

Faculty of Computer Science and Information Technology

Multi-Agent Approach for Occlusion Handling in Crowded Surveillance Video

Lee Beng Yong

Doctor of Philosophy
2018

Multi-Agent Approach for Occlusion Handling In Crowded Surveillance Video

Lee Beng Yong

A thesis submitted

In fulfilment of the requirements for the degree of Doctor of Philosophy

(Computer Science)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2018

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citation, which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted in the candidature of any other degree.

_____

Name of the student: Lee Beng Yong

Date: 7 May 2018

DEDICATION

To my wife, Liew, Leehung, my daughter, Joanne Lee and my parents for their unconditional love and support.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my supervisor Professor Dr Wang Yin Chai and my co-supervisors Dr Cheah Wai Siang and Associate Professor Dr Noor Alamshah Bolhassan. It has been a great pleasure to work with them and to learn from their experience. Their patience and support guided me through my PhD study. Also, I would like to thank my colleagues and friends in the Faculty of Computer Science and Information Technology with whom I had the pleasure to work and discuss.

ABSTRACT

The task of object tracking is the key to the effective use of more advanced technologies such as event recognition in the surveillance video. To recognise an event, people and objects must be tracked. Tracking also enhances the performance of tasks such as crowd analysis or human identification. Among all existing object tracking methods, Continuously Adaptive Mean-shift (CAMShift) is one of the most popular methods adopted by many researchers. However, to use CAMShift to track an object, one should also verify that the chosen target representation is a sufficient discriminator for the application domain to ensure CAMShift can efficiently work in its application. This thesis investigates improvements to the performance of object tracking algorithms through improved target representation selection using a novel multi competitive agents system that consists of a manager agent, an evaluator agent and multiple CAMShift tracker agents. The interaction between the manager agent and the CAMShift tracker agents in the proposed is performed via the Call-For-Proposal protocol. Through the interaction, the CAMShift tracker agents in the system can compete among themselves to select an optimum target representation. The proposed algorithm is tested on lab simulation videos and shows significant improvement in tracking. Kalman filter is then added to the CAMShift tracker agents to solve full occlusion problem. When a tracked object is fully occluded, it is not visible in the video scenes. CAMShift tracker agent will not be able to retrieve any fully occluded object representation model from the scene, and therefore Kalman filter is used to predict the location of the tracked object based on spatial motion characteristic of the tracked object in previous frames. The proposed system is then tested on two famous benchmark surveillance video dataset, namely the PETS 2009 and CAVIAR. The results

show that the proposed system can track objects in crowded scenes even when a full occlusion occurs during the tracking process.

Keywords: Continuously Adaptive Mean-shift, Kalman Filter, Multi-Agents System Occlusion, Crowded Surveillance Video, Object Tracking

Kaedah Multi Ejen untuk Pengendalian Halangan bagi Penjejakan Objek dalam Video Pengawasan yang Sesak

ABSTRAK

Tugas penjejakan objek adalah kunci kepada keberkesanan penggunaan teknologi yang lebih maju seperti pengecaman peristiwa di video pengawasan. Untuk pengecaman peristiwa, manusia dan objek perlu dikesan. Penjejakan objek juga dapat meningkatkan prestasi tugas-tugas seperti menganalisa kelakuan manusia atau pengenalan manusia. Di antara semua kaedah penjejakan objek yang sedia ada, kaedah Peralihan Purata Berpenyesuaian Berterusan (CAMShift) adalah salah satu kaedah yang paling popular digunakan oleh kebanyakan penyelidik. Walau bagaimanapun, untuk menggunakan CAMShift untuk menjejak objek, seseorang itu perlu juga mengesahkan bahawa perwakilan sasaran yang dipilih adalah cukup diskriminan untuk domain aplikasi bagi memastikan CAMShift boleh bekerja dengan berkesan dalam aplikasi mereka. Tesis ini mengkaji cara penambahbaikan untuk prestasi algoritma penjejakan objek melalui pertambahbaikan dalam pemilihan perwakilan sasaran. Satu sistem multi ejen berkompetitif novel yang terdiri daripada ejen pengurus, ejen penilai dan ejen-ejen pengesan CAMShift yang mampu untuk bersaing dengan satu sama lain dalam usaha untuk pilih sasaran perwakilan optimum telah dicadangkan. Interaksi antara ejen-ejen dalam sistem yang dicadangkan adalah dilaksanakan melalui protokol Memanggil Untuk Cadangan (CFP). Algoritma yang dicadangkan ini telah diuji dengan menggunakan video simulasi makmal dan ianya menunjukkan peningkatan prestasi pengesanan yang ketara berbanding dengan sistem asas. Kemudian, penapis Kalman telah ditambah ke dalam ejen CAMShift tracker untuk menyelesaikan masalah halangan penuh. Apabila objek yang

dikesan adalah dihalang sepenuhnya, ia tidak boleh dilihat dalam babak video. ejen CAMShift tracker tidak akan dapat apa-apa model perwakilan objek dari babak video dan oleh itu penapis Kalman telah digunakan untuk meramalkan lokasi objek dikesan berdasarkan ciri-ciri gerakan spatial objek dikesan dalam bingkai sebelumnya. Sistem yang dicadangkan ini kemudian diuji dengan menggunakan dua set data pengawasan video penanda aras yang terkemuka, iaitu PETS 2009 dan CAVIAR. Keputusan menunjukkan bahawa sistem yang dicadangkan boleh mengesan objek dalam adegan sesak walaupun halangan penuh berlaku semasa proses pengesanan.

Kata kunci: *Peralihan Purata Berpenyesuaian Berterusan, Penapis Kalman, Sistem Multi Agen, Halangan, Video Pengawasan yang Sesak, Penjejakan Objek*

TABLE OF CONTENTS

Page

DECLARATION i

DEDICATION ii

ACKNOWLEDGMENT iii

ABSTRACT iv

ABSTRAK vi

TABLE OF CONTENTS viii

LIST OF TABLES xii

LIST OF FIGURES xiv

CHAPTER 1: INTRODUCTION 1

1.1 Introduction 1

1.2 Background of Study 2

1.3 Research Questions 5

1.4 Aims and Objectives 5

1.5 Research Scope 6

1.6 Outlines of Thesis 6

CHAPTER 2: LITERATURE REVIEW 9

2.1 Introduction 9

2.2 Object Tracking 9

2.3 Kernel-Based Object Tracking 12

2.3.1 Continuously Adaptive Mean-Shift (CAMShift) Tracking 13

2.3.2 CAMShift Limitation 19

2.4 Occlusion 19

2.5 Occlusion Handling 22

2.5.1 Depth Analysis 23

2.5.2 Optimal Camera Placement Method 24

2.5.3 Motion Prediction 25

2.5.4 Occlusion Handling Limitation 31

2.6 Multi Models Object Tracking 32

2.7 Multi-Agent System in Object Tracking 33

2.7.1 Previous Work of MAS in Object Tracking System 35

2.7.2 Role-Oriented Analysis and Design for Multi-Agent Programming (ROADMAP) 40

2.7.3 Negotiation Mechanism in Object Tracking 41

2.8 Surveillance Video Benchmark Dataset 43

2.9 Measurement of Detection Accuracy 48

2.10 Summary 51

CHAPTER 3: MULTI CAMSHIFT AGENT FRAMEWORK FOR OCCLUSION HANDLING IN OBJECT TRACKING 54

3.1 Introduction 54

3.2 Goal Model 55

3.3 Role Model 56

3.4 Organisation Model 59

3.5 Domain Model 59

3.6 Agent-Role Coupling 61

3.7 Interaction Model 63

3.8 Knowledge Model 67

3.9 Behaviour Model 68

3.10 Summary 74

CHAPTER 4: IMPLEMENTATION 75

4.1 Introduction 75

4.2 Negotiation Mechanism in Object Tracking 76

4.3 Implementation of JADE Based Tracking Manager Agent 76

4.4 Implementation of JADE Based Object Tracker Agent 83

4.5 Implementation of JADE Based Proposal Evaluator Agent 87

4.6 Simulation of Object Tracking System in JADE Platform 89

4.7 Summary 93

CHAPTER 5: RESULTS AND DISCUSSION 94

5.1 Introduction 94

5.2 Preliminary Experiment 94

5.2.1 Simulation Video Data 95

5.2.2 Tracking Performance Measurement 98
5.2.3 Evaluation Methods 98
5.2.4 Preliminary Experiment Results 99
5.2.5 Effect of Occlusion 102
5.2.6 Finding of Preliminary Experiment 105
5.3 Benchmark Dataset 105
5.4 Results and Analysis 106
5.4.1 Qualitative Result 107
5.4.2 Quantitative Result 111
5.5 Agents Communication in JADE 116
5.6 Summary 120
CHAPTER 6: CONCLUSION AND LIMITATION 122
6.1 Contributions 122
6.2 Future Work 124
REFERENCES 125
APPENDICES 145

LIST OF TABLES

Page

Table 2.1 Reviewed papers that use MAS for object tracking. 39

Table 2.2 Common performative used in a negotiation mechanism 42

Table 2.3 Reviewed benchmark dataset 43

Table 3.1 Role model for user 57

Table 3.2 Role model for object tracking manager 57

Table 3.3 Role model for object tracker 58

Table 3.4 Role model for proposal evaluator 58

Table 3.5 Descriptor of the type Tracking manager agent 62

Table 3.6 Descriptor of the type Object tracker agent 62

Table 3.7 Descriptor of the type Proposal evaluator agent 62

Table 3.8 A scenario for managing object tracking 69

Table 3.9 A scenario for tracking object 71

Table 3.10 The models of R1 until R6 for tracking object 72

Table 5.1 Video label and description 96

Table 5.2 Tracking result (SFDA) for six simulation videos 99

Table 5.3 Tracking result (SFDA) for the PETS 2009 and CAVIAR Video 111

Table 5.4 The lower boundary (_vmin) and upper boundary (_vmax) assigned for each object tracking agents when a different number of agents is used 112

Table 5.5 Tracking result on simulation video A1 dataset by using a different number of “CAMShiftAgents” 113

Table 5.6 Tracking result on simulation video A3 dataset by using a different number of “CAMShiftAgents” 113

Table 5.7 Tracking result on PETS 2009 view 001 S1 datasets by using a different number of “CAMShiftAgents” 115

Table 5.8 Tracking result on CAVIAR atrium dataset 116

LIST OF FIGURES

Page

Figure 1.1 The Essential process of Intelligent video surveillance System 3

Figure 2.1 Object representations. (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) complete object contour, (h) control points on object contour, (i) object silhouette (Yilmaz, Javed, & Shah, 2006). 13

Figure 2.2 An ellipsoidal region represents a human target in the PETS 2009 video frame..... 14

Figure 2.3 Steps for generating normalised colour histogram within the target (track window)..... 15

Figure 2.4 Example of the colour histogram of a target. 16

Figure 2.5 Back Projection of the target object in the video frame. 17

Figure 2.6 Mean-shift tracker window always has the same size when target car is farther away, and it is very close to the camera (OpenCV, 2017). 18

Figure 2.7 The state vector of a moving robot is described as its position (x, y) and its direction heading (θ). Source (Baltzakis, 2004)..... 26

Figure 2.8 The initial estimate values for $\hat{x}$ and P. (Source: Ulusoy, 2017)..... 28

Figure 2.9 The initial estimate values for $\hat{x}$ and P. (Source: Ulusoy, 2017)..... 28

Figure 2.10 A frame from CAVIAR entrance lobby video 44

Figure 2.11 Sample frames from CAVIAR shopping mall videos..... 44

Figure 2.12 Sample images from PETS 2009 video dataset. 46

Figure 2.13 Ground truth data of PETS 2009 S1L1 view 001. 47

Figure 3.1 Goal model for object tracking 56
Figure 3.2 Organisation model for object tracking 59
Figure 3.3 The domain model for object tracking 60
Figure 3.4 The agent-role coupling diagram for object tracking 61
Figure 3.5 Interaction-frame diagram for object tracking 65
Figure 3.6 Interaction-sequence diagram for object tracking 66
Figure 3.7 Knowledge model for object tracking multi-agents system 68
Figure 3.8 Behaviour model for object tracking multi-agent system 73
Figure 4.1 “VideoAgent” graphical user interface 77
Figure 4.2 Extracted “VideoAgent” implementation code to let the user choose a targeted object 79
Figure 4.3 Extracted “VideoAgent” implementation code to search for object tracker agents from directory facilitator (df) agent 81
Figure 4.4 Extracted “VideoAgent” implementation code calling for a proposal to track targeted object 82
Figure 4.5 Extracted “VideoAgent” implementation code to determine the best-tracking result 83
Figure 4.6 Extracted “CamshiftAgent” implementation code for registration and resources setup 84
Figure 4.7 Extracted “CamshiftAgent” implementation code for responding the CFP received 87
Figure 4.8 Extracted “EvaluatorAgent” implementation code for registration 88
Figure 4.9 Extracted “Evaluator Agent” implementation code for replying back the evaluation score 89

Figure 4.10 JADE Agent Management GUI 90
Figure 4.11 Tracking Results (Frame No. 1, 20, 40, 60, 80, 100: Order from left to right, top to bottom) based on parameter (_vmin = 1, v_max = 128) 91
Figure 4.12 Tracking Results (Frame No. 1, 20, 40, 60, 80, 100: Order from left to right, top to bottom) based on parameter (_vmin = 129, v_max = 256) 92
Figure 5.1 Video Frame (a) Frame no. 1, (b) Frame no. 10, (c) frame no. 22, (d) frame no. 30 and (e) frame no. 50 of simulation video A1. 97
Figure 5.2 Video Frame (a) Frame no. 1, (b) Frame no. 10, (c) frame no. 22, (d) frame no. 30 and (e) frame no. 50 of simulation video A2. 97
Figure 5.3 Video Frame (a) Frame no. 1, (b) Frame no. 10, (c) frame no. 12, (d) frame no. 13, (e) frame no. 22, (f) frame no. 30 and (g) frame no. 50 of simulation video A3. 97
Figure 5.4 Video Frame (a) Frame no. 1, (b) Frame no. 10, (c) frame no. 15, (d) frame no. 22, (e) frame no. 30 and (f) frame no. 50 of simulation video A5. 98
Figure 5.5 Particle distribution: a) particles cover a large area at the initial state; b) when object trajectory remains consistent between frames, the particle area shrunk; c) particle area become so small and fail to detect the moving object change direction..... 100
Figure 5.6 Frame Detection Accuracy of Frame No. 12 to Frame no. 31 for Video Sequence A3..... 103
Figure 5.7 Frame Detection Accuracy of Frame No. 12 to Frame no. 31 for Video Sequence A4..... 103
Figure 5.8 Frame Detection Accuracy of Frame No. 12 to Frame no. 31 for Video Sequence A5..... 104

Figure 5.9 Frame Detection Accuracy of Frame No. 12 to Frame no. 31 for Video Sequence A6 105

Figure 5.10 Track result from CAMShift Tracker in MATLAB (Frame number 15, 20, 25, 30 (Left to right). 107

Figure 5.11 Track result from CAMShift Tracker agent (Frame number 15, 20, 25, 30 (Left to right). 108

Figure 5.12 Track result from CAMShift Tracker in MATLAB (Frame number 1, 50, 70, 90, 110 and 130. Left to right, top to bottom). 109

Figure 5.13 Track result from CAMShift Tracker agent (Frame number 1, 50, 70, 90, 110 and 130. Left to right, top to bottom) with parameter (_vmin = 25 and _vmax = 50). 110

Figure 5.14 Interaction between *CAMShift* agents, *df* agents and *video01* agent. 118

Figure 5.15 Message from Video agent to Directory Facilitator agent requesting for agent Id that provides object tracking service. 118

Figure 5.16 Content of *Request* message from *video01* agent to *df* agent. 119

Figure 5.17 Content of *Inform* message from *df* agent to *video01* agent. 119

CHAPTER 1
INTRODUCTION

1.1 Introduction

Due to the rapid drop in the cost of video cameras in recent years and the rise in security awareness, demand on surveillance has shown a steady increase from time to time (Cropley, 2016). In general, conventional video surveillance networks are typically monitored by one or more security personnel, looking at several monitors showing footage recorded from several cameras. Therefore, it is very likely to have missed important events as they happen because it is tough for a human to monitor all these cameras at the same time (Ali & Dailey, 2012).

Recently, many research on intelligent video surveillance have begun to address ways to automatically process some of the data acquired from surveillance video, to assist human operators to effectively detect events (Mathur & Bundele, 2016) such as people fighting in a building (Fisher, 2005), loitering, theft, and unattended luggage in subway (Computational Vision Group, Reading University, 2007) as they happen. The success of event recognition in intelligent video surveillance relies heavily on the fundamental processes in the system such as motion detection and objects tracking in the video surveillance (Mathur & Bundele, 2016).

As a result, many types of research on object tracking have been conducted recently to promote efficiency and effectiveness in object tracking especially in a difficult situation such as when target tracked object is occluded (Yilmaz, Javed, & Shah, 2006). This research attempts to improve the process of object tracking through design and implement

a parameter selection method for Multi Continuously Adaptive Mean-shift (CAMShift) object tracking algorithm.

In this chapter, the background of the intelligent video surveillance is presented in the next section before proceeding to a discussion of the research question and research objectives. The background study provided a brief explanation of what are the essential processes that involve in an intelligent video surveillance system. The background information on intelligent video surveillance help to identify the underlying problems in current work and lead to the objectives of this research.

The objectives of this research are listed in this chapter too. The objectives describe the motivations of this research. Next, the scope of the research is discussed in the last section of this chapter where the outlines of the research are listed.

1.2 Background of Study

Intelligent video surveillance involves the study of computer vision techniques to detect, track and understand objects and events in video monitoring (Pérez, García, Berlanga, & Molina, 2005; Mathur & Bundele, 2016). A common intelligent video surveillance consists of several important processes, namely the video acquisition, video recording, video compression, object detection, object tracking and event recognition as shown in Figure 1.1.

Video acquisition in intelligent video surveillance relies on hardware technology for capturing video images of higher resolutions while recording and compression are the arts of creating more efficient methods to store and index more video information into smaller storage and faster retrieval. Object detection with event recognition as the highest-level

processing in intelligent video surveillance, that is, the effectiveness of event recognition relies heavily on the accuracy of object tracking.

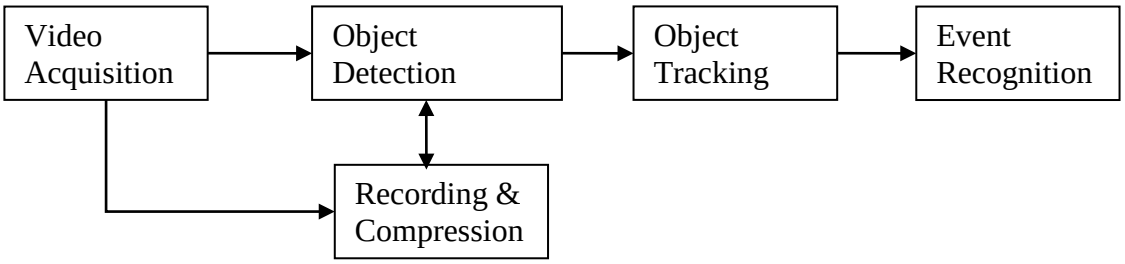


Figure 1.1: The Essential process of Intelligent video surveillance System

For instance, to recognise the path a person has walked through in a building, the system must first track the person based on the person's appearance from entry to exit. Another example is to recognise abnormal activity such as a person running in a subway station and a vehicle driven in the opposite direction; all these activities require accurate object tracking.

Object tracking is the process of following one or more objects in a video from the object's first appearance until the object exit the video scene. Some applications of object tracking system require the tracking system to continue the tracking if the object reappears in the video scene within a predefined time elapse. In other words, tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene. A tracker assigns consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object. The tracked object in object tracking can be anything of interest within the scene, from human to a soccer ball and the vehicle.

An ideal object tracking system is expected to operate in a variety of conditions. Any suitable algorithm must be able to adjust to the following challenges (Yilmaz, Javed, & Shah, 2006):

i. loss of information caused by the projection of the 3D world on a 2D image,
ii. complex object motion,
iii. non-rigid or articulated nature of objects,
iv. change in illumination,
v. partial and full object occlusions,
vi. scene illumination changes,
vii. real-time processing requirements,
viii. shadows of the moving objects

Among all the challenges, the first challenge is due to the limitation of the hardware. To solve the projection problem, more calibrated cameras can be added to the surveillance area, but that will increase the cost of the entire system. For occlusion problem (challenge v), current research common practice is to integrate a spatial-temporal model to predict the location of a target track object when the object disappears from the camera view (see Section 2.5). For challenge ii, iii, iv, vi, and viii, they can be solved by choosing a suitable tracker that utilise appearance model that frequently update its values (see Section 2.2) such as Mean-shift (Ali & Terada, 2009) and CAMShift (OpenCV, 2017). However, a tracker that based on appearance model requires proper selection of appearance model parameters (Yilmaz, Javed, & Shah, 2006). These parameters are the inclusive lower boundary array and the inclusive upper boundary array during the process of generating the colour histogram. The effect on these parameters is found based on the lab experiments

conducted in this research. The detail of the experiment is discussed in chapter 5. The purpose of the experiment is to identify the gap between current research achievements with actual practicality. The experiment also helps to answer some of the research questions.

1.3 Research Questions

There was many occlusion handling research previously. However, the limitation of the previous studies led to the establishment of this research. The research questions of this research are:

i. Can combining motion features to appearance features in an object model improve object occlusion handling compared to using appearance features on its own?
ii. Can better occlusion handling and object re-identification improve the overall tracking performance?
iii. What is the best way to fuse various features into an object model?

1.4 Aims and Objectives

This research aims to improve the performance of intelligent video surveillance system by solving the challenges for object tracking mentioned in the earlier sections by contributing originally:

- To design a method to select the inclusive lower boundary array and the inclusive upper boundary array for the colour histogram in CAMShift tracker.

- To formulate a combination approach of the spatial-temporal model tracker and appearance model tracker using the multi-agent system to handle occlusion problem in object tracking.
- To evaluate the proposed multi-agent system with simulation data and publicly available benchmark videos.

1.5 Research Scope

The tracking systems proposed within this research do not consider using any learned model approached to object detection, relying solely on simple detection techniques based on the analysis of motion segmentation output. Learned models have not been considered due to the requirement in training such models to accurately locate people and cars across a wide range of data, and due to motion segmentation research being the main component of this research. Ultimately the use of the simple motion-based detection techniques allows a broader range of data to be tested that would have otherwise been possible.

The videos from PETS 2009 and the CAVIAR database are used to train and evaluate the performance of object tracking systems. The performance of the tracking ability is based on accuracy calculated based on frame detection and recall rate.

1.6 Outlines of Thesis

Every stage of the research is documented. The outline gives a brief introduction to each chapter in this dissertation.

Chapter 1 presents the motivation, objectives and the scope of this research to improve the existing object tracking method in intelligent video surveillance system to enhance the accuracy of the tracking.

Chapter 2 provides a detailed survey of literature related to existing object tracking techniques. In this chapter, the strengths and the weaknesses of object tracking method used previously in the research are reviewed and their limitation mentioned. The object occlusion handling methods are also discussed and studied on their suitability in object tracking system.

Chapter 3 starts with highlighting the requirements and the specifications of the design of the proposed method. This chapter also explains the baseline design of the proposed system particularly on the modification of the current existing feature analysis and selection method by studying carefully on the criteria to select the best feature object tracking. The outline of the evaluation method used for evaluating the performance of the tracking system within this research is also documented in this chapter.

Chapter 4 explains the prototyping protocol of the proposed occlusion handling algorithm. The prototyping is done mostly using JADE. Script and the function written in JADE to implement the proposed system are discussed in detail in this chapter. The discussion also highlights several aspects derived during the development of the prototype.

Chapter 5 analyses the result after running the experiment using the developed proposed system. Information regarding sample sets used in the experiment, performance of the proposed system and the reasoning on the results obtained are explained one by one in this chapter.

Chapter 6 discusses the findings, accomplishments, limitations and future enhancement of the research. The achievement shows the contribution of this research in the field of object occlusion handling in object tracking system.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

This chapter presents the reviews of literature that are related to this research. The literature reviewed includes previous research on object tracking to know the general principles of existing object tracking and occlusion handling methods. From the literature review, the limitation of existing object tracking and object handling methods are identified. Since CAMShift tracking method stand out when comparing to other existing object tracking methods, it has been examined closely, and its limitation has been identified. The limitation of CAMShift motivates the proposed solution in this research which is to use multi-agent system (MAS) to find the optimum parameters for CAMShift tracker. A list of literature reviews on the usage of MAS in object tracking is also reported at the end of the chapter.

