magenta), TT ((dashed) green), OAB ((dashed) magenta), VTD ((dashed) blue).

Bibliography

A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 798–805, 2006.

A. Ali and J.K. Aggarwal. Segmentation and recognition of continuous human activity. In *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, pages 28–35, 2001.

M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, Feb 2002. ISSN 1053-587X.

S. Avidan. Ensemble tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):261 –271, feb. 2007. ISSN 0162-8828.

AVSS2007. Advanced video and signal based surveillance 2007 (avss2007 dataset). URL ftp://motinas.elec.qmul.ac.uk/.

B. Babenko, Ming-Hsuan Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1619–1632, 2011. ISSN 0162-8828.

S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004. ISSN 0920-5691.

H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *Computer Vision in ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33832-1.

D. Berleant and G.T. Anderson. Decision-making under severe uncertainty for autonomous mobile robots. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 2360–2365, Oct 2007.

G. Bianchi and I.Tinnirello. Kalman filter estimation of the number of competing terminals in an ieee 802.11 network. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 844–852 vol.2, March 2003.

S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 232–237, Jun 1998.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM. ISBN 1-58113-057-0.

J. Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.

M.D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1515–1522, Sept 2009.

M.Z. Brown, D. Burschka, and G.D. Hager. Advances in computational stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(8):993–1008, Aug 2003. ISSN 0162-8828.

K. Cannons. A review of visual tracking. Technical report, Department of Computer Science and Engineering - York University, 2008.

J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *Radar, Sonar and Navigation, IEE Proceedings* -, 146(1):2–7, 1999. ISSN 1350-2395.

R. Cipolla and A. Pentland. *Computer Vision for Human-Machine Interaction*. Cambridge University Press, 1998.

R.T. Collins. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–234–40 vol.2, June 2003.

R.T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10): 1631 –1643, oct. 2005. ISSN 0162-8828.

D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. pages 142–149, 2000.

D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564 – 577, may 2003. ISSN 0162-8828.

J. Czyz, B. Ristic, and B. Macq. A particle filter for joint detection and tracking of multiple objects in color video sequences. In *Information Fusion, 2005 8th International Conference on*, volume 1, pages 7 pp.–, July 2005.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

C. Dancey and J. Reidy. *Statistics Without Maths for Psychology*. Prentice Hall, 2011.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society, Series B*, 39(1):1–38, 1977.

K. G. Derpanis, K. G. Derpanis, S. Richard, P. Wildes, J. K. Tsotsos, C. Members, M. Spetsakis, and H. R. Wilson. Characterizing image motion, 2006.

A. P. Dhawan. *Medical Image Analysis*. John Wiley & Sons, 2011.

T. B. Dinh and G. Medioni. Co-training framework of generative and discriminative trackers with partial occlusion handling. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 642–649, Jan 2011.

T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1177–1184, June 2011.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000. ISSN 0960-3174.

M. Everingham, L. Gool, C.K.I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. ISSN 0920-5691.

G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, pages 363–370, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-40601-8.

W. Förstner and E. Gülch. A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features, 1987.

N.J. Gordon, D.J. Salmond, and A. F M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993. ISSN 0956-375X.

H. Grabner and H. Bischof. On-line boosting and vision. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 260–267, 2006.

H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 234–247, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88681-5.

H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1285–1292, June 2010.

F. Gustafsson, F. Gunnarsson, Niclas Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *Signal Processing, IEEE Transactions on*, 50(2):425–437, Feb 2002. ISSN 1053-587X.

G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(10):1025–1039, Oct 1998. ISSN 0162-8828.

Z. Han, Q. Ye, and J. Jiao. Combined feature evaluation for adaptive visual object tracking. *Computer Vision and Image Understanding*, 115(1):69 – 80, 2011. ISSN 1077-3142.

C. Harris and M. Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):pp. 97–109, 1970. ISSN 00063444.

W. He, T. Yamashita, H. Lu, and S. Lao. Surf tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1586–1592, Sept 2009.

B. K.P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17: 185 – 203, 1981. ISSN 0004-3702.

J. Huang, S.R. Kumar, M. Mitra, and W. Zhu. Spatial color indexing and applications. In *Computer Vision, 1998. Sixth International Conference on*, pages 602–607, Jan 1998.

C. Hue, J. P. Le Cadre, and P. Prez. Tracking multiple objects with particle filtering, 2000.

M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the 4th European Conference on Computer Vision-Volume I - Volume I*, ECCV '96, pages 343–356, London, UK, UK, 1996. Springer-Verlag. ISBN 3-540-61122-3.

M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Computer Vision, 1998. Sixth International Conference on*, pages 107–112, 1998.

J. K. and K.. M. Lee. Tracking by sampling and integrating multiple trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints):1, 2013. ISSN 0162-8828.