2.2 Object Tracking

Object tracking can be defined as the process of continue estimating the location of a tracked object in the image frames of a video. In other words, a tracker program assigns labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object (Yilmaz, Javed, & Shah, 2006).

One can simplify tracking by imposing constraints on the motion and appearance of objects. For example, almost all tracking algorithms assume that the object motion is smooth with no abrupt changes (Yilmaz, Javed, & Shah, 2006). Extra predefined

information such as the size and the number of objects, or the object colour and orientation, can simplify the problem by selecting a suitable method for tracking the object in the video.

Object tracking is an essential task within the field of computer vision. The availability of high-powered computers, high quality, and inexpensive video cameras, and the increasing need for automated video analysis has generated a great deal of interest in object tracking algorithms.

Therefore, the use of object tracking is valid in various tasks. For instance, in motion-based recognition, that is, human identification based on gait and automatic object detection. Automated surveillance also depends heavily on object tracking, which is, monitoring a scene to detect suspicious activities or unlikely events (Besada, Garc´ıa, Portillo, Molina, Varona, & Gonzalez, 2005; Ali & Dailey, 2012). Robots also rely on continuously tracking on the surrounding environment and obstacle for navigation (Benavidez & Jamshidi, 2011) and interaction with a human (Capi, Toda, & Nagasaki, 2010). Recently, object tracking has also been implemented on human-computer interaction, that is, gesture recognition, eye gaze tracking for data input such as in video game control and computer command using Microsoft Kinects™ sensor (Pal & Kakade, 2016). Traffic monitoring can also utilise the technology of object tracking, that is, a real-time gathering of traffic statistics to direct traffic flow. Some modern vehicles have also implemented object tracking in-vehicle navigation, that is, video-based path planning and auto obstacle avoidance capabilities.

Some comprehensive surveys have been conducted in the field of object tracking. Among them, Yilmaz, Javed, & Shah (2006) have done a comprehensive survey on the

topic, and Kim, Choi, Yi, Choi, & Kong (2010) also surveyed on intelligence surveillance system.

Object tracking in the videos has been active research since more than a decade ago. This interest is motivated by numerous applications, such as surveillance (Rasid & Suandi, 2010; Huang, Tao, Yuan, Li, & Tan, 2011; Ali & Dailey, 2012; Nie, Liu, & Su, 2012), human-computer interaction (Rautaray & Agrawal, 2012; Tao & Yu, 2012), and robot's navigation (Capi, Toda, & Nagasaki, 2010; Benavidez & Jamshidi, 2011). Many of these applications of object tracking have the goal to have the ability to automatically understand events happening at a site (Elgammal, Duraiswami, Harwood, & Davis, 2002). To have a higher level of understanding of events requires that certain lower level computer vision tasks to be performed. These may include tracking objects, handling occlusion and detection of unusual motion.

Object tracking is a process of monitoring an object's spatial and temporal changes during a video sequence, including its presence, position, size, and shape (Ali & Mirza, 2006). Many challenges remain big on tracking objects because of abrupt object motion, changing appearance patterns of objects and the scene, non-rigid object structures and most significantly in handling occlusion of the tracked object (Mathur & Bundele, 2016).

Many studies have been conducted to improve object tracking technique reliability and accuracy. An article written by Pulford (2005) summarised the techniques in widespread use and classified them into 35 different algorithmic types as well as providing a comprehensive literature survey of object tracking. Another comprehensive study was also produced by Yilmaz, Javed & Shah (2006) who reviewed the techniques used in object tracking and categorised the techniques based on the object and motion representations

used. The review also comprehensively provided detailed descriptions of representative methods in each category and examined their strength and weaknesses.

2.3 Kernel-Based Object Tracking

According to Yilmaz, Javed, & Shah (2006), tracking methods are classified into three categories, namely the point tracking, silhouette tracking and kernel-based tracking. The categorisation is based on the objects representation methods. For example, if an object is represented as a point or a set of multiple points (e.g. (a) and (b) in Figure 2.1), then it is known as point tracking.

For kernel-based object tracking, an object is represented using a geometric shape representation like a rectangle or an ellipse (e.g. (c) and (d) in Figure 2.1), parametric motion models like affine or projective transformations are appropriate. These representations can approximate the motion of rigid objects in the scene.

For a non-rigid object, silhouette or contour is the most descriptive representation (e.g. (e), (f), (g), (h) and (i) in Figure 2.1). Both parametric and nonparametric models can be used to specify their motion (Yilmaz, Javed, & Shah, 2006).

When comparing the three types of tracking methods, point tracking requires less computational cost but with scarification of accuracy. For silhouette tracking, the accuracy is high, and it can handle transformable tracking object, but the computational cost is much higher. Kernel-based tracking like mean shift tracker is widely used because it can provide high accuracy and the computational cost is lower than silhouette based tracking (Comaniciu & Ramesh, 2000; Bernhardt, 2012; Yilmaz, Javed, & Shah, 2006). Sample

code on mean shift tracker by Bernhardt (2012) is available at MATLAB File Exchange website. The code is easy to use and positively rated by the user at the website.

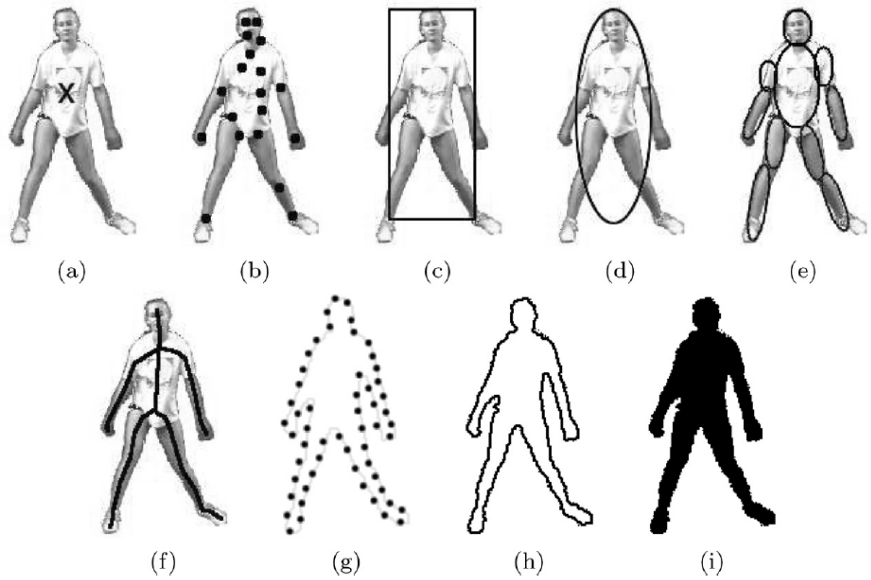


Figure 2.1: Object representations. (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) complete object contour, (h) control points on object contour, (i) object silhouette (Yilmaz, Javed, & Shah, 2006).

2.3.1 Continuously Adaptive Mean-Shift (CAMShift) Tracking

Continuously Adaptive Mean-shift (CAMShift) is an improved version of Mean-shift for object tracking. To understand CAMShift, one must first understand how Mean-shift tracker works. Mean-shift tracker was first introduced as a kernel-based object tracking method by Comaniciu, Ramesh, & Meer (2003). Their paper has been cited more than 5,000 times. After its introduction, Mean-shift tracker has quickly been adopted by many other researchers (Deguchi, Kawanaka, & Okatani, 2004; Sattar & Dudek, 2006; Babu & Makur, 2007; Ali & Terada, 2009; Yun, Kim, & Lee, 2010; Liu, Hu, & Aggarwal, 2011; Lee, Kim, & Sohn, 2011; Chen, Song, Kyoungho, & Guo, 2013; Zhang & Wong, 2014; Shilpa, 2016; Zhao, Xiang, Cao, & Wang, 2016; Han, Zhang, Wen, Xie, & Li, 2017). The

literature review on these works that adopted mean-shift tracker supports that mean-shift tracking is a reliable approach toward target representation and localisation the central component in visual tracking of non-rigid objects (Comaniciu, Ramesh, & Meer, 2003).

A mean-shift tracker tracks an object by performing the following steps (i-v), and CAMShift has an extra step (vi):

i. First, an object target for tracking is chosen from the initial frame of a video. A human target is commonly represented by a kernel which is either a bounding box or ellipsoidal region in the video frame (Figure 2.2) because it provides a better fit than a rectangle.



Figure 2.2: An ellipsoidal region represents a human target in the PETS 2009 video frame.

The ellipsoidal region defined as track window ($track\_window$),

$$track\_window = (c, r, w, h)$$

where (c, r, w, h) represent image column, row, width and height respectively

Next, to ensure low-computational cost imposed by real-time processing object tracking, discrete colour densities m-bin histograms are commonly used as the

feature model to represent the target. The steps to generate a colour histogram based on the hue of the target is summarised in Figure 2.3.

```
# set up the ROI for tracking
roi = frame[r:r+h, c:c+w]
hsv_roi =  cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv_roi, lowerb, upperb)
roi_hist = cv2.calcHist([hsv_roi],[0],mask,[180],[0,180])
cv2.normalize(roi_hist,roi_hist,0,255,cv2.NORM_MINMAX)
```

Figure 2.3: Steps for generating normalised colour histogram within the target (track window)

Note that in Figure 2.3, only hue is considered here to generate the colour histogram. To generate a better-normalised histogram for tracking, the low light value in hue has to be removed using *cv2.inRange( )* function. The *cv2.inRange( )* checks if the pixel value of the target (*roi* array elements) lie between the elements of two other arrays, namely the inclusive lower boundary array (*lowerb*) and the inclusive upper boundary array (*upperb*):

$$dst(I) = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_{\dots} \cap lowerb(I)_{\dots} \leq src(I)_{\dots} \leq upperb(I)_{\dots} \cap lowerb(I)_n \leq src(I)_n \leq upperb(I)_n$$

Where *I* is the index of the element in the array and n is the dimension of the array.

ii. After the low light value in hue is removed, the probability of the feature $\hat{q}_u$ for $u = 1 \dots m$ in the target model is then computed as:

Target model: $\hat{q} = \{\hat{q}_u\}_{u=1\dots m}$ $\sum_{u=1}^{m} \hat{q}_u = 1$ where

$$\hat{q}_u = C \sum_{i=1}^{n} k(\|x_i^*\|^2)\, \delta[b(x_i^*) - u]$$

and δ is the Kronecker delta function. The function b associates to the pixel at location (x_i^*) the index $b(x_i^*)$ of its bin in the quantised feature space and C is the normalisation constant,

$$C = \frac{1}{\sum_{i=1}^{n} k\left(\left\|x_i^*\right\|^2\right)}$$

The code for generating a normalised histogram in OpenCV using Python is shown in line 3 and line 4 in Figure 2.3. A sample result of the colour histogram of a target is shown in Figure 2.4.

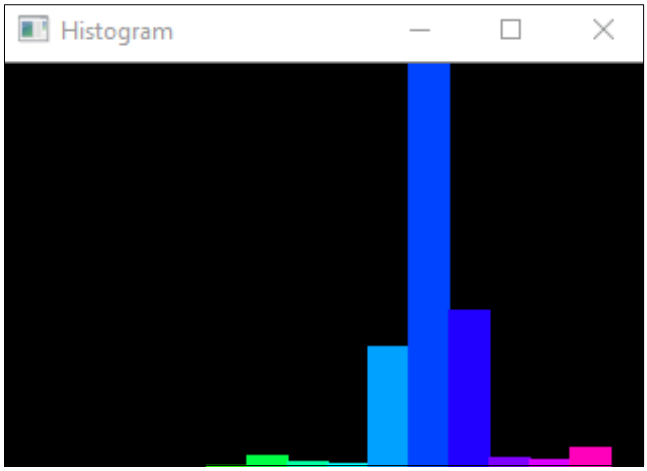


Figure 2.4: Example of the colour histogram of a target.

iii. Next, a back projection of the colour histogram values is applied to each pixel of the subsequent frame based on its colour. Back projection is a way of recording how well the pixels of the frame fit the distribution of pixels of the target model histogram. The histogram of the target is used to convert the frame image into a target-colour probability image as shown in Figure 2.5. Back projection of the target object in video frame when pixels in the frame are compared to target model histogram (the ellipsoidal region is added to indicate the target).



Figure 2.5: Back Projection of the target object in the video frame.

iv. After that, based on the location of the target in (i) and the distribution of back projection, a new mean is calculated. Then the target centre is shifted to the position of the new mean, and the target model is updated. This position is used to track the target iteratively in the subsequent frames of video. The size and the orientation of the target window do not change in Mean-shift tracking throughout the iteration. However, this is not good for object tracking (OpenCV, 2017). As shown in Figure 2.6, Mean-shift tracker always uses the same size window when the targeted car is farther away, and it is very close to the camera. When the target car is close, the window only represents part of the car rather than the entire car like it was in the earlier frame.

v. To handle the same size window problem, the window size needs to be able to adapt to the size and rotation of the target in a different frame. CAMShift updates the size of the window as, $s = 2 \times \sqrt{\frac{M_{00}}{256}}$. It also calculates the orientation of best fitting ellipse to it. Lastly, CAMShift applies the mean-shift with new scaled search window and previous window location. The process is continued until the shifting of the window is less than a threshold value.



Figure 2.6: Mean-shift tracker window always has the same size when target car is farther away, and it is very close to the camera (OpenCV, 2017).

2.3.2 CAMShift Limitation

CAMShift is an extension of Mean-shift, and many limitations of Mean-shift are inherited to CAMShift. For example, CAMShift would lose track of a target object when a full occlusion occurs or when another object with similar appearance overlapped it, even if it is not entirely occluded (Guo, Li, Zhao, & Zhao, 2016).

Another limitation of CAMShift tracker is it requires proper selection of appearance model (Yilmaz, Javed, & Shah, 2006). To obtain a better segmentation of a track object in video sequences, the selection for the inclusive lower boundary array and inclusive upper boundary array during the process of generating the colour histogram discussed earlier in 2.3.1 are crucial. The selection of these values is critical to achieve good segmentation of target object and affect the tracking result. Many recent research are still trying to improve CAMShift in object tracking (Guo, Li, Zhao, & Zhao, 2016; Zhao, Xiang, Cao, & Wang, 2016; Han, Zhang, Wen, Xie, & Li, 2017). An experiment has been conducted, and Section 5.4 later to show how these parameters affect tracking result.

In many cases of object tracking, especially in crowded surveillance videos, it is impossible to have one set of parameters that will work in optimum level for tracking different targets. The reason is that of the target object for tracking, and the crowd in the scene will be different every time and therefore require different parameters values.

2.4 Occlusion

In video object tracking, occlusion happens when a tracked object or its key attributes used to recognise its identity is not available for a camera sensor to keep tracking its spatial state while the object is still present at the scene. Occlusion occurs due to numerous reasons.

Occlusion is categorising into self-occlusion, inter-object occlusion and background occlusion (Yilmaz, Javed, & Shah, 2006). Self-occlusion happens when one part of a tracked object occludes another. This situation most frequently arises while tracking articulated objects. An example of self-occlusion is when tracking a person using the face as a key attribute (Bradski, 1998); self-occlusion happens when the target person turns her face away from the camera or closes her face using her hand. Inter-object occlusion occurs when two objects being tracked occlude each other. When a tracked object stands behind another tracked object, often the object that is at the back will disappear from the camera. Occlusion by the background occurs when a tracked object stands behind a building structure or furniture that is static in the scene.

Occlusion has been a serious problem in object tracking (Guha, Mukerjee, & Venkatesh, 2005) and many types of research such as Baum, Izhaki, Rivlin, & Katzir (2014), Li, Bloch, & Shen (2015), Camuñas-Mesa, Serrano-Gotarredona, Ieng, Benosman, & Linares-Barranco, (2017), Li, Zhou, Chen, Chan, & Ju (2017), and Li, Cao, Liu, & Zhang (2017) have been done to ensure tracking method can continuously track an object during and after an occlusion event. Handling occlusion in a single object environment is a straightforward task. If only a single object appears within the scene, the tracking task will only involve detecting object appearance.

However, for a scene with multiple object appearance, handling occlusion becomes complicated. For inter-object occlusion, previous research such as Maccormick & Blake (2000), Ali & Dailey (2012), Wu, Zhong, Tsao, Yang, Chen, & Chien (2017), Li, Cao, Liu, & Zhang (2017) viewed the multi-object tracking problem as a challenging problem,

especially when the targets are “identical” in the sense that the same model is used to describe all targets.

Occlusion handling ability of a tracking technique also depends on the severity of occlusion. Existing literature (Guha, Mukerjee, & Subramanian, 2011) has identified many occlusion categories in the context of tracking in ad hoc manner. One way to categorise occlusion is based on the severity of the occlusion. Occlusion can be categorised as non-occlusion, partial occlusion, full occlusion and long-term full occlusion. In multi-camera setup, the interval between leaving the field of view of a camera and moving to another camera or return to the previous camera can also be considered as occlusion.

During non-occlusion, the tracked object appears as a single blob having all tracking features exposed to the camera sensor for tracking. Most of the existing tracking methods that are based on appearance attribute to track a target such as a template matching and mean-shift methods can track non-occlusion object accurately (Baum, Izhaki, Rivlin, & Katzir, 2014).

Partial occlusion happened when some of the key features of the tracked object are hidden from the camera during tracking. Partial occlusion can happen when part of the tracked object is blocked by other objects or background structure or during self-occlusion. Partial occlusion of an object by a foreground structure is hard to detect since it is difficult to differentiate between the object changing its shape and the object getting occluded. When a partial occlusion occurs, a tracker that only adopts simple template matching without consideration of the evolution of the object over time will fail. Therefore, many early developed tracking methods such as mean-shift (Yun, Kim, & Lee, 2010; Liu, Hu, & Aggarwal, 2011) include adaptive step into the tracking method to handle partial occlusion.

Full occlusion happens when a tracked object is completely not visible while knowing that the object has not left the area of view of the camera. When a full occlusion occurs, no one method that relies on image appearance can continue to track the object because there will be no more appearance clue left in the scene for occlusion. To handle full occlusion, many of the state of the art tracking method incorporated state estimator into the tracking process. For instance, some previous works (Zhu, Ji, Fujimura, & Lee, 2002; Ali & Terada, 2009; Zhao, Qiao, & Men, 2009) used Kalman filter for estimating the location and motion of objects in videos.

The problem of combining the appearance model and the state estimator remains an active research (Guo, Li, Zhao, & Zhao, 2016). It is vital to combine two or more tracking methods to increase the tracking accuracy but at the same time remain responsive. Therefore, it is one of the objectives of this research to formulate a combination approach of the spatial-temporal model tracker and appearance model tracker to handle occlusion problem in object tracking using the multi-agent system. Multi-agent system performs better with respect to hardware resources and software limitations when a problem requires high urgency (van Lon & Holvoet, 2016).

2.5 Occlusion Handling

Occlusion handling is an inevitable problem in object tracking. The problem of occlusion occurs during the occlusion and after the occlusion. During the time of occlusion, two types of challenges may arise. First, when two foreground objects occlude one another, the foreground blob of the objects will group as one, and it will become challenging to classify the pixels in the blob to the respective object accurately. Secondly, during occlusion, the

actual location of a tracked object is difficult to determine since the visibility of the tracked object becomes limited or entirely missing.

After the event of occlusion, especially the full occlusion, the challenge of associating the reappeared object is tough. When an object reappears, it is difficult to decide if an object is a new object or it is an object that reappears after occlusion. Compares to single object tracking, the problem is more complicated especially when tracking multiple objects with a similar appearance. The solutions to this problem are generally known as occlusion recovery methods.

Various methods have been proposed to handle occlusions problem. These methods are summarised based on the nature of the problem the methods tried to solve. These methods are listed in the following subsections.

2.5.1 Depth Analysis

Solution to merge foreground blob problem is through determining the depth of the tracked objects (Harville, Gordon, & Woodfill, 2001; Greenhill, Renno, Orwell, & Jones, 2008; Ma & Chen, 2010). By determining the depth of the object, when two or more objects occlude one another creating a grouped blob, the depth model of the object can be used to separate the object blobs because the object nearer to the camera will have shorter depth compare to the object that is further. Depth analysis makes segmenting the object blob during occlusion easier.

Various methods were used to estimate the depth of the tracked object and the structure in a scene. Previous researchers (Harville, Gordon, & Woodfill, 2001; Ma & Chen, 2010) used video sequences obtained with stereo cameras to calculate the depth of the objects in

the video. A per-pixel Gaussian mixture was applied by Harville, Gordon, & Woodfill (2001) to model recent scene observations in the combined space of depth and luminance-invariant colour. Another method of generating the probability density functions for the depth of the scene at each pixel from a training set of detected blobs through the projection of the 3D scene as 2D was proposed by Greenhill, Renno, Orwell, & Jones (2008). The occlusion landscape is made explicit such that structures can defeat the process of tracking individuals through the scene. In Suarez & Murphy (2012) article, they proposed using Microsoft Kinects™ to recognise hand gesture by tracking hands (object) when the players interact with the computer program.

To detect depth in object tracking, a stereo must be used. The problem of using stereo camera is that the camera must be calibrated from time to time to maintain accuracy (Rathnayaka, Baek, & Park, 2017). Other than that, stereo cameras are more expensive than an ordinary camera.

2.5.2 Optimal Camera Placement Method

The chance of occlusion can be reduced by an appropriate selection of camera placement positions (Batista, 2004). For instance, if a 360-degree ultra-wide-angle camera is mounted on the ceiling of a room, which is when a birds-eye view of the scene is available, occlusions between objects on the ground do not occur. However, the ultra-wide-angle camera will distort images captured and objects captured in the video will have very identical appearance features. Other than that, installing more cameras in the surveillance scene can also help to solve occlusion problem (Batista, 2004; Mozerov, Amato, Roca, & González, 2009; Raman, Sa, & Majhi, 2012). However, the cost of implementation will increase when more cameras are needed to monitor the objects. When many cameras are

installed, at the incidence where an object is obstructed from view by a camera, the object may still be visible from other cameras which reduce the chances of being occluded.

2.5.3 Motion Prediction

In a review article written by Yilmaz, Javed, & Shah (2006), they stated that previous researchers believe they can exploit the knowledge of the position and the appearance of the occluding object and occluded object to detect and resolve occlusion. A conventional approach to handle complete occlusion during tracking is to fuse the object motion with appearance model by linear dynamic models (Yin, Bo, Zhao, & Zou, 2013) or by nonlinear dynamics and, in the case of occlusion, to keep on predicting the object location until the object reappears. For example, a Kalman filter is used for estimating the location and motion of objects in (Ali & Terada, 2009; Zhao, Qiao, & Men, 2009; Ren & Hao, 2012; Guo, Li, Zhao, & Zhao, 2016; Zhang, Yang, Zhang, & Shen, 2016; Kou, Wang, Guo, Zhang, & Liu, 2017).

A nonlinear dynamic model was used in (Isard & MacCormick, 2001; Duan, Cai, & Yu, 2009; Tang & Zhang, 2011) where a particle filter was employed for state estimation. Among the kernel-based tracking methods, most of the current works focused on Kalman Filter, Particle Filter and Mean-shift tracking.

The Kalman filter is named after Rudolf Kalman, who is the primary developer of its theory. It is an optimal estimation algorithm that predicts the parameter of interest such as location, speed, and direction in the presence of noisy measurements. Common applications of Kalman filters include guidance, navigation, and control systems; computer vision systems; and signal processing.

Kalman filter is suitable for object tracking when the object is occluded where its position in the video can only be estimated based on its state in the previous frame. Although the position of a target object can be calculated based on motion model, in reality the estimated position of the object can be entirely different because the motion of the object in real world can be affected by noise such as the table surface friction of a moving ball, terrain of the walking path of a pedestrian and the traffic light at the junction of a road where a tracking car is moving. Kalman filter is known as a "filter" because the process of finding the "best estimate" from noisy data amounts to "filtering out" the noise. The word noise here refers to the difference between the estimated state and the actual state. In object tracking, the estimated state of the target object can be calculated using its motion model. However, in reality, the actual state can be different because of the targeted object. For example, the object can change its motion when the object decided to slow down at a junction or the terrain of the path suddenly changed. However, a Kalman filter does not just clean up the data measurements but also projects these measurements onto the state estimate (Kleeman, 1996).

Consider when tracking a moving target, such as a robot car, the concept of the state vector of the object is fundamental to the description of its linear, discrete-time dynamical system. The state vector of the robot car is shown in Figure 2.7.

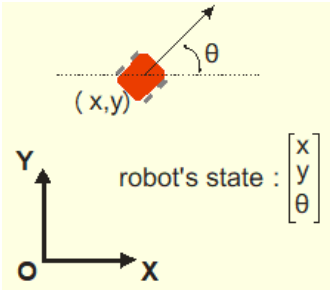


Figure 2.7: The state vector of a moving robot is described as its position (x, y) and its direction heading (θ). Source (Baltzakis, 2004)

The state vector or simply state, denoted by x_k, is defined as the minimal set of data that is sufficient to describe the unforced dynamical behaviour of the system uniquely; the subscript k denotes discrete time. In other words, the state is the least amount of data on the past behaviour of the system that is needed to predict its future behaviour. Typically, the state x_k is unknown. To estimate it, we use a set of observed data, denoted by the vector y_k.

A Kalman filter is used to estimate the state of a linear system where the state is assumed to be distributed by a Gaussian. Kalman filtering is composed of two steps, prediction and update. The update step uses the state model to predict the new state of the variables:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$$\hat{P}_k^- = AP_{k-1}A^T + Q$$

where $\hat{x}_k$ and $\hat{P}_k$ are the state and the covariance predictions at time k. A is the state transition matrix which defines the relation between the state variables at time k and $k-1$. Q is the error covariance of the noise (Bu_k).

Here, the system model is used to calculate the a priori state estimate and the error covariance P. For our single-state system, P is the variance of the a priori estimate. The error covariance P can be thought of as a measure of uncertainty in the estimated state. This variance comes from the process noise and propagation of the uncertain $\hat{x}_{k-1}$. At the very start of the algorithm, the $k-1$ values for x̂ and P come from their initial estimates as shown in the Figure 2.8.

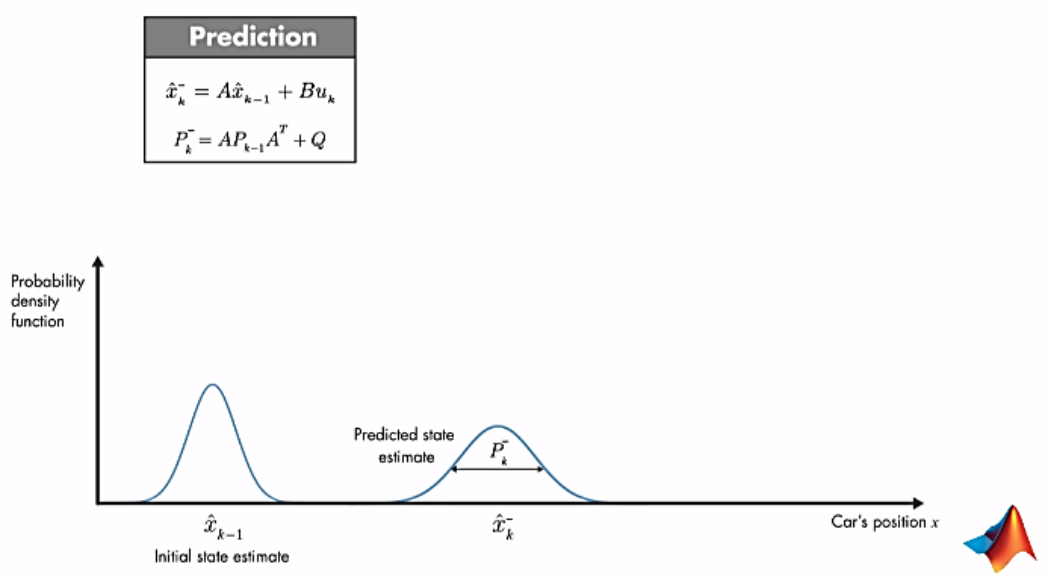


Figure 2.8: The initial estimate values for x̂ and P.
(Source: Ulusoy, 2017)

Then the correction step uses the current observations $\hat{y}_k$ to update the object's state as shown in the Figure 2.9.

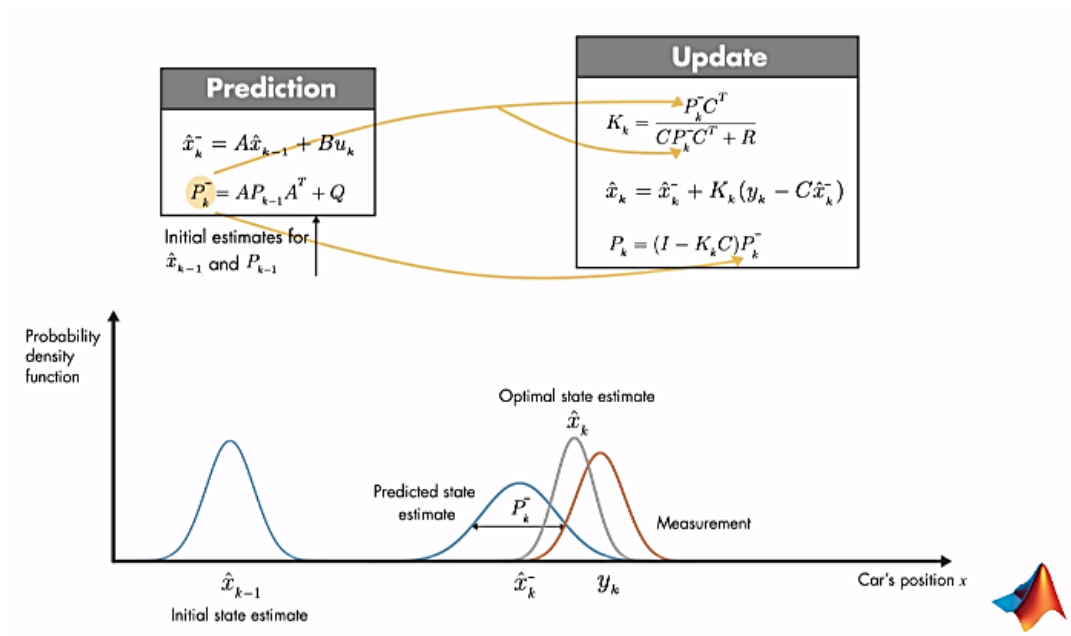


Figure 2.9: The initial estimate values for x̂ and P.
(Source: Ulusoy, 2017)

where ($y_k - C\hat{x}_k^-$) is called the innovation, C is the measurement matrix, K is the Kalman gain, which is the Riccati Equation used for propagation of the state models and R is the limit of a Gaussian distribution. Note that the updated state $\hat{x}_k$ is still distributed by a Gaussian.

The next step of the algorithm uses the "a priori estimates", $\hat{x}_k^-$ calculated in the prediction step and updates them to find the "a posteriori estimates", $\hat{x}_k$ of the state and error covariance. The Kalman gain is calculated such that it minimizes the a posteriori error covariance. Let this bar represent the calculation of $\hat{x}_k$. By weighing the correction term, the Kalman gain determines how heavily the measurement and the "a priori estimate" contributes to the calculation of $\hat{x}_k$. If the measurement noise is small, the measurement is trusted more and contributes to the calculation of $\hat{x}_k$ more than the a priori state estimate does. In the opposite case, where the error in the a priori estimate is small, the a priori estimate is trusted more and the computation of $\hat{x}_k$ mostly comes from this estimate.

Once we have calculated the update equations, in the next time step the "a posteriori estimates" are used to predict the new "a priori estimates" and the algorithm repeats itself. Notice that to estimate the current state, the algorithm does not need all the past information. It only needs the estimated state and error covariance matrix from the previous time step and the current measurement. This is what makes the Kalman filter recursive. When tracking an object, Kalman filter is computationally more efficient than computing the estimate directly from the entire past observed data at each step of the filtering process. (E. Cuevas, 2005).

In the research of Mirabi & Javadi (2012) and Wang, Huo, & Zhang (2011), they have experimented using Kalman filter tracker on real-world video sequences. In both

experiments, their system can deal with stressful situations such as noise, shadow and illumination changes. Kalman filter tracker is also claimed as computationally cost-effective in tracking the object. These experiments do not consider tracking a moving object with arbitrary trajectory.

Lien, Hsia, & Su (2015) proposed a method of tracking a moving object for improving the performance of the system by combining CAMShift and Kalman filter. In their system, moving objects are caught using CAMShift when the appearance of the object is not occluded. Then, when the object is occluded, The Kalman filter can provide the estimated positions continuously. The estimation position will be the initial position of CamShift when the target appears again.

Yilmaz, Javed, & Shah (2006), Liu, Hu, & Aggarwal, (2011), Salhi & Yengui Jammoussi (2012) Bedruz, Sybingco, Bandala, Quiros, Uy, & Dadios, (2017) and Zhang, Zhang, Wang, & Yu, (2017) and have developed a tracker based on mean-shift to derive target object candidate based on appearance model similarity. The results from their research showed that the mean-shift tracker is robust to partial occlusion, background clutter, target scale variations and rotations in depth. However, Comaniciu, Ramesh, & Meer (2003) and Yilmaz, Javed, & Shah (2006) did not discuss on how the mean-shift tracker will perform in a situation of full occlusion and an object with arbitrary trajectory. Recent research like Bedruz, Sybingco, Bandala, Quiros, Uy, & Dadios, (2017) and Zhang, Zhang, Wang, & Yu, (2017) have to use mean-shift to track vehicle and pedestrian but both claimed that the method would fail when the targeted track object in under occlusion and therefore mean-shift need enhancement in their research.

Another state estimator that is used for object tracking is Particle filters. Particle filters provide robust tracking of moving objects in a cluttered environment especially in tracking moving objects that move in non-linear and non-Gaussian trajectory. Cho, Jin, Pham, & Jeon (2006) has used Particle Filter tracker with images' grey level model to track the moving object while Liang (2012) combined Particle Filter tracker with colour and shape model to track objects in video sequences.

The Kalman Filter, Particle Filter and Mean-shift tracking methods are experimented in this research and published in Lee, Liew, Cheah, & Wang (2014). In the paper, the Particle filter tracker code was adopted from Paris (2007) work, the Kalman filter was adopted from FarrokhNejad, (2007) work and the mean-shift tracker was adopted from Bernhardt, (2012) work. The code was highly rated in MATLAB central where researchers gather to exchange idea. The result of the tracking is also shown in Table 5.2 in Chapter 5.

2.5.4 Occlusion Handling Limitation

All the occlusion handling methods discussed earlier each has their limitation. Depth analysis occlusion technique requires a special device like the stereo camera to measure the depth of the object. For optimal camera placement techniques, extra money must be spent to purchase more cameras for the system. On top of that, more cameras mean more data to process. Thus, this will increase computational time drastically. Motion prediction relies heavily on object recognition process. When an object is occluded for too long, it is impossible for motion prediction to track the object based on motion prediction alone.

Because of these limitations, this research proposes to combine the well-accepted appearance based CAMShift object tracker with Kalman filter as motion prediction to

handle short occlusion. A multi-agent system is designed to optimise CAMShift object tracker to handle long occlusion so that the occluded object could be recognised when it reappears in the video scenes. The detail on the fusion model is discussed in Chapter 3.

2.6 Multi Models Object Tracking

Many past research works have proposed to combine appearance model like Mean-shift and CAMShift and motion predication spatial model (i.e. Kalman Filter and Particle Filter) to improve tracking result (Tang & Zhang, 2011; Yin, Bo, Zhao, & Zou, 2013;Yao, Xia, Shen, Zhou, & Niu, 2016; Han, Zhang, Wen, Xie, & Li, 2017). The common approach to combine multiple models is by setting one of the models as primary tracker while the other model as a supportive model. For instance, in Comaniciu & Ramesh (2000) and Tang & Zhang (2011), the primary tracker was a Mean-shift tracker (appearance model) but once the tracked object is occluded to a certain degree (when the similarity measure of the track object is less than a predetermined threshold value) the system will switch to the supportive model such as Kalman filter (Comaniciu & Ramesh, 2000) and Particle Filter (Tang & Zhang, 2011) accordingly. Han, Zhang, Wen, Xie, & Li (2016) and Guo, Li, Zhao, & Zhao (2016) have proposed to track the object by combining CAMShift and Kalman filter. In both previous researches, Kalman filter is used to assist the CAMShift tracker the track the object when the target object is under occlusion. Han, Zhang, Wen, Xie, & Li (2016) define occlusion as when the ratio of current tracking bounding box area over the initial bounding box area is less than 0.3 or 30%.

By combining appearance model with the spatial model, it solves the problem of full occlusion by “guessing” the location of the tracked object when the object is fully

occluded. The method does not solve the underlying problem of appearance model where a right parameter setting is needed to track an object as mentioned in Section 2.3.2.

Besides, using one model as primary and another model as supportive does not fully utilise the potential of both object tracking model. For instance, when tracking an object that has an arbitrary movement pattern (Lee, Liew, Cheah, & Wang, 2012). An experiment has been conducted to test the performance of the combined tracker, and it is found that when the tracked object in the video make a quick turn, the tracker will lose accuracy in tracking immediately and slowly recover from the problem. The condition gets worse when the target tracked object is occluded when it makes the turn; the tracker sometimes will miss tracking the occlusion as the target track object (Lee, Liew, Cheah, & Wang, 2014).

2.7 Multi-Agent System in Object Tracking

As discussed in the previous section, the main problem in object tracking and object occlusion handling in surveillance video remains on how to set proper parameters for object representation in kernel-based object tracking like CAMShift. To design a system to look for the autonomous dynamic setting of the parameter for object representation, it is important to choose a software modelling method that supports simplification of dynamic interaction component. Such a model can be implemented in various software engineering paradigms like functional paradigm, object-oriented paradigm or the recently emerging paradigm, multi-agents oriented software engineering.

During the past two decades, with the increase in complexity of projects associated with software engineering, multi-agents oriented software engineering concepts that originated

from Artificial Intelligence (AI) have been considered to devise a new paradigm for handling complex systemised programming (Akbari, 2010). Software architectures that contain many dynamically interacting components, each with their own thread of control and engaging in complex, coordinated protocols, are typically orders of magnitude more complex to engineer correctly and efficiently than those that simply compute a function of some input through a single thread of control (Sterling & Taveter, 2009). As a consequence, a major research topic in object tracking over at least the past two decades (Refer section 2.7.1) has been the development of tools and techniques to model, understand, and implement object tracking systems in which interaction is the norm.

MASs perform better with respect to hardware resources and software limitations when a problem has all following three properties: namely high urgency, medium to high dynamism, and medium to large scale (van Lon & Holvoet, 2016). To achieve autonomous dynamic setting of the parameter for object representation for CAMShift, this research explores the opportunity on Multi-Agent System (MAS) because the task for autonomous of selection optimum parameters for object tracking fulfil the advantage of MAS that van Lon & Holvoet, (2016) reported.

Next, MAS is suitable for developing object tracking systems from the aspect of high dynamism. In every surveillance video, the scene of the video changes according to the time and the content in the video, a tracked object can appear differently in each tracking event. For instance, the subject for tracking may be a man in black in the current event while it may be a little girl who is wearing a pink dress in another event.

The last criteria that support the usage of MAS in finding a good set of parameters for CAMShift object tracking are the scale of the solution. Because the interaction between the

target object for tracking with the background and other objects in the video could not be pre-determined. Hence, multiple combinations of parameter need to be implemented as multi-agents for tracking the object in the video. The possibility of the combination is great, and therefore the scale of the solution is high.

These three properties reported by van Lon & Holvoet, (2016) matches with the task of the problem on looking for proper parameters for object representation in CAMShift object tracking. For object tracking in video surveillance, the system response is urgent; it needs to identify an object from 12-30 frames a second. Each frame can consist of thousands to millions of pixels (Cropley, 2016).

2.7.1 Previous Work of MAS in Object Tracking System

Multi-agent System (MAS) is defined in Wooldridge (2002) as a system which composed of multiple computing elements that can perform their designed purpose autonomously while interacting with other computing elements. These computing elements are known as agents. The interaction between agents can be conducted by engaging in social activity such as cooperation, coordination, negotiation, and the like.

MAS has only been studied since about 1980 and gained widespread recognition since about the mid-1990s (Wooldridge, 2002). This rapid growth has attracted many beliefs that agents can be an appropriate software paradigm for object tracking systems (Marchesotti, Piva, & Regazzoni, 2003; Li, Wang, Qi, & Li, 2010; Castanedo, García, Patricio, & Molina, 2011; Baum, Izhaki, Rivlin, & Katzir, 2014; Fekir & Benamrane, 2015; Sánchez, Rodríguez, De la Prieta, De Paz, & Bajo, 2015).

Multi-agent systems is a natural metaphor for understanding and building a wide range of what users might crudely call artificial social systems. The ideas of multi-agent systems are not tied to a single application domain, but, like objects before them, being able to find currency in a host of different application domains.

Previous research like Marchesotti, Piva, & Regazzoni, (2003) and Castanedo, García, Patricio, & Molina (2011) demonstrated the possibility of using the interaction feature of agents in the Multi-Agent System, to automatically synchronise the videos from different camera in a network of cameras. It is important to synchronise the video data from the different camera because only when the video time is syncronised, the image from the video frame can be used to identify an object in different cameras. When using the multi-agent system for synchronisation, it gives an extra advantage of scalability because a new agent can be added to the system from time to time.

Li, Wang, Qi, & Li (2010), has proposed a different way to use the multi-agent system in object tracking by improving the particle filter algorithm when tracking the moving targets in video image sequences. Multi-agent system was implemented to increase the particle diversity through implementing the particles in the system as agents. The agents have the co-evolutionary behaviours such as competition, crossover, mutation and self-learning. As a result, the system achieves better performance when tracking objects in complex video scenes.

Baum and his colleagues (Baum, Izhaki, Rivlin, & Katzir, 2014) implemented a multi-agent system that employs two layers to deal with occlusions of different lengths. The first layer is for short- or medium-term occlusions based CAMShift object tracking algorithm and Kalman filter. The second layer is for reidentification of occluded object that has been

occluded for long-term based on the game theoretic approach known as “prediction with expert advice” that was introduced by Cesa-Bianchi' & Lugosi' (2006). The research validates that their learning algorithm showed improved results and produced a data structure that can be shared among agents or sent to the central control of a multi-agent system. The research also validates that the implementation of CAMShift and Kalman is suitable for tracking object during short-term occlusion. However, the research does not explore the possibility of improving the implementation of CAMShift and Kalman filter, rather it added another layer to the system to handle the long occlusion problem during object tracking.

Another research that utilises the multi-agent system in object tracking is done by Fekir and Benamrane (2015). In their research, they exploit the advantages of the multi-agent solution, snake active contour model (Kass, Witkin, & Terzopoulos, 1988) and multiresolution treatment in achieving better object tracking system. The proposed system is organised as a two-level structure where each level is composed of many cooperative agents. In edge detection step, snake agents in a higher level of image pyramid cooperate to get object boundaries in the low-resolution image. Then, it transmits its result to subsystem agents of the inferior level and these agents cooperate to refine object edge in next image, and so on. In the second step, high-level agents migrate to the next pyramid for tracking edges in the next image of the sequence, in parallel with detection step of the inferior subsystem in the previous image, using the same cooperation and scheduling strategy.

Sánchez and his colleague developed a system using multi-agent system paradigm that is capable of regulating itself dynamically taking into account specific parameters about

detection, tracking and classification. They want to reduce the computational burden as low as possible at all times without in any way compromise the reliability of the result (Sánchez, Rodríguez, De la Prieta, De Paz, & Bajo, 2015). The reason they use MAS paradigm in their development is that MAS provides the advantage of developing a complete decentralised system that consists of multiple scalable modules. On top of that, they also justify that the standardised communication protocols in MAS allow the agent modules to exchange information. The purpose of this communication between agents is to dynamically set the appropriate values for the parameters of the system, which is depending on the state of the scene.

Myint, Wong, Dooley, & Hopgood (2016) track a table tennis ball for umpiring in real-match scenarios, they presented the design of a multi-view based tracking system that can overcome the challenges of tracking a ball that travels fast and can become occluded or merged with other background objects. To tackle this problem, they proposed a multi-agent system that consists of some interconnected intelligent agents, which can either independently or jointly achieve goals. Their result shows that MAS is ideal for object tracking that facilitates shared workloads of object tracking via a network of computers and the system is scalable.

Table 2.1 shows a summary of review papers that are related to multi-agent and object tracking.

Table 2.1: Reviewed papers that use MAS for object tracking.

Author	Single / Multiple objects tracking	Occlusion handing level	Remark
Marchesotti, Piva, & Regazzoni (2003)	Multiple	Partial and Short Full	Used Agent-based architecture design to functionally combine data from a homogeneous network of sensors for tracking purposes.
Li, Wang, Qi, & Li (2010)	Single	Partial and Short Full	This research solves particle filter degradation by simulating particle using multi-agent. When used for target tracking, the tracking accuracy will not deteriorate after a period as a conventional particle filter.
Castanedo, García, Patricio, & Molina (2011)	Multiple	Partial and Short Full	Agent allows more robust and decentralised system to be designed, where management is distributed between the different camera-agents.
Baum, Izhaki, Rivlin, & Katzir (2014)	Multiple	Short and Long	Solving occlusion as two-phase problem. CAMShift and Kalman Filter were used to handle short-term occlusion
Fekir & Benamrane (2015)	Multiple	Short	The cooperative multi-agent system was implemented to create an edge detection level and an edge tracking level. Agents in the system are coordinated by a scheduling strategy
Sánchez, Rodríguez, De la Prieta, De Paz, & Bajo (2015)	Multiple	Short	Implementation of the system is faster with the standardise communication protocols in a multi-agents system that allow the agent modules to exchange information.
Myint, Wong, Dooley, & Hopgood (2016)	Single	No occlusion but complex background present	Track the ball's real-world 3D position and use it to compare with the positions of other objects such as the table and net.

In a nutshell, from the review on the past works on using MAS in object tracking, the advantages of MAS in this research is the modular design in MAS allow the object tracking system to be scalable (Myint, Wong, Dooley, & Hopgood, 2016). Agents in MAS also allow various camera view to work together to complete complicated tasks. The distributed structure allows workloads of tracking to be shared among a network of computers.

2.7.2 Role-Oriented Analysis and Design for Multi-Agent Programming (ROADMAP)

Agent-oriented software development is a relatively new software engineering approach. It is a promising approach for developing applications in the complex domain. A large number of agent-oriented methodologies have been introduced such as Gaia (Wooldridge & Ciancarini, 2001), MaSE (DeLoach, 2001), Tropos (Bresciani, Perini, Giorgini, Giunchiglia, & Mylopoulos, 2004) and Prometheus (Padgham & Winikoff, 2005). Since each methodology has its strength in different aspects, some researcher such as Juan & Sterling (2003) and Ghandehari, Saadatjoo, & Chahooki (2015) have suggested to combined the methodologies from different research to create a better software modelling method.

The ROADMAP (Role-Oriented Analysis and Design for Multi-Agent Programming) methodology, an extended work from the Gaia methodology was designed to support the development of complex open systems (Juan & Sterling, 2003). ROADMAP supports formal environment and knowledge models, and a dynamic role hierarchy to constrain the behaviour of agents in the organisation (Juan, Pearce, & Sterling, 2002). The RAP/AOR (Radical Agent-Oriented Process/Agent-Object-Relationship) methodology introduced by

Taveter & Wagner (2005) is a method to create distributed organisational information systems. It focuses on declarative models of communication and interaction. The RAP/AOR models the behaviour of an agent as founded on its perceptions and its basic mental state components: beliefs and commitments.

The combined ROADMAP and RAP/AOR methodology introduces a more unified way to engineer a multi-agent system from analysis to design and implementation and supports rapid prototyping of multi-agent systems (Sterling & Taveter, 2009). It covered all the process as needed in system specification such as role model and goal model, domain model, organisation model, interaction model, knowledge model and behaviour model. Both of these methodologies complement each other well, where the ROADMAP emphases on domain and systems analysis while the RAP/AOR is geared toward design. With the combination of the goal, role, and domain models of the ROADMAP and the models of the RAP/AOR methodology would provide the most robust support for the abstraction layer of platform-independent computational design (Sterling & Taveter, 2009).

2.7.3 Negotiation Mechanism in Object Tracking

Negotiation is a standard process that applied to the business and commerce world. In a multi-agent system, negotiation is a mechanism of decision making achieved through communication that involves a group of agents looking for a possible solution with the goal of reaching a consensus (Radu, 2013). In this study, a simple agent negotiation mechanism is applied in object tracking process. It is based on the case study of book trading done by (Bellifemine, Caire, & Greenwood, 2007).