R. E Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, pages pp. 35–45, 1960.

T. Kanade, H. Kano, S. Kimura, A. Yoshida, and K. Oda. Development of a video-rate stereo machine. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 95–100 vol.3, Aug 1995.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

Z. Khan, T. Balch, and F. Dellaert. Mcmc - based particle filtering for tracking a variable number of interacting targets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1805 –1819, nov. 2005. ISSN 0162-8828.

J. Kim, Z. Lin, and I. S. Kweon. Rao-blackwellized particle filtering with gaussian mixture models for robust visual tracking. *Computer Vision and Image Understanding*, (0):–, 2014. ISSN 1077-3142.

Z. Kim. Real time object tracking based on dynamic feature grouping with background subtraction. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):pp. 1–25, 1996. ISSN 10618600.

D.A. Klein, D. Schulz, S. Frintrop, and A.B. Cremers. Adaptive real-time video-tracking for arbitrary objects. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 772–777, Oct 2010.

M. Kristan, J. Pers, S. Kovacic, and A. Leonardis. A local-motion-based probabilistic model for visual tracking. *Pattern Recognition*, 42(9), 2009. ISSN 0031-3203.

M. Kristan, S. Kovacic, A. Leonardis, and J. Pers. A two-stage dynamic model for visual tracking. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(6):1505–1520, 2010. ISSN 1083-4419.

C. Kuo, C. Huang, and R. Nevatia. Multi-target tracking by on-line learned discriminative appearance models. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 685 –692, june 2010.

J. Kwon and K. M. Lee. Tracking of abrupt motion using wang-landau monte carlo estimation. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision - ECCV 2008*, volume 5302 of *Lecture Notes in Computer Science*, pages 387–400. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88681-5.

J. Kwon and K. M. Lee. Visual tracking decomposition. In *in CVPR*, 2010.

J. Kwon and K. M. Lee. Highly nonrigid object tracking via patch-based dynamic appearance modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(10):2427–2441, October 2013. ISSN 0162-8828.

J. Kwon and K.M. Lee. Tracking by sampling trackers. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1195–1202, 2011.

J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled hybrids of generative and discriminative models. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 87–94, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0.

B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3): 259–289, May 2008.

X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel. A survey of appearance models in visual object tracking. *ACM Trans. Intell. Syst. Technol.*, 4(4):58:1–58:48, October 2013. ISSN 2157-6904.

J. Liu, P. Carr, R.T. Collins, and Y. Liu. Tracking sports players with context-conditioned motion models. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1830–1837, June 2013.

D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop*, pages 674–679, 1981.

J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 572–578 vol.1, 1999.

E. Maggio and A. Cavallaro. Hybrid particle filter and mean shift tracker with adaptive transition model. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 2, pages 221 – 224, 18-23, 2005a.

E. Maggio and A. Cavallaro. Multi-part target representation for color tracking. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 1, pages I–729–32, Sept 2005b.

E. Maggio, F. Smeraldi, and A. Cavallaro. Adaptive multi-feature tracking in a particle filtering framework. *IEEE Transactions on Circuit and Systems for Video technology*, Vol. X, No. X, 2007.

I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):810–815, 2004. ISSN 0162-8828.

J. McIntyre, A. Church, and F. Labrosse. Efficient image-based tracking of apparently changing moving targets. *In Proceedings of Towards Autonomous Robotic Systems*, 2009.

S.J. McKenna, Y. Raja, and S. Gong. Object tracking using adaptive colour mixture models. In Roland Chin and Ting-Chuen Pong, editors, *Computer Vision - ACCV'98*, volume 1351 of *Lecture Notes in Computer Science*, pages 615–622. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-63930-5.

A. Y. Ng and Mi. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, 2001.

K. Nickels and S. Hutchinson. Estimating uncertainty in ssd-based feature tracking. *Image and Vision Computing*, 20:47–58, 2002.

F. M. Noguer. *Multiple Cue Integration for Robust Tracking in Dynamic Environments: Application to Video Relighting.* PhD thesis, 2005.

K. Nummiaro, E. Koller-Meier, and L. V. Gool. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99 – 110, 2003. ISSN 0262-8856.

K. Okuma, O. Freitas A. Taleghani, N. D. Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *In ECCV*, pages 28–39, 2004.

N.C. Oza. Online bagging and boosting. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 3, pages 2340 – 2345 Vol. 3, oct. 2005.

N.C. Oza. *Online Ensemble Learning.* PhD thesis, University of California, Berkeley, 2011.

P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *In Proc. ECCV*, pages 661–675, 2002.

D. Ramanan and D.A. Forsyth. Using temporal coherence to build models of animals. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 338–345 vol.1, Oct 2003.

R. A. Robb. *Biomedical Imaging, Visualization and Analysis*. Wiley-Blackwell, 2000.

D. A. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. 2008.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Computer Vision in ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33832-1.

D. Rowe, J. Gonzalez, M. Pedersoli, and J. Villanueva. On tracking inside groups. *Machine Vision and Applications*, 21:113–127, 2010. ISSN 0932-8092. 10.1007/s00138-009-0194-y.