Agent messages are the basic form of interaction among the agents. The structure of a message is a set of key values written in FIPA agent communication language (ACL). It denotes communication in terms of a function or action which is known as a communicative act (Bellifemine, Caire, & Greenwood, 2007). The FIPA-ACL message comprises a set of message parameters such as the performative, sender, receiver and content. The compulsory parameter is the performative which means the type of the communicative act of the message. The commonly used performatives in a negotiation mechanism are listed in Table 2.2 as shown by Bellifemine, Caire, & Greenwood (2007).

Table 2.2: Common performative used in a negotiation mechanism

FIPA performative	Description of action
CALL FOR PROPOSAL	Calling for proposals to execute a given action.
PROPOSE	Submitting a proposal to execute a certain action.
ACCEPT PROPOSAL	Accepting an earlier submitted proposal to execute an action.
REJECT PROPOSAL	Rejecting a proposal to do some action during a negotiation.
AGREE	Agreeing to do some action.
REFUSE	Refusing to do a given action and explaining the reason for the refusal.
INFORM	The sender informs the receiver that a given proposition is true.
REQUEST	The sender requests the receiver to do some action.

The book trading case study (Bellifemine, Caire, & Greenwood, 2007) describes the negotiation between book buyer agents and bookseller agents. The book buyer agent of the system initiates the negotiation process by sending a message which indicates a call for proposal to the bookseller agents. The book buyer agent then waits for a full set of proposals or until waiting timeouts from bookseller agents. After that, the book buyer

agent would accept the best proposal with the lowest price and reject others. The book buyer notifies the success of the book purchasing the end of the process.

2.8 Surveillance Video Benchmark Dataset

This section reviews a set of publicly available benchmark dataset that researchers could use for evaluating their tracking system. The dataset varies from the surroundings (indoor or outdoor) the moving objects (human, car or balls). Some popular video databases that have been used as a benchmark are summarised in Table 2.3.

Table 2.3: Reviewed benchmark dataset

Dataset Name	Indoor / Outdoor	Single / Multiple Camera
CAVIAR (Fisher R. B., 2005)		
INRIA Labs	Indoor (LC)	Single
Shopping centre hallway (3 viewing angle)	Indoor	Multiple (Overlap)
ETISEO (Nghiem, Bremond, Thonnat, & Valentin, 2007)		
Silogic	Outdoor	Single
INRETS (5 video angles)	Outdoor	Multiple (Overlap)
CEA	Indoor & Outdoor	Multiple (Optical and Infrared camera)
RATP	Indoor	Single
PETS 2007 (Computational Vision Group, Reading University, 2007)		
Subway entrance	Outdoor	Multiple Camera (Overlap)
VOTD (Rasid & Suandi, 2010)		
Set 1 (5 scenarios)	Outdoor (different time)	Single
Set 2 (6 Scenarios)	Indoor	Single
PETS 2009 (IEEE Computer Society, 2009)		
L1	Outdoor	M (Overlap)

The Context Aware Vision using Image-based Active Recognition (CAVIAR) project (Fisher R. B., 2005) started on October 1, 2002, and continues until September 30, 2005. The video clips in CAVIAR are created on July 11, 2003, and January 20, 2004. There are two set of videos in this benchmark dataset. The first set of the data is from a wide angle

camera lens in the entrance lobby of the INRIA Labs at Grenoble, France (Figure 2.10). The video is created to perform city centre surveillance where it is believed that many large cities have night time crime and antisocial behaviour problems.



Figure 2.10: A frame from CAVIAR entrance lobby video.

The second set of data also used a wide angle lens along and across the hallway in a shopping centre in Lisbon (Figure 2.11). The shopping mall videos were created to track mall customer behaviours in the mall. Marketers are interested in the behaviour of potential customers in a commercial setting.

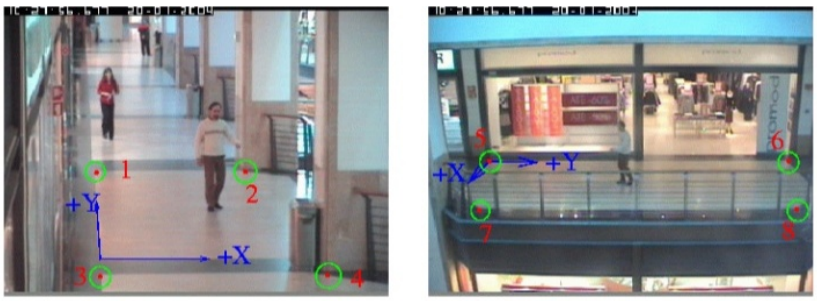


Figure 2.11: Sample frames from CAVIAR shopping mall videos.

ETISEO (Nghiem, Bremond, Thonnat, & Valentin, 2007) collects video sequences illustrating only a given problem. The video sequences were intended to exemplify the video processing problem of shadows and intensity levels variety and occluded objects.

There are four sets of video in ETISEO database namely the Silogic (airport apron surveillance video), the INRETS (building entrance and car park video), CEA (corridor and road video) and RATP (Station).

The datasets from PETS 2007 (Computational Vision Group, Reading University, 2007) are videos from multiple cameras containing the following three scenarios: loitering, attended luggage removal (theft), and unattended luggage. Because the video is captured in a subway, the video scene is always crowded. Besides, due to the setting of the camera used, at most of the time, the people video is always either partially or fully occluded.

The Versatile Object Tracking Standard Database, VOTD (Rasid & Suandi, 2010) is created to evaluate and compare the object tracking methods. The database introduced is divided into two categories; outdoor environment and indoor environment, and it covers different head population, poses, orientation, lighting conditions, etc. One limitation of the database is the small number of people that appear in the scene at one time. In contrast to PETS 2007, the most that can be observed from a scene in only five people at one time.

The PETS 2009 (IEEE Computer Society, 2009) datasets are multi-sensor sequences containing three (3) types of crowd surveillance characteristics/events within a real-world environment. The scenarios are filmed from multiple cameras and involve up to approximately forty actors. The video in this database are well planned, and the ground truths of the objects are systematically recorded. The PETS 2009 datasets are multi-sensor sequences containing three (3) types of crowd surveillance characteristics/events within a real-world environment. The scenarios are filmed from multiple cameras and involve up to approximately forty actors. More specifically, the challenge includes estimation of crowd

person count and density, tracking of the individual(s) within a crowd, and detection of flow and crowd events as shown in Figure 2.12.

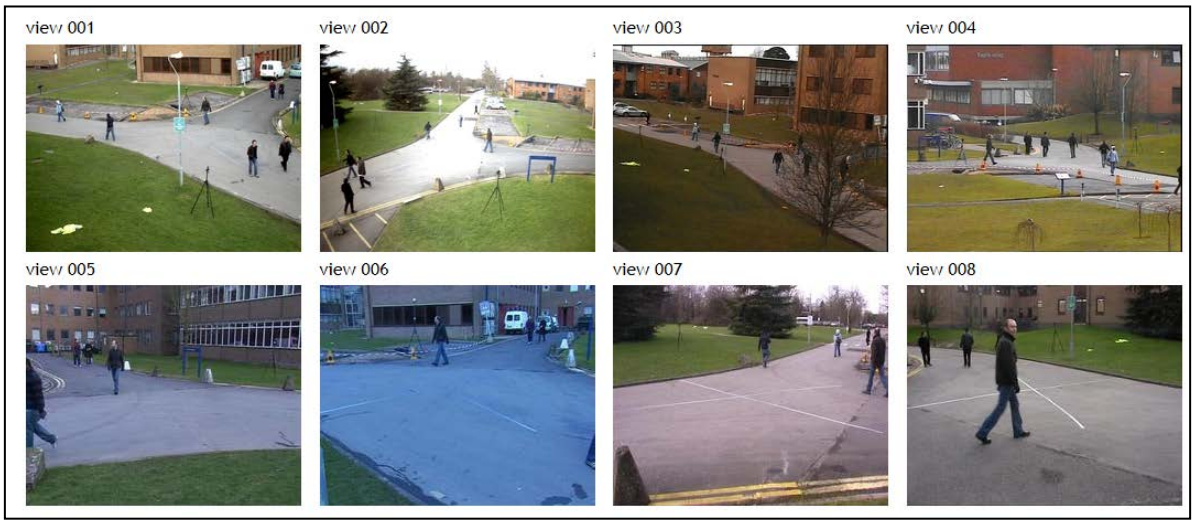


Figure 2.12: Sample images from PETS 2009 video dataset.

The dataset consists of four packages (S0 to S3) and the contents of each package are organised in the following order:

- S0: Training Data: contains sets background, city centre, regular flow
- S1: Person Count and Density Estimation: contains sets L1, L2, L3
- S2: People Tracking: contains sets L1, L2, L3
- S3: Flow Analysis and Event Recognition: contains sets Event Recognition and Multiple Flow

The video in PETS 2009 come with the camera set up information in the XML file. For example, for view001 in S1 video, the camera setup is the number of frame buffer pixels (*Nfx*), the number of Charge-coupled Device (CCD) sensors in a row (*Ncx*), the distance between CCD sensors in a row (*dx*), the distance between CCD sensors in a column (*dy*) and finally the horizontal uncertainty scale factor (*Sx*). This information is useful to

convert the ground truth data in XML to row and column data in the video frame. An example of the ground truth data is shown in Figure 2.13.

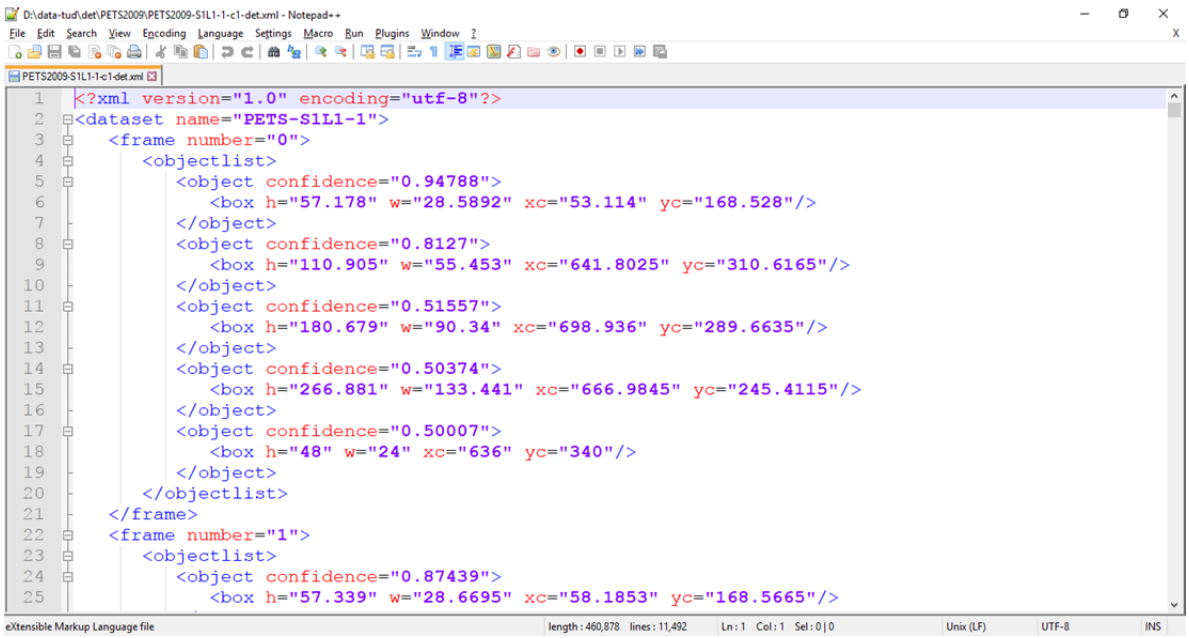


Figure 2.13: Ground truth data of PETS 2009 S1L1 view 001.

Other than benchmark databases mentioned earlier that use real-life footage captured using surveillance camera, there are also simulated video footages that have been created to evaluate the performance of object tracking algorithm. Taylor, Chosak, & Brewe (2007) has created a simulated surveillance video called Object Video Virtual Video (OVVV) that is based on the game Half-Life 2 from Valve Software. Taylor, Chosak, & Brewe (2007) claimed those simulated videos are ideal to provide a good indication of which algorithms work well in a given scenario. To support performance evaluation, OVVV generates detailed automatic ground truth for each frame including target centroids, bounding boxes and pixel-wise foreground segmentation. The limitation of OVVV dataset is that not much research has used the videos for benchmarking.

After reviewing the benchmark datasets, this research proposed to use the CAVIAR lobby video set and PETS 2009 view001 L1. In the CAVIAR video, a person is walking from left bottom corner to top right corner of the scene. The person is not occluded in the video until he walked out the video scene. The video is suitable to test how well an object tracking can track a real-world object without occlusion.

PETS 2009 view001 S1 video is also suitable for testing an object tracking system. The S1 video set is recorded for person counting; therefore, it is also suitable to test an object tracking system for tracking object under occlusion. The view001 in S1 have many people walking on the road in different directions, and objects in the video are partially and fully occluded. Therefore it is suitable to test the proposed multi CAMShift agent object tracking system. Tracking method such as Conte, Foggia, Percannella, & Vento (2010) work, used PETS 2009 database as their benchmark database. Their result can be used to compare with the result obtained by this research later.

2.9 Measurement of Detection Accuracy

The problem of visual tracking evaluation is sporting a large variety of performance measures and mainly suffers from lack of consensus about which measures should be used in experiments (Čehovin, Leonardis, & Kristan, 2016). This makes the cross-paper tracker comparison difficult. Therefore, it is essential to use renowned performance measures for this research so that the result could be the use to benchmark with others.

To evaluate the performance of an object tracking system, the most basic evaluation is to use the precision and recall measures (Godil, Bostelman, Shackleford, Hong, & Shneier, 2014). In the context of object tracking, precision and recall measures can be defined as:

$$\text{Precision} = \frac{\text{No. of correct Correspondences}}{\text{No.of established correspondences}}, \quad (5.1)$$

$$\text{Recall} = \frac{\text{No. of correct Correspondences}}{\text{No.of actual correspondences}}, \quad (5.2)$$

The *No. of actual correspondences* denotes the correspondences available in the ground truth. Precision and recall are known as the most basic evaluation method because it did not consider the location where the object is detected.

For instance, when using an object tracking system to track a target that appears in 50 frames in a video, if the system manages to discover the object in 40 frames, then the recall rate of the system is 40/50 which is 80%. Out of the 40 frame, if the location of the tracked target (e.g., centroid of the object) is correct for 20 frames, then the precision rate of the system is 20/40, which is 50%.

Another method used to evaluate tracking result is to compare the centroid of objects detected and ground truth in the graph as used by Guo, Li, Zhao, & Zhao, 2016. In their work, they show tracking result of a pedestrian using position changing curves of the pedestrian centroid in a vertical and horizontal direction. Ts method allows the researchers to compare spatial alignment of the centroid of the system output and ground truth only, the boundary of the object detected is not considered in this method.

Based on the literature review conducted by Čehovin, Leonardis, & Kristan (2016), the most famous measures for multi-target tracking performance is using the Multiple Object Tracking Precision (MOTP) proposed by Kasturi, et al. (2009) work. Kasturi, et al. (2009) create a set of summative metrics that address the important types of errors such as misses, false alarms, spatial mismatches, and tracking ID switches.

Kasturi, et al. (2009) work proposed the evaluation methods for object tracking system that provide more details on the tracking result, the Sequence Frame Detection Accuracy (SFDA) as denoted in equation (2.1) and the Frame Detection Accuracy as expressed in equation (2.2).

$$SFDA = \frac{\sum_{t=1}^{t=Nframce} FDA(t)}{\sum_{t=1}^{t=Nframce} \exists(N_G^{(t)} \; or \; N_D^{(t)})} \tag{2.1}$$

$$FDA(t) = \frac{Overlap\_Ratio}{\left\lceil \frac{N_G^{(t)} + N_D^{(t)}}{2} \right\rceil} \tag{2.2}$$

Where, $Overlap\_Ratio = \sum_{i=1}^{N_{mapped}^{(t)}} \frac{\left|G_i^{(t)} \cap D_i^{(t)}\right|}{\left|G_i^{(t)} \cup D_i^{(t)}\right|}$

Unlike precision and recall, SFDA can measure more aspect of tracking accuracy. It measures the number of objects detected and missed detection, false positives and spatial alignment of the system output and ground truth object. The measurement requires calculation of the overlap ratio. To calculate the result for both mentioned measurements, the ground truth must be generated. In most cases, ground truth can be generated manually or by performing background subtraction if the background is simple.

SFDA proposed by Kasturi, et al. (2009) has been cited heavily since it was introduced . It is still highly accepted by current research such as Shu, Dehghan, Oreifej, Hand, & Shah (2012), Meng, Yuan, Tan, & Wang (2015), Li & Li (2016) and Wu, Zhong, Tsao, Yang, Chen, & Chien (2017). The SFDA obtained by Conte, Foggia, Percannella, & Vento (2010) on PETS 2009 S2-L1 video sequences is 0.505.

Therefore, this research will use SFDA as a measurement to test the proposed object tracking system so that the result obtained from the experiment can benchmark with many previous works. SFDA also measure more aspect than precision and recall. This research will also use position changing curves of the pedestrian centroid in a vertical and horizontal direction to compare the changes of position of the target object and ground truth.

2.10 Summary

In this chapter, state of the art on occlusion handling in object tracking and its related studies are reviewed. The literature reviews found that object tracking methods are categorised into three main categories of object tracking methods namely the point tracking methods the silhouette tracking methods, and the kernel based tracking methods. The point base tracking is getting less attention in object tracking while silhouette based tracking has too much data to process. The literature review shows that many recent object tracking projects are using CAMShift. CAMShift is a kernel-based tracking, and that has the advantage of using an updatable colour histogram to track an object from frame to frame. CAMShift also uses the resizable kernel when tracking the object making.

However, there is a limitation on CAMShift tracker, when using CAMShift tracker to track a target object, the tracker will fail to track the target when the object is occluded by another object that has close appearance similarity. During the event of occlusion, the tracker will include the occluding object appearance information in the colour histogram update process in the CAMShift. As a result, the tracker will track the wrong object after the occlusion. Through literature review and preliminary experiments conducted in this research, it is found that CAMShift requires accurate setting for the inclusive lower

boundary array and the inclusive upper boundary array during the process of generating the colour histogram. These values are critical to achieve proper segmentation of target object and improve the tracking result, especially during partial occlusion.

This aspect of CAMShift in object tracking has not been addressed in previous research. Therefore, this research studies the requirement to design a system that can select the optimum values for the inclusive lower boundary array and the inclusive upper boundary array.

To design a system to look for an autonomous dynamic setting for the CAMShift object tracker, it is essential to choose a software engineering method that supports simplification of dynamic interaction component. The literature review suggests that the multi-agent software engineering approach is suitable for designing the system since recent research has claimed that multi-agent can meet the requirement of building a system that has high urgency, medium to high dynamism, and medium to large scale. Therefore, this research studies how to model the CAMShift tracker using the multi-agent software engineering approach so that the optimum values for the inclusive lower boundary array and the inclusive upper boundary array can be selected.

Another limitation on CAMShift tracker is on tracking object during full occlusion. During full occlusion, the entire target object is occluded from the camera hence no appearance data is available for tracking. Therefore, the only solution is to predict the location of the target object based on motion model that subject to motion noise. Motion noise existed when the object do not move exactly as calculated based on the motion model due many resaons such as the object has to avoid occlusion when moving or the terrain of the object travel are not consistence. To cater the motion noise, the review

suggests that a motion estimator need to be added to the object tracker. As a result, the research also studies method to integrate Kalman filter motion estimator with the CAMShift tracker in the multi-agent environment. In the next chapter, the design of MAS for the proposed solution will be presented.

CHAPTER 3
MULTI CAMSHIFT AGENT FRAMEWORK FOR OCCLUSION HANDLING IN OBJECT TRACKING

3.1 Introduction

Generally, each agent deliberatively makes decisions to carry out the system tasks coherently with other agents, considering both the information generated in its local process and the information available in the network. An agent only knows some external events and decides with its partial knowledge. Consequently, the quality of the decision made by an agent could be considered not optimal. A distributed solution could become an alternative depending on the scalability and fault-tolerance requirements. Distribution could be achieved through a multi-agent system. In this research, a CAMShift tracker with unique parameter settings would be represented by a single software agent. It is proposed to create the cooperation between the CAMShift agents to improve the local information through message exchange. This research would like to show that with the coordination among the agents could improve the quality of decision made and bringing it close to optimum even each agent just with partial knowledge. The primary purpose of using multi-agents approach is to improve the recognition and interpretation process (object size, location and kinematics) of mobile targets through cooperation correctly and therefore could to improve the surveillance performance of the tracking system generally.

In this chapter, the components of the multi-agent framework architecture for designing surveillance systems are described. The design of the multi-agent framework is built based on the combined ROADMAP (Role-oriented Analysis and Design for Multi-Agent Programming) and RAP/AOR (Radical Agent-Oriented Process/Agent-Object-

Relationship) methodology. It includes the Computation Independent Models (CIMs) and the Platform Independent Models (PIMs) for object tracking.

In the CIMs, the requirements of the system are analysed using goal model, role model, organisation model, domain model and interaction model. Goal model represents the purposes applied to the system as a whole while role model represents the duties that need to be exercised by enactors of individual roles to achieve the goals (Sterling & Taveter, 2009). Relevant functional and quality requirements are described when refining goal and role diagrams. The arrangement of the roles is shown in the organisation model. The domain entities and roles, and relationships between them are denoted using domain model. After that, the roles in the object tracking are mapped onto agent types using agent-role coupling diagram. The interactions between enactors of the roles are then sketched using interaction-frame diagrams. In the PIMs, the domain model is later transformed into knowledge model. This chapter also described the behaviour model to complete the design of the multi-agent framework at the end of the chapter.

3.2 Goal Model

The goal model outlines a high-level overview of the system requirements. The main objective of goal model is to enable both domain experts and developers to identify the goals of the system. A goal is associated with a specific role which indicates the actor that is involved in accomplishing the goal. Figure 3.1 reflects the goal model for the development of object tracking. The overall goal of the system is to track an object, expressed by the overall goal “Track object”.

As shown in Figure 3.1, two roles named as User and Object tracking manager, are indicated as relevant to the overall goal. The object tracking manager role is responsible for achieving this goal. The quality goal “Correct object tracked” indicates that the output of object tracked should be accurate. The goal “Track object” can be achieved via sub-goals which are known as “Track targeted object location” and “Evaluate tracking result”.

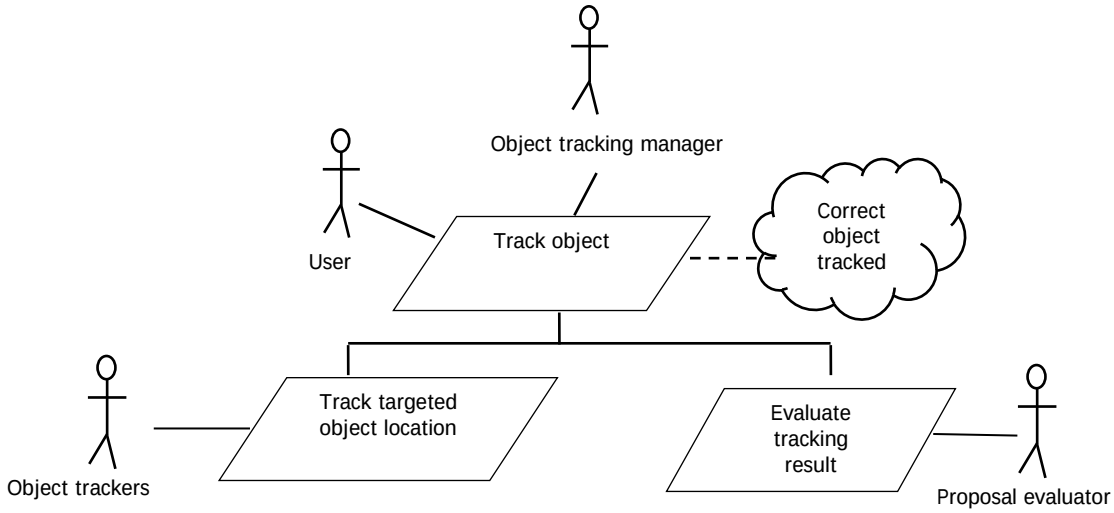


Figure 3.1: Goal model for object tracking

3.3 Role Model

Roles are determined by the goals. A role model represents a role schema which consists of a role name, role description, responsibilities and constraints. A role can be played by a human or software agent (Luo, Sterling, & Taveter, 2007). A role is required to perform common tasks and subtask under it. It lists the duties of an agent and the constraints that applied to the agent when carries out the duties. Table 3.1, Table 3.2, Table 3.3 and

Table 3.4 show the role models for *User*, *Object tracking manager*, Object *tracker* and *Proposal Evaluator*.

Table 3.1: Role model for user

Role name	User
Description	Request to track the object.
Responsibilities	• Provide target object. • Provide video sources.
Constraints	Target object and video sources must be available.

Table 3.2: Role model for object tracking manager

Role name	Object tracking manager
Description	Manage object tracking process.
Responsibilities	• Acquires video frames. • Set and start the tracking process. • Call for Proposal / Send request to track the object. • Receive proposal / Receive tracked object. • Accept the best proposal, so an occluded object could be recognised when it reappears in the video scenes (to handle long occlusion). • Terminate the tracking process.
Constraints	Video frames must be available.

Table 3.3: Role model for object tracker

Role name	Object tracker
Description	Track targeted object location.
Responsibilities	• Remove the noise from video frames. • Remove the shadow from video frames. • Check if the object is occluded • If not occluded, track the object in video frames using CAMShift • If occluded, track the object with Kalman filter • Send track result to proposal evaluator. • Send evaluation result to object tracking manager.
Constraints	Video frames must be available.

Table 3.4: Role model for proposal evaluator

Role name	Proposal evaluator
Description	Determine the quality of the tracking result based on the smoothness of tracking result and variance of track object size in track result.
Responsibilities	• Evaluate the proposed track result. • Send evaluation result to object tracker.
Constraints	Track result must be available.

3.4 Organisation Model

The organisation model shows the arrangement of the roles involved in task achievement. Figure 3.2 illustrates an organisation model for object tracking. The organisation model as shown in Figure 3.2 is arranged with the object tracking manager coordinates the interactions among the roles played. The user and object tracking manager rely on each other when performing the task. The user relies on the object tracking manager for object tracking while the object tracking manager relies on the user to receive the request for object tracking and provide the video sources.

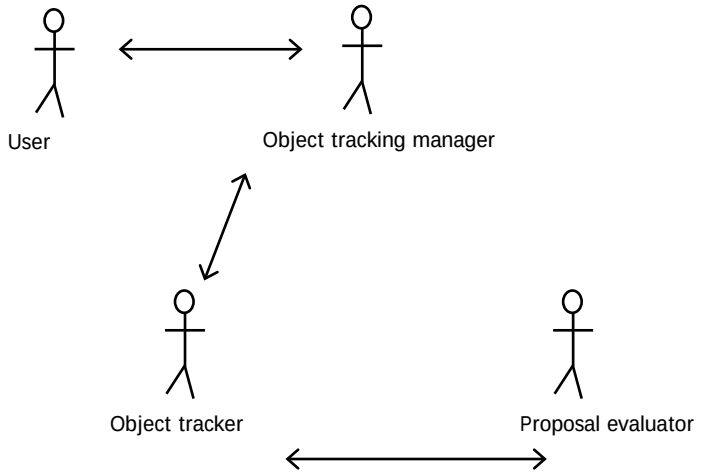


Figure 3.2: Organisation model for object tracking

3.5 Domain Model

The domain model represents the knowledge required to achieve the goals by decomposing the knowledge into modular entities and specifying their relationships (Lu, 2013). Figure 3.3 shows the domain model for object tracking.

The target object and surveillance video are domain entities represent the object and video provided by the user when requesting for object tracking. Both of these inputs are used by object tracking manager to find the initial location of the tracking object. Tracking results are proposed by object trackers using the information of the initial location of the tracking object obtained from object tracking manager.

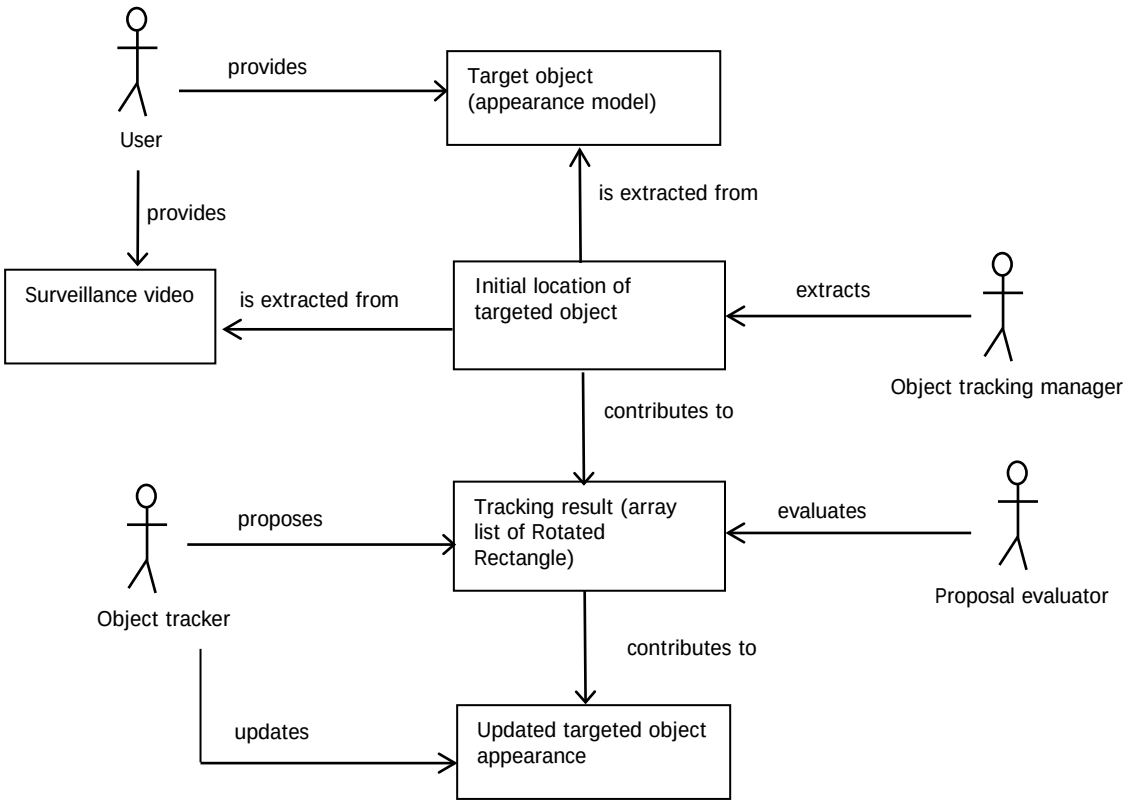


Figure 3.3: The domain model for object tracking

Each object tracker can use different tracking algorithms. However, the CAMShift tracker is chosen as the object tracker in the implementation of this study as it is a widely used tracker which could provide a promising result (Coşkun & Ünal, 2016). Object trackers with different parameter settings find the location of the targeted object in the

video frames and update the target object appearance until the end of the video frames. The tracking result will be sent to proposal evaluator for tracking result evaluation.

3.6 Agent-Role Coupling

Agent-role coupling shows how the responsibilities of roles in the object tracking multi-agent system have been mapped into agents. It helps to check the role assignments against the fulfilment of goals. Based on Prometheus methodology, a role might have to be mapped to different agent types (Sterling & Taveter, 2009). Several agents could work together to achieve the responsibilities of one role. Sometimes, one agent can play several roles too. The roles for object tracking system agents are shown in Figure 3.4. The agent types are modelled by the corresponding agent descriptor as shown in Table 3.5, Table 3.6 and Table 3.7.

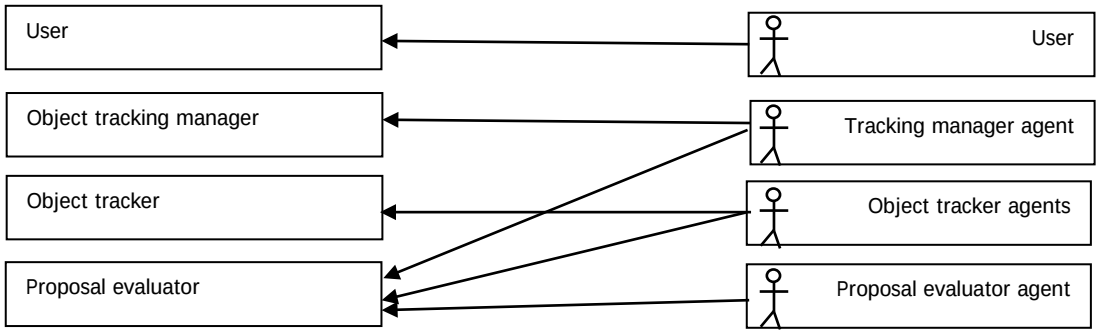


Figure 3.4: The agent-role coupling diagram for object tracking

Table 3.5: Descriptor of the type Tracking manager agent

	Tracking manager agent
Description	An agent of the Object tracking manager type receives a request from the user, call for tracking proposal and accepts the best proposed tracking result.
Goal achieved	Handle tracking process.
Creation	Initialisation.
Destruction	The best proposed track result has been received.

Table 3.6: Descriptor of the type Object tracker agent

	Object tracker agent
Description	An agent of the Object tracker agent type receives a request from Tracking manager agent and sends proposed and evaluated tracking result.
Goal achieved	Track object location.
Creation	Call proposal of tracking object is received.
Destruction	The proposed tracking result has been evaluated and sent back to the tracking manager agent.

Table 3.7: Descriptor of the type Proposal evaluator agent

	Proposal evaluator agent
Description	An agent of the Proposal evaluator agent type receives proposed tracking result from Object tracker agent and evaluates the result.
Goal achieved	Evaluate proposed tracking result.
Creation	Proposed tracking result is received.
Destruction	The proposed tracking result has been evaluated and sent back to the object tracker agent.

3.7 Interaction Model

The interactions between agents could be displayed with interaction model after defining the agent types. Interaction models represent interaction patterns between agents in details based on their responsibilities defined for the corresponding roles (Sterling & Taveter, 2009). The interaction model is represented with an interaction-frame diagram. It models the possible interactions between two or more types of agents and all interaction alternatives for both physical interactions and communication.

The interaction-frame diagram for object tracking is shown in Figure 3.5. It comprises three interaction frames: between User and the tracking manager agent, between tracking manager agent and object tracker agents, and between object tracker agents and proposal evaluator. The order of the possible interactions is represented in the interaction-sequence diagram.

An interaction-sequence diagram models a sequence of action events performed and received by agents (Sterling & Taveter, 2009). Each action event is characterised by a number which denotes the order of interaction between the agents involved. The interaction-sequence diagram for object tracking is depicted in Figure 3.6. The different between Figure 3.5 and Figure 3.6 is that the Figure 3.6 labels every message between the agents to explicitly represent the order of interactions. (Sterling & Taveter, 2009).

Figure 3.6 shows that the model starts with the request from User by providing target object and video sources. Object tracking manager agent would request Object tracker agents to track the object in the video provided. A message would be sent to proposal evaluator agent for proposed tracking result evaluation when the task is done. Proposal

evaluator agent would evaluate the proposed tracking result provided by each object tracker agent. The proposal evaluator agent would send a message back to the object tracker agents once the proposed result has been evaluated. The proposed tracking result and evaluation result would be sent back to object tracking manager agent by object tracker agents. Lastly, the object tracking manager agent would choose the best proposed tracking result based on the evaluation result and inform the user about the termination of the process to retrieve the result.

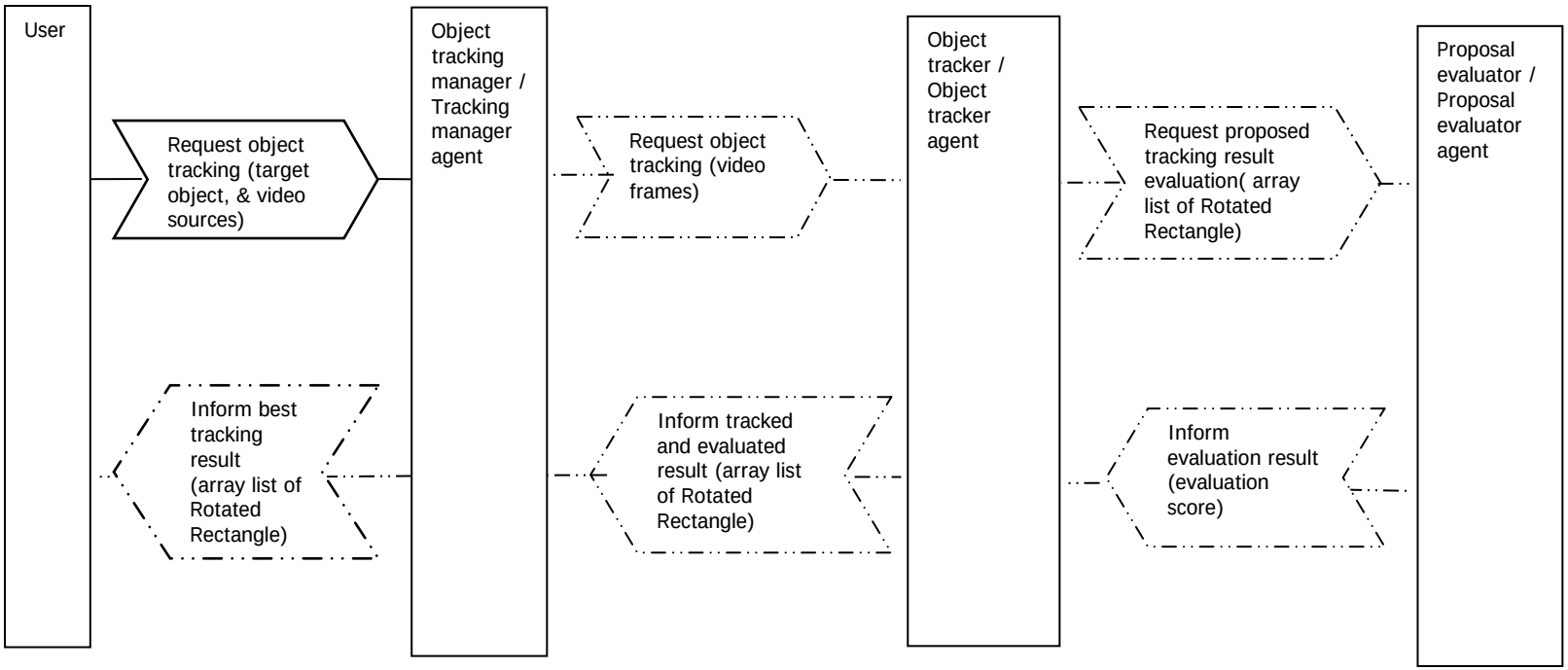


Figure 3.5: Interaction-frame diagram for object tracking

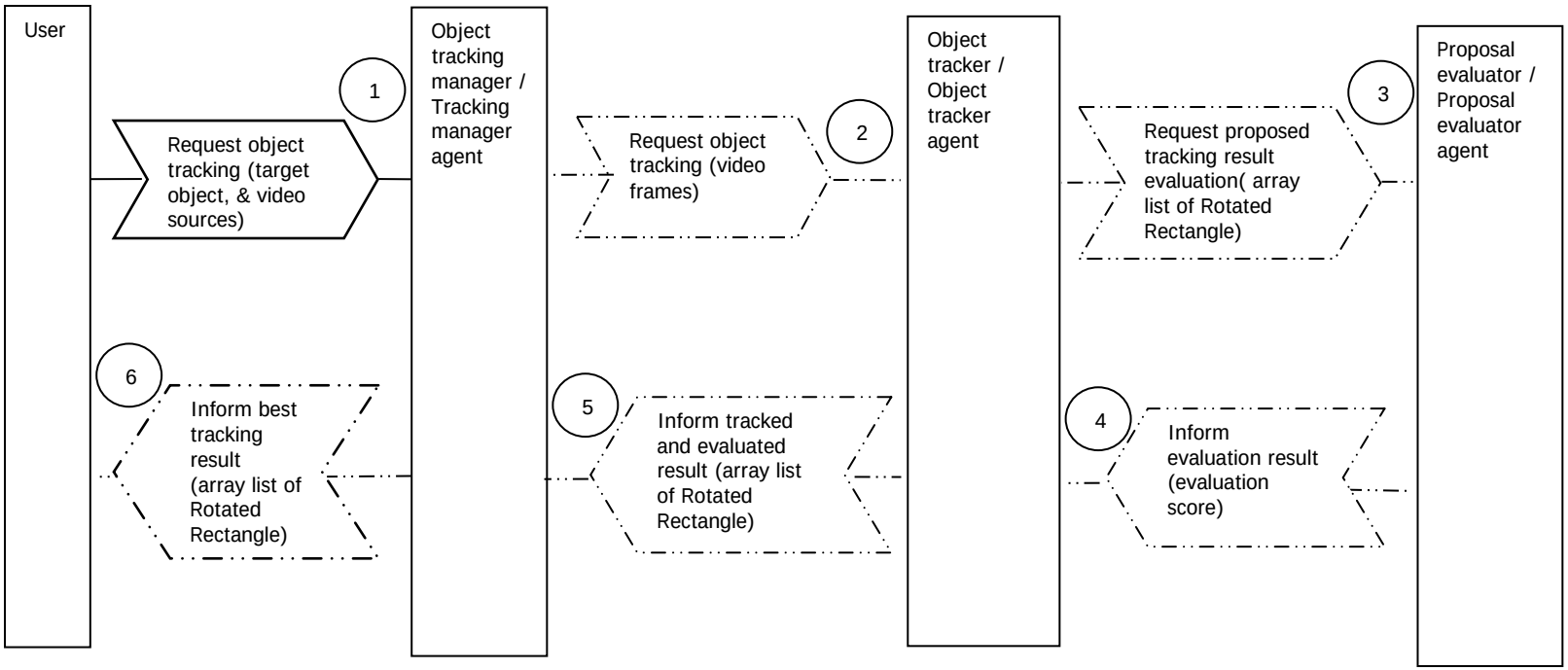


Figure 3.6: Interaction-sequence diagram for object tracking

3.8 Knowledge Model

The framework of knowledge about the agents and the objects within the problem domain is generated in knowledge model (Sterling & Taveter, 2009). The knowledge model extends the domain model by interpolating the entities with attributes and their relationships which provide a common knowledge to be shared among agents. The knowledge model for object tracking is illustrated in Figure 3.7.

The tasks of object tracking are allocated among three types of the agent with own defined responsibility. The agents are known as Tracking manager agent, Object tracker agent and proposal evaluator agent. The Tracking manager agent coordinates various tasks that need to be performed between the user and the agents. The Tracking manager agent manages the user interaction. Besides that, the initial location of the targeted object will be prepared by Tracking manager agent and sent to Object tracker agents.

Each Object tracker agent tracks the targeted object through the video frames with different parameter settings to prepare the tracking result. During the tracking process, the Object tracker agents update the appearance of the targeted object. After reaching the end of the video frames, the tracking result will be sent to the Proposal evaluator agent. The Proposal evaluator agent evaluates the tracking results from each Object tracker agent. The evaluation score for each tracking result will be sent to the Tracking manager agent. The Tracking manager agent will select the best tracking result with the highest score.

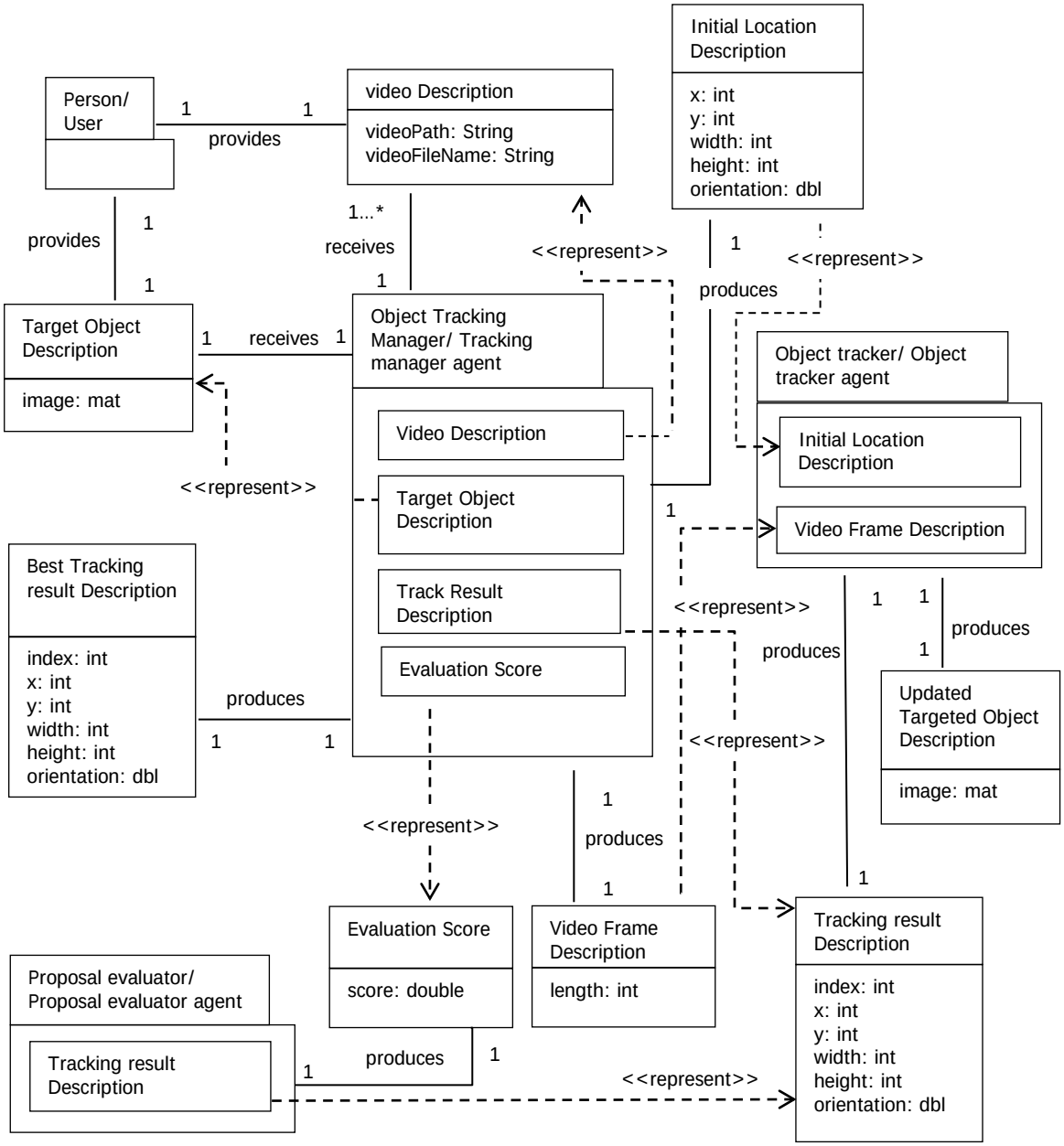


Figure 3.7: Knowledge model for object tracking multi-agents system

3.9 Behaviour Model

Behaviour model denotes the behaviour of an agent making decisions and performing activities (Sterling & Taveter, 2009). It is firstly addressed by scenarios presenting how the multi-agent system achieves the goals set can be accomplished by the defined agents. A

scenario is described as collective activities that model how a particular goal is accomplished by agents performing their particular roles. Then, what individual agents do is modelled and controlled in terms of rules.

The overall process for tracking object is shown in two scenarios. The first scenario for managing object tracking is given in Table 3.8. It consists of two steps. Its first step starts with the request from a user by providing the resources of the target object and video source for tracking. Its second step is the notification of the completeness of the object tracking with the best tracking result by the Tracking manager agent to the User at the end of the process.

Table 3.8: A scenario for managing object tracking

<table>
<tr><td colspan="6">Scenario 1</td></tr>
<tr><td>Goal</td><td colspan="5">Managing object tracking</td></tr>
<tr><td>Initiator</td><td colspan="5">Tracking manager agent</td></tr>
<tr><td>Trigger</td><td colspan="5">The user sends a request of object tracking</td></tr>
<tr><td>Failure</td><td colspan="5">-</td></tr>
<tr><td colspan="6">Description</td></tr>
<tr><td>Condition</td><td>Step</td><td>Activity</td><td>Agent types and roles</td><td>Resources</td><td>Quality goals</td></tr>
<tr><td rowspan="2">When a user requests to track the object.</td><td>1</td><td>Track object (Scenario 2)</td><td>User, Tracking manager agent</td><td>Target object, video source</td><td rowspan="2">Correct object tracked</td></tr>
<tr><td>2</td><td>Inform the best tracking result</td><td>Tracking manager agent, User</td><td>Best tracking result</td></tr>
</table>

The second scenario corresponds to the goal of “Track object” as modelled in Figure 3.1. It consists of six steps. It realises the sub-goal of “Track targeted object location” and “Evaluate tracking result” as displayed in Table 3.9. The second scenario begins with the call of the proposal to track the targeted object location by the Tracking manager agent to the Object tracker agents. The Tracking manager agent provides the resources of the initial location of the targeted object and the video frames used. The second step is updating the appearance of the targeted object until the end of the video frames by the Object tracker agent. The third step is the evaluation of the proposed tracking result by the Proposal evaluator agent. The evaluation score for each proposed tracking result is sent back to each Object tracker agent during the fourth step. The fifth step is the notification of the proposed tracking result and the evaluation result by the Object tracker agents to the Tracking manager agent. The last step is the selection of best tracking result done by the Tracking manager agent.