D. J. Rumsey. *Statistics For Dummies*. John Wiley & Sons, 2011.

A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400, Sept 2009.

J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 723–730, June 2010.

R. E. Schapire. The boosting approach to machine learning: An overview, 2002.

A. Sears and J. A. Jacko. *The Human-computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. Lawrence Erlbaum Associates Inc, 2007.

D. Serby, E.K. Meier, and L. Van Gool. Probabilistic object tracking using multiple features. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 184–187 Vol.2, 2004.

S.M. Shahed Nejhum, J. Ho, and M.H. Yang. Visual tracking with histograms and articulating blocks. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

C. Shan, Y. Wei, T. Tan, and F. Ojardias. Real time hand tracking by combining particle filtering and mean shift. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 669–674, May 2004.

C. Shen, A. V. Den Hengel, and A. Dick. Probabilistic multiple cue integration for particle filter based tracking. In *in Proceedings of the VIIth Digital Image Computing : Techniques and Applications*, pages 309–408, 2003.

J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

K. Smith, D. Gatica-Perez, and J.-M. Odobez. Using particles to track varying numbers of interacting people. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 962 – 969 vol. 1, june 2005.

M. Spengler and B. Schiele. Towards robust multi-cue integration for visual tracking. In *Machine Vision and Applications*, page 2003, 2001.

S. Stalder, H. Grabner, and L. V. Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1409–1416, Sept 2009.

I. KolingerovÃą T. VomÃącka. Early warning system for air traffic control using kinetic delaunay triangulation. In Leonard Bolc, Ryszard Tadeusiewicz, LeszekJ. Chmielewski, and Konrad Wojciechowski, editors, *Computer Vision and Graphics*, volume 6375 of *Lecture Notes in Computer Science*, pages 350–356. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15906-0.

F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct 2007.

R. Taylor. Interpretation of the correlation coefficient: A basic review. *Journal of Diagnostic Medical Sonography*, 6(1):35–39, 1990. URL http://jdm.sagepub.com/content/6/1/35.abstract.

S. Thrun. Particle filters in robotics. In *in Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI*, 2002.

J. Triesch and C. V. D. Malsburg. Democratic integration: Self-organized integration of adaptive cues. 2001.

C.J. Veenman, M. J T Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(1):54–72, Jan 2001. ISSN 0162-8828.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511 – I–518 vol.1, 2001.

D. Walther, D.R. Edgington, and C. Koch. Detection and tracking of objects in underwater video. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–544–I–549 Vol.1, June 2004.

H. Wang and D. Suter. Efficient visual tracking by probabilistic fusion of multiple cues. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pages 892–895, 2006.

D. Wei and P. Justus. A probabilistic approach to integrating multiple cues in visual tracking. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 225–238, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88685-3.

G. Welch and G. Bishop. An introduction to the kalman filter, 1995.

M. Werlberger, W. Trobin, T.Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *Proceedings of the British Machine Vision Conference (BMVC)*, London, UK, September 2009.

T. Woodley, B. Stenger, and R. Cipolla. Tracking using online feature selection and a local generative model. 2007.

Y. Wu and T. S. Huang. Robust visual tracking by integrating multiple cues based on co-inference learning. *International Journal of Computer Vision*, 58:55–71, 2004.

Y. Wu, J. Lim, and M. Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

F. Yang, H. Lu, and M. Yang. Robust superpixel tracking. *Image Processing, IEEE Transactions on*, 23(4):1639–1651, April 2014. ISSN 1057-7149.

H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song. Recent advances and trends in visual tracking: A review. *Neurocomput.*, 74(18):3823–3831, November 2011. ISSN 0925-2312.

M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(7):1195–1209, July 2009. ISSN 0162-8828.

A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1531–1536, Nov 2004. ISSN 0162-8828. doi: 10.1109/TPAMI.2004.96.

A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey, 2006.

Q. Yu, T.B. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision Ãćâĺňâ ĂIJ ECCV 2008*, volume 5303 of *Lecture Notes in Computer Science*, pages 678–691. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88685-3.

X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu. Sequential particle swarm optimization for visual tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

Q. Zhao and H. Tao. Object tracking using color correlogram. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 263–270, Oct 2005.

Q. Zhao and H. Tao. Motion observability analysis of the simplified color correlogram for visual tracking. In Yasushi Yagi, SingBing Kang, InSo Kweon, and Hongbin Zha, editors, *Computer Vision-ACCV 2007*, volume 4843 of *Lecture Notes in Computer Science*, pages 345–354. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-76385-7.

H. Zhou, Y. Yuan, and C. Shi. Object tracking using {SIFT} features and mean shift. *Computer Vision and Image Understanding*, 113(3):345 – 352, 2009. ISSN 1077-3142. Special Issue on Video Analysis.