An individual agent’s behaviour is determined by a controller according to the agent’s perceptions and knowledge. The controller can be modelled by rules. The rules constructed for tracking object are presented in Table 3.10 by showing each rule with the activity is triggered, the condition and the action prescribed by the rule.

Table 3.9: A scenario for tracking object

Scenario 2					
Goal	Track object				
Initiator	Tracking manager agent				
Trigger	A request to track object is received				
Failure	Produce incorrect tracked object				
Description					
Condition	Step	Activity	Agent types and roles	Resources	Quality goals
When a request to track object has been received.	1	Track targeted object	Tracking manager agent, Object tracker agent	Initial location of targeted object, video frames	Correct object tracked
	2	Update appearance of a targeted object	Object tracker agent	Targeted object appearance	
	3	Evaluate proposed tracking result	Object tracker agent, Proposal evaluator agent	Proposed tracking result	
	4	Inform evaluated result	Proposal evaluator agent, Object tracker agent	Evaluation result	
	5	Inform proposed tracking result and evaluated the result	Object tracker agent, Tracking manager agent	Proposed tracking result and the evaluation result	
	6	Choose the best tracking result	Tracking manager agent, User	Best tracking result	

Table 3.10: The models of R1 until R6 for tracking object

Rule	Trigger	Condition	Action if condition true	Action if condition false
R1	Start event	User's request is accepted	Start "Track object" activity	-
R2	Start "Track object" activity	There exist a target object and video source	Start calling for proposal "Track targeted object location" activity	-
R3	Start "Track targeted object location" activity	Not yet reached the end of video frames	Start "Update targeted object appearance" activity	-
R4	End "Update targeted object appearance" activity	There exist tracking results	Start "Evaluate proposed tracking result" activity	-
R5	End "Evaluate proposed tracking result" activity	There exists evaluation result	Inform evaluation result	-
R6	End "Track targeted object location" activity	-	Inform proposed tracking result and evaluation result	-
R7	End "Track object" activity	Tracking is done	Choose best tracking result and inform the best tracking result	-

The tracking process is triggered by the start event and is activated if a request is received from the User as modelled with rule R1 in Figure 3.8. An activity type of "Track object" is started by rule R2. The rule R2 is triggered only if there are a target object and video source. When it is triggered, this rule will start an activity type of "Call for proposal to track targeted object location". After that receiving the call for proposal, an activity type of "Track targeted object location" is started by rule R3.

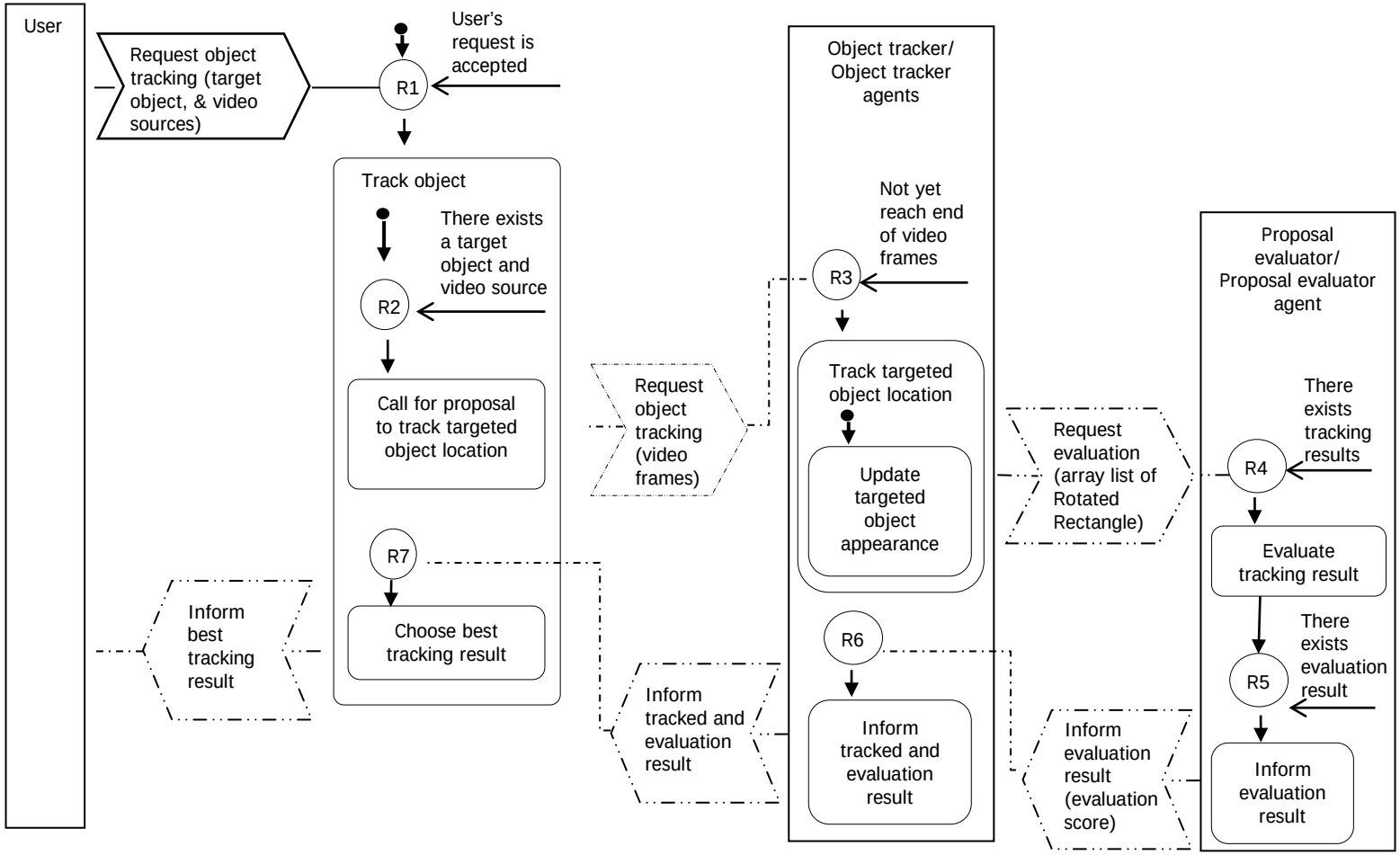


Figure 3.8: Behaviour model for object tracking multi-agent system

The activity type of “Track targeted object location” has a sub-activity type of “Update targeted object appearance”. The rule R3 is triggered until it reaches the end of the video frames. It is then followed by an activity type of “Evaluate proposed tracking result” which is modelled by rule R4. When the evaluation is completed, an activity type of “Inform evaluation result” is started by rule R5.

The process is continued with an activity type of “Inform proposed tracking result and evaluation result” as modelled by rule R6. Rule R7 is triggered at the end of the process. It starts an activity type of “Choose best tracking result” and informs the user the best track result.

3.10 Summary

This chapter discusses the components of the multi-agent framework architecture for designing object tracking in surveillance systems. It includes the independent computation models which consist of goal model, role model, organisation model and interaction model of the multi-agent system for object tracking. Besides that, it also further discusses the framework with the platform independent models which involve agent knowledge model and agent behaviour model. The next chapter will describe the platform specific model for object tracking which concentrates on the implementation of the proposed framework.

CHAPTER 4
IMPLEMENTATION

4.1 Introduction

In this chapter, the Platform Specific Models (PSMs) for object tracking is presented. It discusses the details of how the design of the object tracking is implemented using JADE (Java Agent Development Framework). The design model a presented in Chapter 3 is implemented based on the ROADMAP guideline proposed by (Sterling & Taveter, 2009).

JADE (Java Agent Development Framework) is an open source middleware to build multi-agent system in Java programming. It complies with the FIPA specifications and provides a set of graphical tools to support the development of the multi-agent system. A JADE platform consists of agent containers where agents live. The main container is the first container to be launched and other containers join it by registering with it.

The main container facilitates special agents such as Agent Management System (AMS) and Directory Facilitator (DF). The AMS provides the naming service. It ensures that each agent in the platform has a unique name and represents the authority in the platform. The DF provides a yellow pages service where one agent could search for other agents required when achieving its goal. Other important tools available that could assist the platform administration and application development includes the Remote Management Agent (RMA) and Sniffer. The RMA is a graphical console for management and control of the platform. The Sniffer could help debugging agent societies by showing how the messages from the agents are exchanged.

A user-defined JADE agent is implemented as an instance of the Java class named jade.core.Agent. It can perform several tasks simultaneously in response to different events. The behaviours of a JADE agent are implemented as instances of the Java class named jade.core.behaviours.Behaviour. The implementation of the proposed multi-agent system is based on the negotiation mechanism. Section 4.2 presents the concept of negotiation mechanism employed in object tracking. Section 4.3 discusses the implementation of JADE based Tracking manager agent while Section 4.4 discusses the implementation of JADE based Object tracker agent using CAMShift tracking algorithm. A simulation of object tracking system is presented in Section 4.5.

4.2 Negotiation Mechanism in Object Tracking

In this study, the tracking manager agent would act like the book buyer agent who initiates the negotiation process. The tracking manager agent sends a call for to the object tracker agents who act like the bookseller agent. The object tracker agents propose their tracking result and the tracking evaluation back to the tracking manager agent. The tracking manager agent decides the best tracking result by accepting the proposal with the highest score of result evaluation and rejects other proposals. The tracking manager agent announces the result to the user at the end of the process.

4.3 Implementation of JADE Based Tracking Manager Agent

The Tracking manager agent as designed in Chapter 3 is implemented using a JADE agent class named as “VideoAgent”. The “VideoAgent” of Tracking Manager agent type performs activities as listed below:

i. Let user chooses which object to track

ii. Look for the object tracker agents from Directory Facilitator (DF) agent

iii. Call for proposal to the object tracker agents

iv. Determine which object tracker agent provides the best object tracking result

The “VideoAgent” begins with creating a graphical user interface (GUI) using the JavaFX package as shown in Figure 4.1. Using the GUI, the user can select an image file of the target object for tracking from the computer by clicking on the “Load Target” button. The object for tracking is an image file name “target.png”. The file can be created earlier using a video player with snapshot features like VLC Video Player and a picture editing software like Microsoft Paint. Using the snapshot feature, the first frame of the video where the targeted object appears will be captured and save as an image file “frame01.jpg”. Then, using a mouse, the target in the “frame01.jpg” can be selected, and the target can be cropped from the video frame and save as a new file name “target.png”.

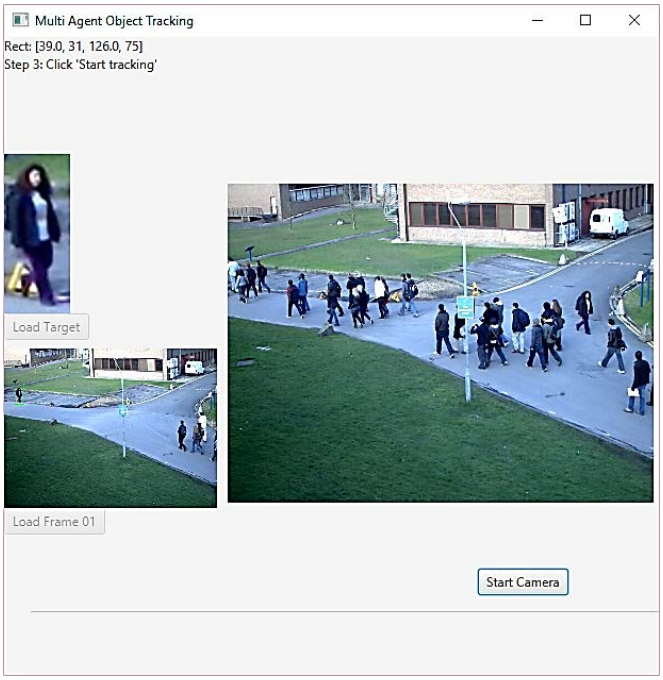


Figure 4.1: “VideoAgent” graphical user interface

After “target.png” is loaded, the next step is to load the video file into the system. Using the button load “Load frame 01”, the video file will be loaded into the system as an array of matrix. Each cell in the matrix is a data structure that stores pixel information. After the file is loaded, the system will use the “target.png” to locate the object in the frame 01. The kernel or the bounding box of the target in frame 01 will be determined automatically by comparing every pixel in the “target.png” (T) with the pixels in the frame 01 (I) using the equation (4.1).

$$R(x,y) = \sum_{x',y'}(T(x',y') - I\,(x+x', y+y'))^2 \qquad (4.1)$$

From the comparison, we will get another matrix (R) that has the similarity score of every pixel that match (are similar) to the template image. Then using R, the value the bounding box is defined as [x, y, width, height] where x and y is the coordinate of the top left corner of the bounding box which is also the coordinate of the cell in R that store the highest value.

After the bounding box, which is the kernel for CAMShift tracker is found, the Tracking manager agent (VideoAgent) will be allowed to start by clicking the button “Start Camera”. Otherwise, the tracking shall not begin. The extracted implementation code of the “VideoAgent” is shown in Figure 4.2.

When the “VideoAgent” begins, it is required to search for the object tracker agents from DF agent. The extracted implementation code looking for object tracker agents is shown in Figure 4.3.

```
1.          /**
2.           *
3.           */
4.          package lby.org.agentTracking;
5.
6.          import java.io.IOException;
7.          import java.io.Serializable;
8.          import java.util.List;
9.
10.         import org.opencv.core.Mat;
11.
12.         import jade.core.AID;
13.         import jade.core.Agent;
14.         import jade.core.behaviours.*;
15.         import jade.domain.DFService;
16.         import jade.domain.FIPAException;
17.         import jade.domain.FIPAAgentManagement.DFAgentDescription;
18.         import jade.domain.FIPAAgentManagement.ServiceDescription;
19.         import jade.lang.acl.ACLMessage;
20.         import jade.lang.acl.MessageTemplate;
21.         /**
22.          * @author bengyong
23.          *
24.          */
25.         public class VideoAgent extends Agent
26.         {
27.               private static final long serialVersionUID = 1L;
28.               private ObjRecognition gui = new ObjRecognition();
29.               protected void setup()
30.             {
31.
32.                 gui = new ObjRecognition();
33.                 new Thread(gui).start();
34.
35.                 //javafx.application.Application.launch(ObjRecognition.class);
36.                 System.out.println("JavaFx executed");
37.
38.                 addBehaviour( new beginTracking( this, 10000 ) );
39.             }
40.
41.               // Put agent clean-up operations here
42.               protected void takeDown() {
43.                       // Printout a dismissal message
44.                       System.out.println("Manager-agent "+getAID().getName()+"
45.         terminating.");
46.               }
47.
48.               …
```

Figure 4.2: Extracted “VideoAgent” implementation code to let the user choose a targeted object

The communication protocol between the Object tracker agent and the Tracking manager agent is based on the CFP (CALL FOR PROPOSAL) performative. After the Object tracker agents have been found, the Tracking manager agent will call for proposal to request the Object tracker agents to track the targeted object location and wait for the

answer from the object tracker agents. Figure 4.4 shows the extracted implementation code calling for a proposal to track the targeted object.

The Tracking manager agent receives tracking result and evaluation score as the reply from the Object tracker agents if the targeted object successfully tracked in the video frame provided. The ACCEPT_PROPOSAL performative is used for the message of the acceptance of the proposal sent by the Object tracker agent. The best-tracking result will be determined from the tracking result, and the evaluation score received from the object tracker agents as shown in Figure 4.5.

```
53.     …
54.
55.     class beginTracking extends TickerBehaviour
56.           {
57.                   …
58.                   private static final long serialVersionUID = 1L;
59.                   private AID bestTracker; // The agent who provides the best
60.     offer
61.                   private AID trackerAgents[];
62.                   private int bestDistance;  // The best offered price
63.                   private int repliesCnt = 0; // The counter of replies from
64.     tracker agents
65.                   private MessageTemplate mt; // The template to receive replies
66.                   private int step = 0;
67.                   private List <Mat> videoFrames;
68.
69.                   //private ACLMessage request;
70.                   public beginTracking(Agent a, int miliseconds)
71.                   {
72.                           super(a, miliseconds);
73.                   }
74.
75.                   protected void onTick()
76.                   {
77.                           switch (step) {
78.                           case 0:
79.                                   // Update the list of tracker agents
80.                                   DFAgentDescription template = new
81.     DFAgentDescription();
82.                                   ServiceDescription sd = new ServiceDescription();
83.                                   sd.setType("object-tracking");
84.                                   template.addServices(sd);
85.                                   try {
86.                                           DFAgentDescription[] result =
87.     DFService.search(myAgent, template);
88.                                           System.out.println("Found the following
89.     tracker agents:");
90.                                           trackerAgents = new AID[result.length];
91.                                           for (int i = 0; i < result.length; ++i) {
92.                                                   trackerAgents[i] =
93.     result[i].getName();
94.
95.           System.out.println(trackerAgents[i].getName());
96.                                           }
                                    }
97.                                   catch (FIPAException fe) {
98.                                           fe.printStackTrace();
99.                                   }
100.                                  ///Only proceed to next step when more than ONE
101.    tracker agent found
102.                                  if (trackerAgents.length > 1)
103.                                  {
104.                                          System.out.println("GUI frame number is :
105.    " + gui.getFrameNumber() );
106.                                          step = 1;
107.                                  }
108.                                  break;
109.                          case 1:
110.                                  System.out.println("Step 2 : GUI frame number is
111.    : " + ObjRecognitionController.getVideoArraySize());

                                      …
```

Figure 4.3: Extracted "VideoAgent" implementation code to search for object tracker agents from directory facilitator (df) agent

```
…
        // Send the cfp to all tracker agents
        ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
        for (int i = 0; i < trackerAgents.length; ++i) {
                cfp.addReceiver(trackerAgents[i]);
        }

        if( ObjRecognitionController.getVideoArraySize() > 100)
        {
                //cfp.setContentObject( (Serializable)
ObjRecognitionController.videoFrames );
                cfp.setContent("Frame count inside: " +
ObjRecognitionController.getVideoArraySize() );
                cfp.setConversationId("image-frame");
               cfp.setReplyWith("cfp"+System.currentTimeMillis());
               // Unique value
                        myAgent.send(cfp);
                        System.out.println("CFP sent.");
                        // Prepare the template to get proposals
                                        mt =
MessageTemplate.and(MessageTemplate.MatchConversationId("image-frame"),

        MessageTemplate.MatchInReplyTo(cfp.getReplyWith()));
                step = 2;
        }
        break;
        case 2:
        // Receive all proposals/refusals from tracker agents
                ACLMessage reply = myAgent.receive(mt);
                if (reply != null ) {
                // Reply received
                System.out.println("Received PROPOSAL / REFUSAL from agents.");
                        if (reply.getPerformative() == ACLMessage.PROPOSE) {
                        // This is an offer
                        int distance = Integer.parseInt(reply.getContent());
                                if (bestTracker == null || distance <
bestDistance) {
                                // This is the best offer at present
                                bestDistance = distance;
                                bestTracker = reply.getSender();
                                                }
                                        }
                                        repliesCnt++;
                                        if (repliesCnt >= trackerAgents.length) {
                                                // We received all replies
                                                step = 3;
                                        }
                                }
                                else {
                                        System.out.println("Reply is null.");
                                        block();
                                }
                                break;
                        …
```

Figure 4.4: Extracted “VideoAgent” implementation code calling for a proposal to track targeted object

```
173.          …
174.          ACLMessage order = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
175.          order.addReceiver(bestTracker);
176.          order.setContent("Frame Number");
177.          order.setConversationId("object-tracking");
178.
179.          order.setReplyWith("order"+System.currentTimeMillis());
180.          myAgent.send(order);
181.          // Prepare the template to get the tracking reply
182.                                   mt =
183.    MessageTemplate.and(MessageTemplate.MatchConversationId("object-tracking"),
184.
185.          MessageTemplate.MatchInReplyTo(order.getReplyWith()));
186.                  step = 4;
187.                  break;
188.          case 4:
189.                  // Receive the tracking reply
190.                  reply = myAgent.receive(mt);
191.                  if (reply != null)
192.                  // Tracking reply received
193.                  System.out.println("Tracking reply received.");
194.                  if (reply.getPerformative() == ACLMessage.INFORM) {
195.                  // Tracking successful.
196.          System.out.println("Target found in frame by
197.    "+reply.getSender().getName());
198.          System.out.println("Distance = "+bestDistance);
199.          myAgent.doDelete();
200.                  }
201.
202.                  …
```

Figure 4.5: Extracted “VideoAgent” implementation code to determine the best-tracking result

4.4 Implementation of JADE Based Object Tracker Agent

The Object tracker agent as designed in Chapter 3 is implemented using a JADE agent class named as “CamshiftAgent”. The tracking algorithm used is based on the CAMShift algorithm. Firstly the “CamshiftAgent” are required to be registered in the yellow pages. The resources like the path to retrieve the initial location of the targeted object and the video frames have to be ready for the startup argument. Figure 4.6 shows the extracted “CamshiftAgent” implementation code for registration and resources set up.

```
1.    …
2.
3.    public class CamshiftAgent extends Agent {
4.
5.          public CamshiftAgent() {
6.                  // TODO Auto-generated constructor stub
7.          }
8.
9.          …
10.
11.         // Put agent initializations here
12.         protected void setup() {
13.
14.                 System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
15.
16.                 // Register the CAMshift trackers in the yellow pages
17.
18.                 // Get the resources for tracking as a start-up argument
19.                 Object[] args = getArguments();
20.                 // Printout a welcome message
21.                 System.out.println("Hallo! Camshift-agent
22.   "+getAID().getName()+" is ready with parameter:" + args);
23.                 String filename =
24.   getAID().getName().substring(0,getAID().getName().indexOf("@"))  +
25.   ".txt";
26.                 try{
27.                     writer = new PrintWriter( filename , "UTF-8");
28.                     writer.println("The first line");
29.                     System.out.println("*** ========== ==========
30.   ========== ========== ========== ***");
31.                 } catch (IOException e) {
32.                    // do something
33.                     System.out.println("Writing file :" + filename + "
34.   failed.");
35.                 }
36.
37.                 if (args != null && args.length > 0) {
38.
39.                         _vmin = Integer.parseInt((String) args[0]);
40.
41.                     if( args.length > 1)
42.                         _vmax = Integer.parseInt((String) args[1]);
43.
44.                     if( args.length > 2)
45.                         smin = Double.parseDouble((String) args[2]);
46.
47.                         System.out.println("Argument is (_vmin, _vmax,
48.   smin): " + _vmin + "," + _vmax + "," + smin);
49.                 }
50.
51.                 DFAgentDescription dfd = new DFAgentDescription();
52.                 dfd.setName(getAID());
53.                 ServiceDescription sd = new ServiceDescription();
54.                 sd.setType("object-tracking");
55.                 sd.setName("camShiftTracker");
56.                 dfd.addServices(sd);
57.
58.                 …
59.
```

Figure 4.6: Extracted “CamshiftAgent” implementation code for registration and resources setup

The Tracking manager agent sends the request for the offer of tracking an object through CFP to the “CamshiftAgent”. The “CAMShiftAgent” will then react on the CFP by detecting the initial bounding box or kernel of the target in the frame 01 of the video.

Then using inclusive lower boundary array (_vmin) and the inclusive upper boundary array (_vmax) assign to the “CAMShiftAgent”, the colour histogram that represents the target object will be created. Colour histogram is created using the colour information of the pixels within the bounding box as discussed in 2.3.1. Take note that in the system, every “CAMShiftAgent” is assigned with unique values. These values are discussed in Table 5.4. Then, during the next frame of the video, a back projection of the colour histogram values is applied to each pixel of the frame. After that, based on the location of the target in the initial frame and the distribution of the back projection, a new target bounding box will be obtained.

Then, the CAMShift agent will update the colour histogram that represents the appearance of the target object using its new bounding box location and size. The new location information is also used to initialise the Kalman filter in the “CAMShiftAgent”. However, at this moment the Kalman filter only acts as an observer of the tracking process. Only the result of the CAMShift is used as tracking result.

Every time the size of the bounding box is obtained, the “CAMShiftAgent” will determine either the target object is occluded or not. Based on the literature review, an object can be defined as occluded when the current bounding box size is less than 30% of its initial size.

When an object is occluded, the CAMShift will switch over as an observer while the Kalman filter is used to predict the target object position. The CAMShift tracker will stop updating its colour histogram because updating the histogram based on the occluded object will cause the tracker to fail to re-identify the object when the object reappears after the occlusion.

Because the “CAMShiftAgent” track an object using CAMShift and Kalman filter therefore when the targeted object is detected at the initial stage, the tracking will only end at the end of the video. If the targeted object is detected and tracked, the PROPOSE performative is used for the messages as the reply with the tracking result and evaluation score. If the targeted object could not be tracked from the video frames provided, the REFUSE performative is used for the message as the reply.

One reason for the agent to reply the REFUSE performative is when the “CamshiftAgent” fails to open the video file or the target image file. A file can fail to open when the “CamshiftAgent” is deployed in a network computer, and the computer does not have permission to access the file. Another reason that can cause the agent to reply the REFUSE performative is when corrupted, or empty video file is used for tracking. However, in this research, all the agents are run on one computer. Figure 4.7 shows the extracted “CamshiftAgent” implementation code for responding the CFP received.

```
90.   …
91.
92.         /**
93.         * This is the behaviour used by tracker agents to response
94.         * incoming requests for offer from tracking manager agent.
95.         * If the targeted object is traced the tracker
96.         * agent replies with a PROPOSE message specifying the track
97.   result.
98.         * Otherwise a REFUSE message is sent back.
99.         */
100.        private class OfferRequestsServer extends SimpleBehaviour {
101.        …
102.        …
103.
104.                public void action() {
105.                        MessageTemplate mt =
106.  MessageTemplate.MatchPperformative(ACLMessage.CFP);
107.                        ACLMessage msg = myAgent.receive(mt);
108.                        if (msg != null) {
109.                                // CFP Message received. Process it
110.
111.                                System.out.println("Running Image Track
112.  Object." );
113.                                //contentImg = (Image)
114.  msg.getContentObject();
115.                                …
116.
117.                                String strContent = msg.getContent();
118.                                System.out.println("OfferRequestsServer
119.  received: " + strContent);
120.                                ACLMessage reply = msg.createReply();
121.                                //Get the score based on the Track Result
122.                                Integer score = new
123.  ProposalEvaluator(trackBoxResult).getDistance();
124.                                if (
125.  msg.getConversationId().equals("image-frame") ){
126.                                        // After receiving image-frame from
127.  video proposal
128.
129.        reply.setPerformative(ACLMessage.PROPOSE);
130.                                        reply.setContent(
131.  Integer.toString(score);
132.                                }
133.                                else {
134.                                        // The targeted object is NOT
135.  tracked.
136.
137.        reply.setPerformative(ACLMessage.REFUSE);
138.                                        reply.setContent("not-available");
139.                                }
140.                                myAgent.send(reply);
141.                                System.out.println(reply.toString());
142.                                finished = true;
143.                        }
144.                …
```

Figure 4.7: Extracted "CamshiftAgent" implementation code for responding the CFP received

4.5 Implementation of JADE Based Proposal Evaluator Agent

The Proposal evaluator agent as designed in Chapter 3 is implemented using a JADE agent class named as "EvaluatorAgent". The "EvaluatorAgent" registers with the DF as shown in Figure 4.8 and waits for the "CamshiftAgent". After receiving the tracking result sent by

the “CamshiftAgent”, “EvaluatorAgent” evaluates the tracking result and replies the evaluation score back to the “CamshiftAgent” as shown in Figure 4.9.

The tracking result from the “CamshiftAgent” is a four-dimensional array that consists of the x-coordinate, y-coordinate, width and height of the bounding box. The “EvaluatorAgent” will calculate the average coefficient of variation (Cv) of the array. The coefficient of variation is the ratio of the standard deviation to the mean of the data in the array. Then, the score of the tracking result is calculated as the Distance travel by targeted object multiply the coefficient of variation.

```
1.      …
2.
3.        public class EvaluatorAgent extends Agent {
4.
5.        …
6.
7.        protected void setup() {
8.                System.out.println(getLocalName()+" STARTED");
9.                Object[] args = getArguments();
10.               if (args != null && args.length > 0) {
11.                       initiator = new AID((String) args[0],
12.       AID.ISLOCALNAME);
13.               }
14.
15.               try {
16.                       // create the agent descrption of itself
17.                       DFAgentDescription dfd = new DFAgentDescription();
18.                       dfd.setName(getAID());
19.                       // register the description with the DF
20.                       DFService.register(this, dfd);
21.                       System.out.println(getLocalName()+" REGISTERED WITH
22.       THE DF");
23.               } catch (FIPAException e) {
24.                       e.printStackTrace();
25.               }
26.       …
27.
28.
```

Figure 4.8: Extracted “EvaluatorAgent” implementation code for registration

```
35.         …
36.         addBehaviour(new CyclicBehaviour(this) {
37.                             public void action() {
38.
39.         if (reply.equalsIgnoreCase(msg.getContent())) {
40.                       // if an EVALUATE message arrives
41.                           // then proceed with the valuation score calculation
42.                       // then send a EVALUATED message
43.
44.           System.out.println(myAgent.getLocalName()+" RECEIVED EVALUATE
45.       MESSAGE FROM "+msg.getSender().getLocalName());
46.                             ACLMessage replyT = msg.createReply();
47.
48.                            // calculation evaluation score based the tracking
49.       result sent
50.                             replyT.setContent("Evaluated");
51.                             myAgent.send(replyT);
52.
53.           System.out.println(myAgent.getLocalName()+" SENT EVALUATED
54.       MESSAGE");
55.         …
56.
57.
```

Figure 4.9: Extracted "Evaluator Agent" implementation code for replying back the evaluation score

4.6 Simulation of Object Tracking System in JADE Platform

A simulation of object tracking system based on the proposed framework is conducted in JADE platform. The simulation is carried out with two (2) CAMShift object tracker agents with different parameters setting. Figure 4.10 shows the JADE Agent Management GUI of the object tracking system. In Figure 4.10, two (2) CAMShift tracker agents were started for tracking, namely agent cam01 and agent cam02.

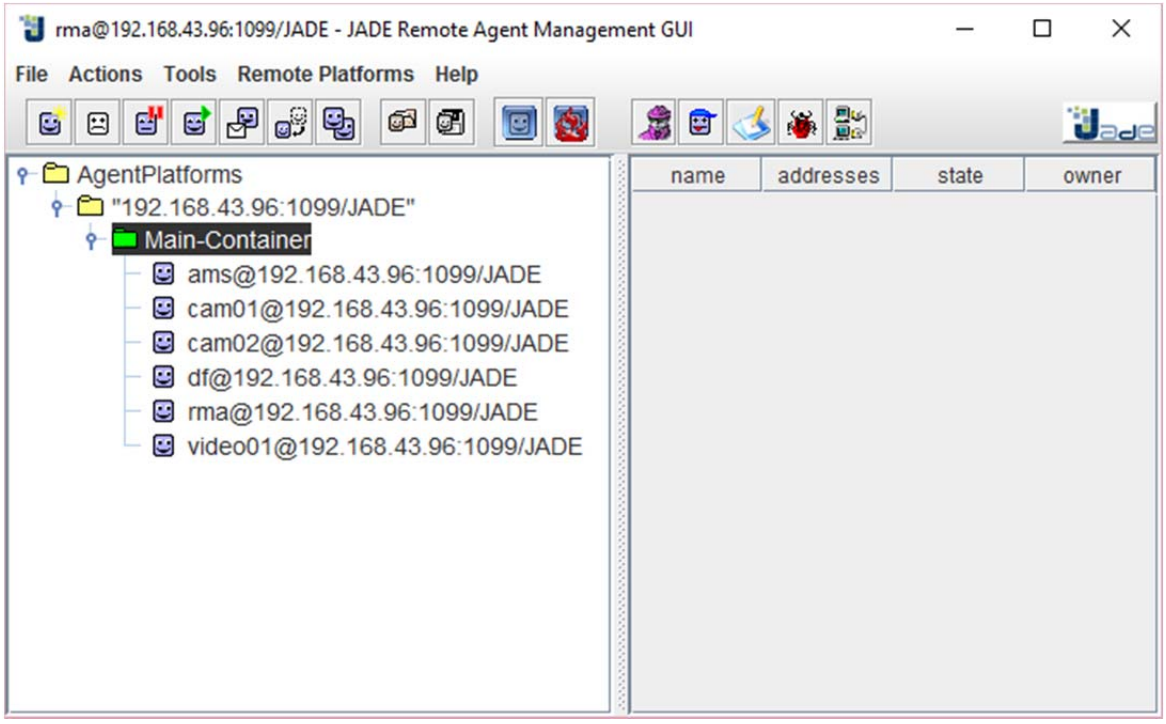


Figure 4.10: JADE Agent Management GUI

The output of the tracking result of the simulation from the CAMShift object tracker agents is shown in Figure 4.11 and Figure 4.12. Figure 4.11 shows the tracking result of CAMSHift when the inclusive lower boundary array (_vmin) and the inclusive upper boundary array (_vmax) during the process of generating the colour histogram are set to (_vmin = 1, v_max = 128). The Figure 4.12 shows tracking result when the parameter were set to (_vmin = 129, v_max = 256). The selection of these values is critical to achieve good segmentation of target object and affect the tracking result.

The tracking result in Figure 4.11 shows that the tracker can successfully track an object in frame number: (0, 20 and 60). However, the tracker begins to track the wrong object in frame number 80 after another pedestrian occluded the object.

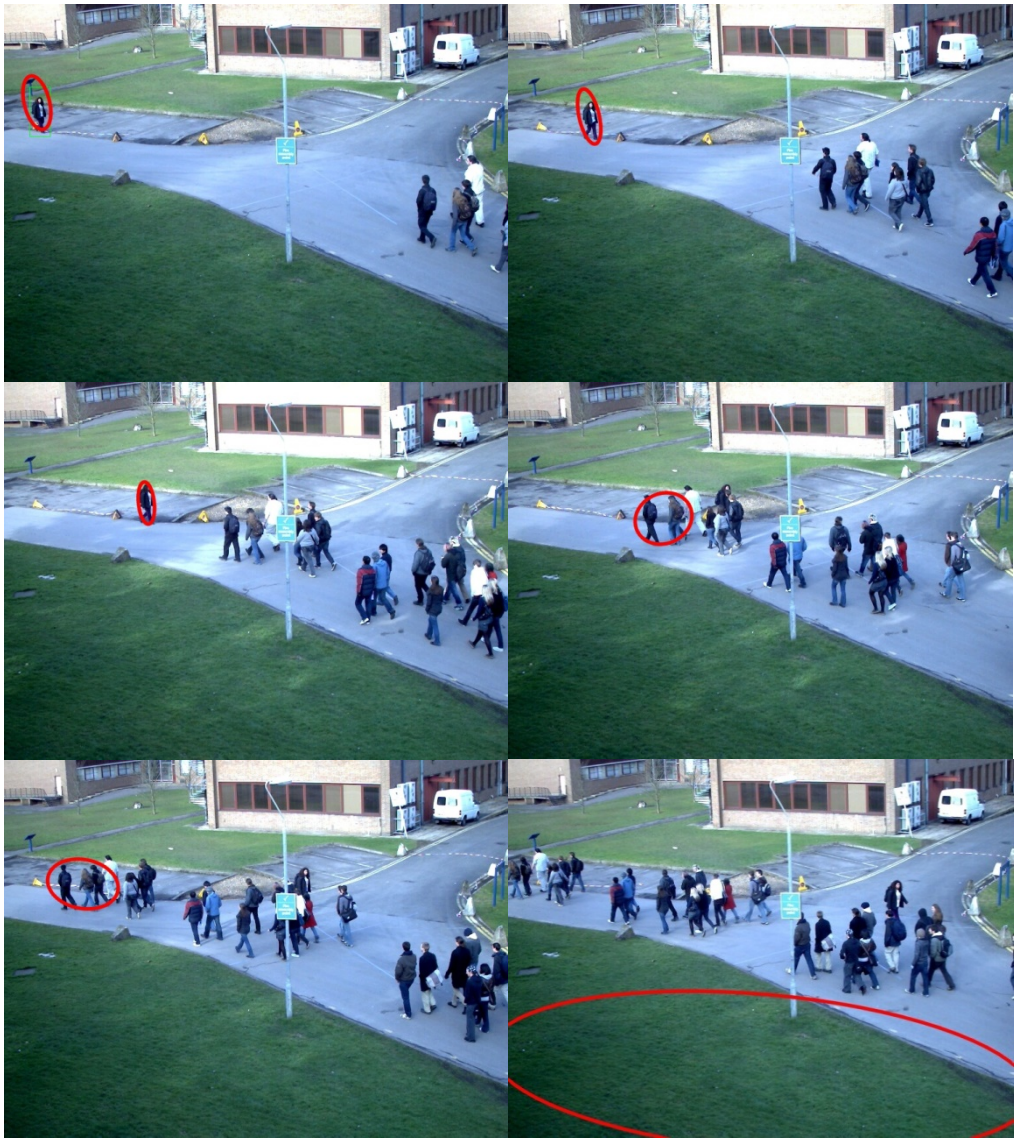


Figure 4.11: Tracking Results (Frame No. 1, 20, 40, 60, 80, 100: Order from left to right, top to bottom) based on parameter (_vmin = 1, v_max = 128)

Although the tracker is not perfect, the result is better than another CAMShift tracker agent where the parameters are set to (_vmin = 129, v_max = 256) as shown in Figure 4.12. The tracking result in Figure 4.11 is better because the position of the target (the ellipsoid) is tightly surrounding the target object until frame number 40 in Figure 4.11 the FDA of frame 40 is 0.95. Figure 4.12 shows that the tracker has missed the object at frame

20, then in Frame 40 the object is in the tracked area again. Although the object is in the ellipsoid and the overlapped (numerator) is one but because the ellipsoid area (denominator) is big, the FDA for frame 40 is 0.03 only.

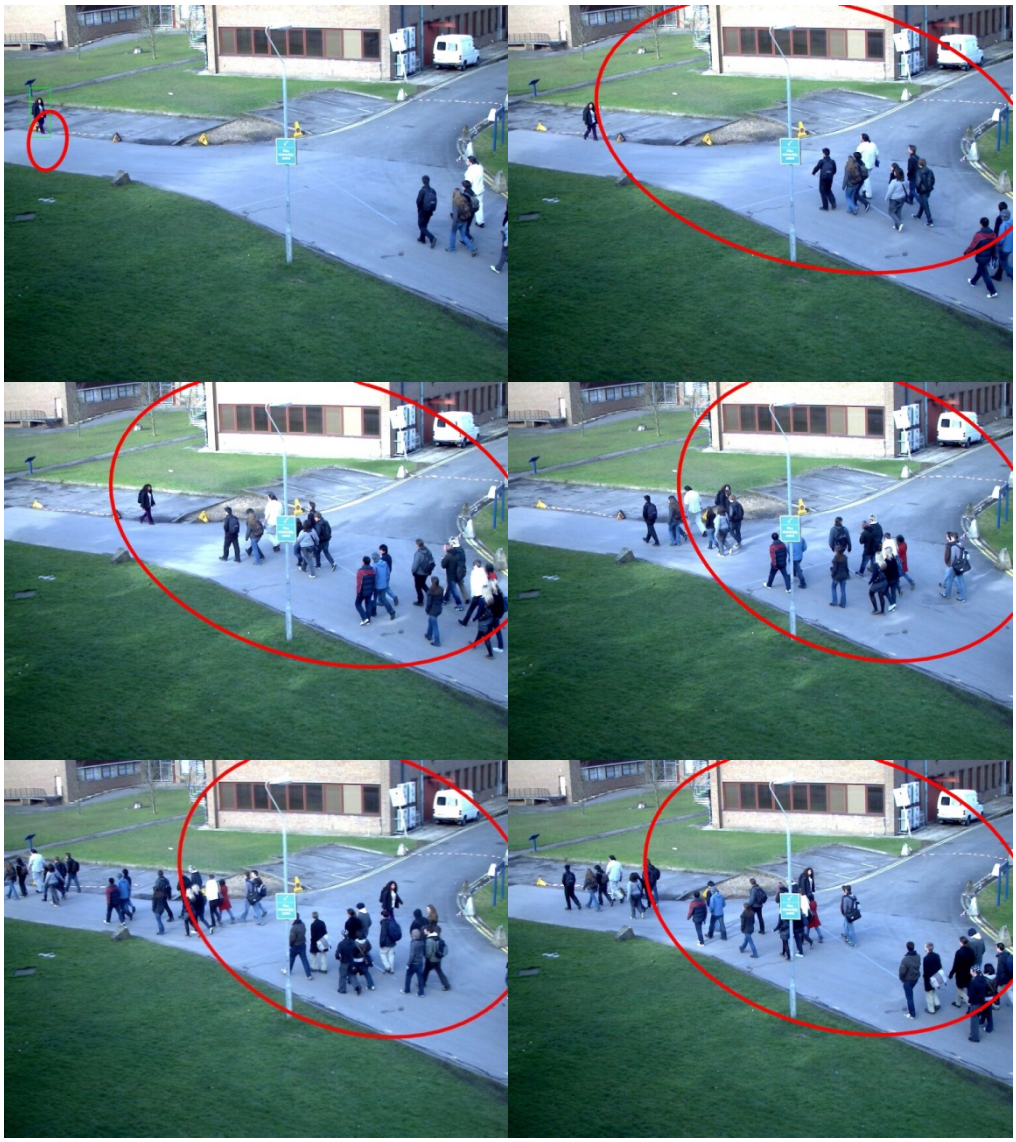


Figure 4.12: Tracking Results (Frame No. 1, 20, 40, 60, 80, 100: Order from left to right, top to bottom) based on parameter (_vmin = 129, v_max = 256)

The tracking result shows that with unsuitable parameter settings, a tracker will make a mistake and give an incorrect tracking result. From this simulation study, it is concluded that parameter settings are very important when tracking an object. Hence, the proposed multi-agent system to use object tracker agents with different parameter settings could assist in improving the tracking result. It is very important to have a Proposal evaluator agent that would act as an evaluator in the multi-agent system to determine the tracking result produced by the “CamshiftAgents”.

4.7 Summary

This chapter presents the Platform Specific Models (PSMs) of the implementation for object tracking using JADE as designed in the previous chapter. The discussion includes the agent negotiation mechanism in object tracking, the implementation of JADE based Tracking manager agent, the implementation of JADE based Object tracker agent and the implementation of JADE based Proposal evaluator agent. The tracking algorithm used is based on CAMShift algorithm and Kalman filter. A simulation study has been carried out to highlight the importance to have a multi-agent system for object tracking with different object tracker agents to track the targeted object using different parameter settings to improve the tracking result. The implemented system in named as multi CAMShift agent object tracking system.

CHAPTER 5
RESULTS AND DISCUSSION

5.1 Introduction

In this chapter, a preliminary experiment that was conducted to show the limitation of the CAMShift tracker, Kalman Filter tracker and Particle Filter tracker was discussed. From the result of this preliminary experiment, the CAMShift tracker was validated. Then, in the next section, the result of the experiment using the benchmark database is discussed. The experiment result supports that the first objective of this research is achieved. Section 5.4 shows the message passing activity between the agents and the communication protocol was performed correctly. With a combination of the CAMShift and Kalman filter implemented as multi-agents system, the occlusion problem has also been solved.

5.2 Preliminary Experiment

This section describes the preliminary experiment conducted using three popular object tracking methods, namely the Mean-shift tracker, the Background Subtraction with Kalman Filter and the Likelihood function with Particle Filter. The preliminary experiment helps in getting a better understanding of the state of the art object tracking methods in term of their strength and limitation. The findings from this experiment motivate the proposed solution in this research.

In the preliminary experiment, simulation videos sequences are used to test the performance of the previously proposed tracker. Simulation videos are generated videos where the objects, location of the objects, lighting and background of the scene are created

using 3D computer graphics. These videos have controlled environment. Also, simulated videos can provide accurate ground truth for performance evaluation.

In simulation videos, an ideal environment without any noise and distraction can be created. The results contained under such environment could reflect the actual performance of the tested tracking methods. It will also be easier to analyse the effect of occlusion in simulation videos since the environment and interaction between objects can be controlled.

When comparing the performance of trackers, it is important to use the same video to test them. This is because the complexity of object tracking varies in different videos. Quantitative measurement of the setting such as the position of cameras, shapes of target object (either human or ball), would affect the tracking result. However as long as the video used to test the trackers are the same, the result obtained could be used to benchmark a precision of a tracker.

5.2.1 Simulation Video Data

A set of six simulation video sequences with moving object are generated. The label of the videos and their description are shown in Table 5.1. The simulation was created using OpenGL and Visual C++.

The simulation videos show a moving ball with a single colour. The rationale of the creating this video is because a ball, unlike human, is symmetrical in every direction. A ball can be seen from different directions and still look the same. Other than that, the simulation video creates a controlled environment where it is easier to analyse the performance of a tracker at dealing with occlusion. In the control environment, there is no other factor other than occlusion that can affect the tracking result. Therefore the actual

strength and limitation of each tracker can be identified in the preliminary experiment. When using a non-controlled video to run the preliminary experiment, the appearance of the target object can be affected by another factor, therefore the result obtained will be lack of just because poor result can be due to another factor such as noise or self-occlusion of the target object.

The video sequence of video label A1 is shown in Figure 5.1, which simulates a single colour ball moves from left, rolls over to the right and exits the video frame. The ball in the video is simulated to move at constant speed and direction.

Table 5.1: Video label and description

Label	Description
A1	Moving ball with constant speed and direction
A2	Moving ball with constant speed and arbitrary direction
A3	Moving ball with constant speed and direction with full occlusion
A4	Moving ball with constant speed and arbitrary direction with full occlusion
A5	Moving ball with constant speed and direction with partial occlusion
A6	Moving ball with constant speed and arbitrary direction with partial occlusion

Figure 5.2 shows another simulated video sequence with a single colour ball moving (from left to right direction) towards the middle of the video frame and then moving backwards to the left and exit the frame. This video sequence is designed to test how tracking methods react to an arbitrary movement which is labelled as A2.

An obstacle is placed in the middle of the video frame to represent occlusion in simulation video. In the experiments, two types of occlusion are concerned which included full occlusion and partial occlusion. Figure 5.3 shows a big rectangle is placed in the middle of the video frame in a video with ball moves at constant speed and direction.

This video labelled as A3 is used to test how an object tracking method could handle moving object after full occlusion. Figure 5.4 shows some video frames of a video sequence of video label A5 for testing partial occlusion at constant speed and direction. Video with partial and full occlusion is also created for moving an object with arbitrary direction change as described of video label A4 and A6.



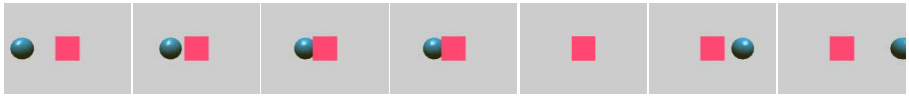
(a) (b) (c) (d) (e)

Figure 5.1: Video Frame (a) Frame no. 1, (b) Frame no 10, (c) frame no. 22, (d) frame no. 30 and (e) frame no. 50 of simulation video A1.



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

Figure 5.2: Video Frame (a) Frame no. 1, (b) Frame no. 10, (c) frame no. 22, (d) frame no. 30 and (e) frame no. 50 of simulation video A2



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

Figure 5.3: Video Frame (a) Frame no. 1, (b) Frame no 10, (c) frame no. 12, (d) frame no. 13, (e) frame no. 22, (f) frame no. 30 and (g) frame no. 50 of simulation video A3.



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

Figure 5.4: Video Frame (a) Frame no. 1, (b) Frame no 10, (c) frame no. 15, (d) frame no. 22, (e) frame no. 30 and (f) frame no. 50 of simulation video A5.

5.2.2 Tracking Performance Measurement

To calculate the result for both mentioned measurements, ground truth is generated using object detection algorithm based on background subtraction. The path of the moving object in video sequence A1, A3 and A5 are identical while A2, A4 and A6 share another similar path. Therefore, only two object movement ground truths are generated for verification of the tracking results.

5.2.3 Evaluation Methods

The method used for evaluating the object tracking system in this research is Sequence Frame Detection Accuracy (SFDA) that is based on the framework by Kasturi et al. (2009). It is a highly cited protocol for performance evaluation of object detection and tracking in video sequences (Shu, Dehghan, Oreifej, Hand, & Shah, 2012). The measurements measure the number of objects detected and missed detection, false positives and spatial alignment of the system output and ground-truth object. The measurement requires calculation of the overlap ratio too.

To calculate the result for both mentioned measurements, ground truth must be generated. In most cases, ground truth can be generated manually or background subtraction for object tracking within simple background scene.

Additionally, a qualitative comparison of object trackers can also be made based on their ability to deal with entries of new objects and exits of existing objects, handle the missing observations (occlusion), and provide an optimal solution to the cost function minimisation problem used for establishing correspondence.

5.2.4 Preliminary Experiment Results

Experiments have been carried out to evaluate the performance of occlusion handling of the Kalman filter (KF) tracker, Particle filter (PF) tracker and Mean-shift (MS) tracker. The tracker algorithms in MATLAB script are modified and customised based on available sources to suit the experiments. The spatial information of the tracked object is written to text files. Tracking results of various tracker used for the experiments are shown in Table 5.2.

Table 5.2: Tracking result (SFDA) for six simulation videos

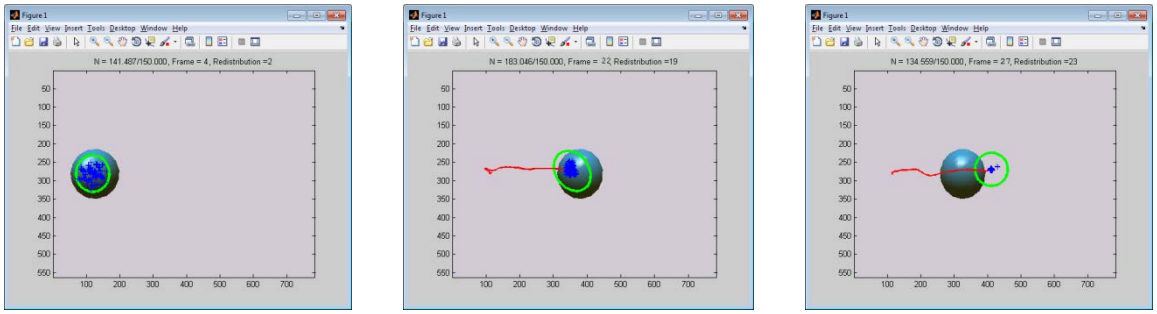
Video Sequence	SFDA		
	KF	PF	MS
A1	0.4628	0.4725	0.5196
A2	0.4583	0.1606	0.6071
A3	0.3434	0.3473	0.0912
A4	0.3732	0.3567	0.4825
A5	0.4568	0.3701	0.3781
A6	0.4677	0.2616	0.5370
Average	**0.4270**	**0.3281**	**0.4359**

Kalman filter (KF) tracker used in the experiments is modified from the background subtraction tracker with Kalman Filter by FarrokhNejad (2007) in MATLAB Central. The SFDA obtained from the tracking experiment using KF tracker is in between 0.3434 and

0.4677. The lowest SFDA score was obtained in video sequence A3 where full occlusion occurred.

Particle filter (PF) tracker is used to track an object in the same set of video sequences. Based on the SFDA score in Table 5.2, the result of PF tracker is poorer than KF tracker except in video Sequence A3. The advantage of PF tracker in A3 over KF is only 0.0038, which equals to 1.1%. The lowest SFDA is achieved when performing PF tracker on video sequence A2.

Based on observation, the PF tracker fails to detect the moving object in the video sequence A2 after frame number 22 (see Figure B.5 in Appendix B). A close examination found that the lost track of the object is due to a long period of the consistent trajectory of the moving object before frame 22. Stretched consistent trajectory caused the distribution area of the particle become contracted and cover only a small area in the video frame. Therefore, when the trajectory of the object changed suddenly, the PF fail to track the moving object as shown in Figure 5.5.



(a) (b) (c)

Figure 5.5: Particle distribution: a) particles cover a large area at the initial state; b) when object trajectory remains consistent between frames, the particle area shrunk; c) particle area become so small and fail to detect the moving object change direction

Lastly, Mean-shift (MS) tracker is used to test the performance of MS tracker. The result of the MS tracker was the best when no occlusion occurred. This can be identified by having the highest SFDA for both video sequence A1 (0.5196) and A2 (0.6071).

When full occlusion occurred in video sequence A3, the SFDA of MS tracker dropped drastically to 0.0912 only. For video sequence A4, full occlusion also occurred at frame number 18 (see Figure C.11 in Appendix C). However, the tracker manages to re-pick up the moving object because the object turns around to the location before full occlusion occurred and reappeared at frame 20 (see Figure C.14 in Appendix C). For partial occlusion, the MS tracker has slightly poorer SFDA result when compared to non-occlusion video sequences, A1 and A2.

In general, the average result obtained in this section are poorer than some improved tracking method such as Conte, Foggia, Percannella, & Vento (2010) work, which used similarity measurement of matrix representation and appearance model to track moving object in the dynamic scene. Conte, Foggia, Percannella, & Vento (2010) tested their tracking method on PETS 2009 S2-L1 video sequences and obtained average SFDA of 0.505 while average SFDA obtained when running the object tracking experiment using the six simulation videos are 0.427 (KF), 0.328 (PF) and 0.436 (MS) accordingly. It is important to note that comparing results between different papers is not fair because the object, the background scene and the occlusion situation in the video sequences used are different. The results of this experiment are suitable for understanding the effect of occlusion which is discussed in next section.

5.2.5 Effect of Occlusion

The SFDA in Table 5.2 generally provides an impression of the performance of three different trackers. However, the SFDA only provides the average performance of each tracker. To closely view the effect of occlusion, the Frame Detection Accuracy from frame number 12 to frame number 31 in video A3 (see Figure A.9 – A.11 in Appendix A) and A4 (see Figure A.13 – A.15 to in Appendix A) is collected and analysed.

Frame number 12 is the frame where the complete moving object is still visible in video sequence A3 and A4. Occlusion begins at frame number 13 for video sequences A3 and A4 while for video sequence A5 (see Figure A.17 - A.19 in Appendix A) and A6 (See Figure A.21 - A.23 in Appendix A); occlusion begins at and frame number 15.

In video sequence A3, as shown in the Frame Detection Accuracy line chart, Figure 5.6, after a full occlusion, MS tracker failed to detect the moving object while KF and PF tracker could recover after the full occlusion is over. The PF tracker consistently maintains a higher FDA along with the video frames even when occlusion occurred.

In video sequence A4, the path of the moving object in the video sequence is identical to the path of the moving object in video sequence A2. In these video sequences, the foreground object moved at a consistent speed to the middle of the video frame. At frame number 22, the object changed its direction and moved backwards (see Figure A.5).

Figure 5.7 shows that the PF tracker could track the moving object better in video sequence A4 (with occlusion) compared to in video sequence A2 (without occlusion). The reason for PF tracker to be able to track better in video sequence A4 is because the occurrence of occlusion that happens gradually allows the PF tracker to distribute the

particle into a bigger area before occlusion occurred. Therefore, when the object changes its moving direction, the particle area is big enough to track the moving object and thus did not lost track of the moving object as in video sequence A2.

In video sequence A3, the tracking result of MS tracker deteriorates drastically after occlusion. From the result, it is rational to say that MS tracker has the worst tracking ability after full occlusion as claimed by many previous works.

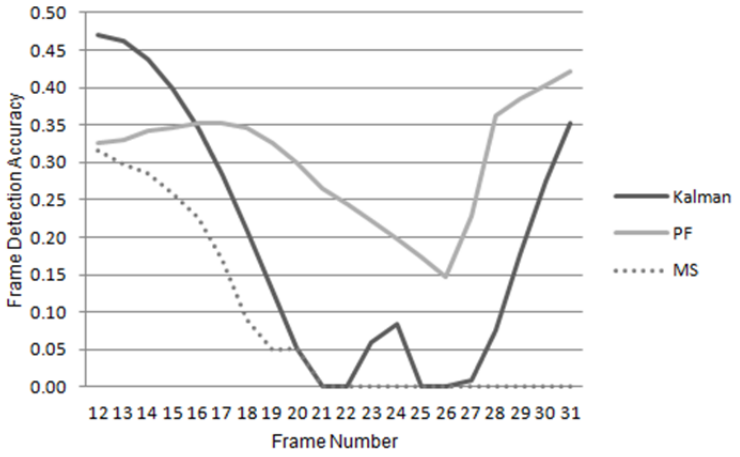


Figure 5.6: Frame Detection Accuracy of Frame No. 12 to Frame no. 31 for Video Sequence A3

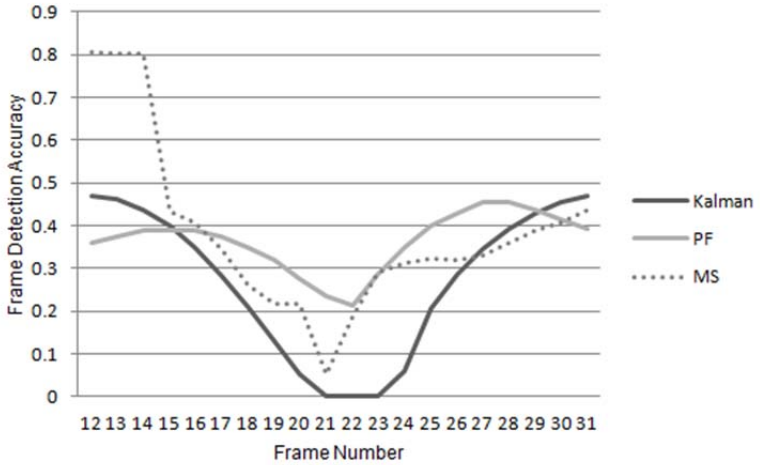


Figure 5.7: Frame Detection Accuracy of Frame No. 12 to Frame no. 31 for Video Sequence A4

For video sequences A5 where a moving object is only partially occluded, all the three trackers show FDA score between 0.3 and 0.5 during occlusion and score between 0.38 and 0.48 after the occlusion is over at frame number 31 as shown in Figure 5.8.

For foreground object that has arbitrary trajectory with partial occlusion, both KF tracker and MS tracker could successfully recover quickly after occlusion as shown in Figure 5.9. PF tracker requires a longer time to recover in video sequence A6 due to the arbitrary trajectory, which means the moving ball changed its moving direction after partial occlusion occurred (see Figure B.22 in Appendix B).

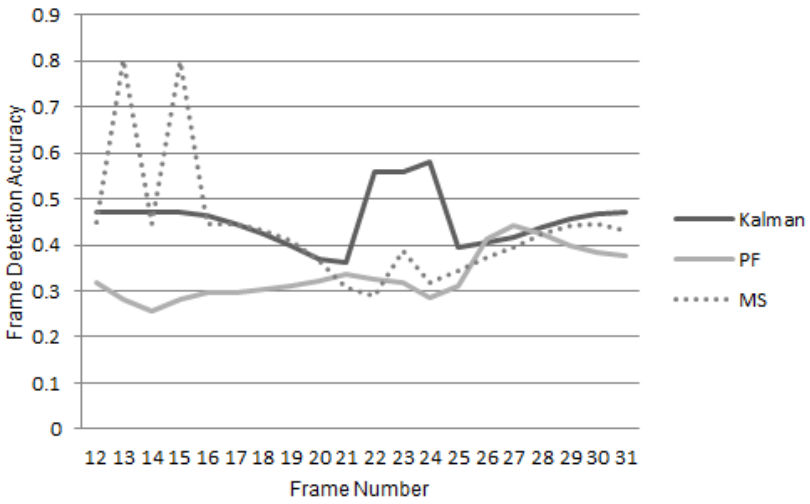


Figure 5.8: Frame Detection Accuracy of Frame No. 12 to Frame no. 31 for Video Sequence A5

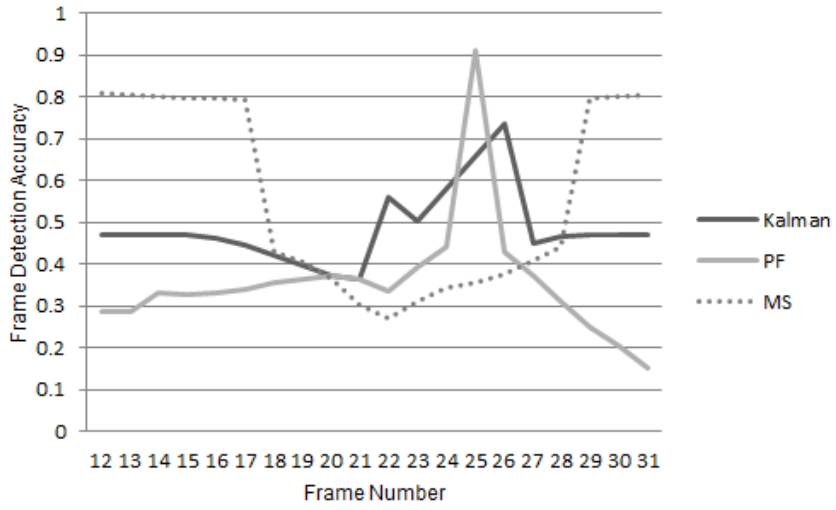


Figure 5.9: Frame Detection Accuracy of Frame No. 12 to Frame no. 31 for Video Sequence A6

5.2.6 Finding of Preliminary Experiment

After evaluating all the famous trackers, the results show the strength and the weakness of each evaluated tracker. This helps to build a better system. The important findings are:

i. Mean-shift tracker performed better than simple background subtraction tracker as mean-shift tracker has the mechanism to update the appearance model of the object.

ii. Mean-shift tracker requires proper selection of upper boundary array and lower boundary array value at the initial stage of object tracking process to ensure the appearance model is created correctly for tracking purpose.

iii. Kalman filter is effective to predict the location of the targeted track object even when the object is fully occluded.

5.3 Benchmark Dataset

After the preliminary experiment was conducted, a solution has been proposed (see Chapter 3) and implemented in the multi-agent platform (see Chapter 4). After the

implementation, the multi CAMShift agent object tracking system is tested using four datasets; namely the A1 simulation video and A3 simulation video (see Section 5.2.1), the PETS 2009 view001 S1 video and the CAVIAR entrance lobby video.

A1 and A3 are selected because these are computer generated videos. Therefore it is easier to analyse the performance of a tracker since there is no other factor such as the shape or the look that can affect the tracking result. A1 is a video with only the moving ball and A3 is a video with occlusion. PETS 2009 is real-world video with occlusion and CAVIAR entrance lobby is a real-world video with no occlusion.

Since the A1 and A3 simulation videos are computer generated videos, the ground truth of the objects in the video is fixed. The ground truth of PETS 2009 view 001 is obtained from Milan (2016) website while CAVIAR ground truth is obtained from the CAVIAR official website (Fisher R. , 2001).

5.4 Results and Analysis

In this section, the result of the multi CAMShift agent object tracking system is analysed both qualitatively and quantitatively. The qualitative result shows the tracking result in some important frame of our videos. This tracking result gives a better understanding of what happened during the tracking process.

In the quantitative result section, the Sequential Frame Detection Accuracy (SFDA) (Kasturi, et al., 2009) is used. The quantitative result compares the result obtained using the multi CAMShift agent object tracking system with some standard object tracking method which is available open source on the Internet.

5.4.1 Qualitative Result

This section shows the tracking result when using a built-in CAMShift tracker in MATLAB and when using multi CAMShift agent object tracking system. We begin the tracking experiment with video A1 and A3 to evaluate the multi CAMShift agent object tracking system in the controlled environment. Based on our observation, the tracking result from both CAMShift tracker in MATLAB and the multi CAMShift agent object tracking system are almost the same when tracking the moving ball in video A1. The similarity can be observed from tracking the result of MATLAB CAMShift and the multi CAMShift agent object tracking system. Because there is no occlusion in Video A1, therefore the advantage of adding Kalman filter cannot be shown here.

Then, when the experiment is conducted using video A3, the MATLAB CAMShift fails to track the object after the occlusion (see Figure 5.10). However, the multi CAMShift agent object tracking system, with the implemented Kalman filter, can predict the location of the ball when the ball is occluded. When the object reappears again, from occlusion, the CAMShift will take over to track the ball again (see Figure 5.11). The result shows that the Kalman filter is successfully implemented in the CAMShiftAgent.

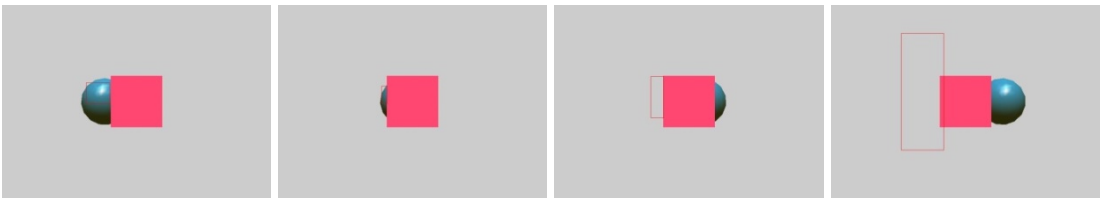


Figure 5.10: Track result from CAMShift Tracker in MATLAB (Frame number 15, 20, 25, 30 (Left to right).

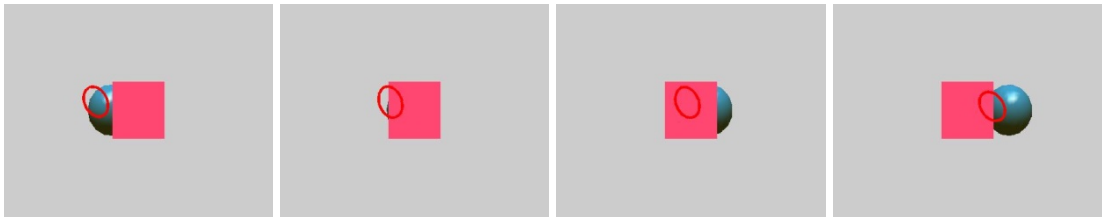


Figure 5.11: Track result from CAMShift Tracker agent (Frame number 15, 20, 25, 30 (Left to right).

After using video sequence A1 and A3 as test video, it is important to see how the tracker would perform when tracking a human target in an uncontrolled environment. PETS 2009 is used to test the tracker on tracking pedestrian. Figure 5.12 and Figure D.9 until D.16 (in Appendix D) shows the tracking result when using a CAMShift tracker in MATLAB to track the lady on the left in the first frame (target object) of the PETS 2009 video. Unlike tracking the ball in the simulation video that was mentioned in the preliminary experiment, the ball in the simulation video is round and look the same no matter how the ball moves. In contrast, when tracking the pedestrian of PETS 2009 (real-world object), many factors such as the movement of the pedestrian, the shadow of the nearby building, the occlusion caused by other moving objects will cause the appearance of the pedestrian to vary significantly from frame to frame. These factors will bring more challenges to the tracker.

The tracking process encounters a problem at frame 50 (See Figure 5.12) where another person occluded the target object in the video. During this time, the tracker included the occluding person as part of the tracking target. The problem caused the appearance model of CAMShift tracker to be wrongly updated. Hence, after the target separated with the occluding person, the tracker wrongly track the occlusion person as a target.



Figure 5.12: Track result from CAMShift Tracker in MATLAB (Frame number 1, 50, 70, 90, 110 and 130. Left to right, top to bottom).

During the occlusion, the occluding person is standing in front of the target (nearer to camera). So, most of the occluding object is visible to the camera while the target object is hidden from the camera. Therefore, when CAMShift tracker updates its appearance model, it used most of the occluding object as appearance model.

As a result, the MATLAB CAMShift tracker begins to track the wrong person in the video until frame number 110 (see Figure 5.12). In frame 110, the person exited the video frame, and the tracker begins to update wrongly causing the track window (the bounding box) to grow randomly.

Using the same video, the multi CAMShift agent object tracking system was used to track the same target object. Ten CAMShift tracking agents were used to track the target object. Each tracking agent runs the object tracking process by using the different values for the lower and upper boundary arrays (see Section 2.3.1). The video manager decided the agent with the parameter (lower boundary, _vmin = 25 and upper boundary, _vmax = 50) give the best result. The result of the tracker is shown in Figure 5.13. In the multi

CAMShift agent object tracking system, the “VideoManager” agent will pick the result from a “CAMShiftAgent” that has the highest Evaluation Score. The “CAMShiftAgent” received its Evaluation Score from “Evaluation Agent”. The Evaluation Score is calculated based on the coefficient of variation mentioned in section 4.5.

Figure 5.13 clearly shown that after the occlusion in frame 50, the target object still can be successfully tracked by the multi CAMShift agent object tracking system that uses the boundary parameters assigned. This is because the object that occludes the target object at frame 50 is very identical to the target object (both objects wearing a black jacket), but the target object is wearing a white shirt in the jacket. Therefore with the lower boundary, _vmin set at 25 and upper boundary set at _vmax = 50, the bounding box of the CAMShift agent does not grow to cover the occluding object. For CAMShift tracker in MATLAB, there is no way to set the boundary parameters. Therefore the tracker will wrongly include the appearance model of occluding object as tracking criteria. The FDA value for frame 50 in Figure 5.12 is 0.00 while in Figure 5.13 is 0.65.



Figure 5.13: Track result from CAMShift Tracker agent (Frame number 1, 50, 70, 90, 110 and 130. Left to right, top to bottom) with parameter (_vmin = 25 and _vmax = 50).

5.4.2 Quantitative Result

After looking at the tracking result qualitatively in the previous section, the racking results of the multi CAMShift agent object tracking system are evaluated quantitatively using Sequential Detection Accuracy (SFDA). Note that SFDA measures false negatives, false positives, and ID-switches.

For comparison purpose, an experiment has been conducted using the Kalman filter tracker, Particle Filter tracker and the Mean shift tracker. The video from PETS 2009 and CAVIAR were used in the experiment. Table 5.3 shows the result obtained after the experiments are conducted. The result of the KF is the worst in Table 5.3 because target object is located using background subtraction method. When there are multiple objects in the PETS 2009 and CAVIAR videos, background subtraction method will assume the blob of multiple objects as one object. Therefore the SFDA result for KF tracer is the lowest.

Table 5.3: Tracking result (SFDA) for the PETS 2009 and CAVIAR Video

Video Sequence	SFDA		
	KF	PF	MS
PETS 2009	0.0237	0.2387	0.2677
CAVIAR	0.1601	0.4688	0.4986

Other than the three trackers in Table 5.3, a MATLAB built-in CAMShift tracker is also used for the trackers used for comparison in this section. The multi CAMShift agent object tracking system that is set up with two “CamshiftAgent”, with ten “CamshiftAgent” and with twenty “CamshiftAgent” are created and tested.

The lower boundary (_vmin) and upper boundary (_vmax) set for each object tracking agent in the system is shown in Table 5.4. When different _vmin and _vmax are assigned to the agents, the colour histogram that is used to track the object in each agent will be different. These will cause the agents to produce a different result, and the tracking manager agent will pick the best result from a tracker agent.

Table 5.4: The lower boundary (_vmin) and upper boundary (_vmax) assigned for each object tracking agents when a different number of agents is used

No. of agents	Array of _vmin and _vmax
2 Agents	{ (1, 128) , (129, 256) }
10 Agents	{ (1, 25), (25, 50), (50, 75), (75,100), (100,125), (125, 150), (150, 175), (175, 200), (200, 225), (225, 256)}
20 Agents	{ (1, 13), (13, 26), (26, 39), (39,52), (52,65), (65, 78), (78, 91), (91, 104), (104, 117), (117, 130), (130, 143), (143, 156), (156, 169), (169,182), (182,195), (195, 208), (208, 221), (221, 234), (234, 247), (247, 256) }

The object tracking result of using a different number of "CamshiftAgent" is shown in Table 5.5, Table 5.6, Table 5.7 and Table 5.8. The MATLAB built-in CAMShift is chosen because it is easy to implement and engineers and scientist widely accept MATLAB as a reliable, quick prototyping tool (MathWorks, 2016).

Table 5.5 shows the object tracking result on simulation video A1 discussed in Section 5.2.1. The video is a video of a moving ball with constant speed and direction. The result in Table 5.5 is almost similar to the result in Table 5.2 because video A1 is a simple video with no occlusion in the video. Therefore, all the trackers perform well with a slight difference.

Table 5.5: Tracking result on simulation video A1 dataset by using a different number of "CAMShiftAgents"

Object Tracking System	SFDA
MATLAB CAMShift	0.51
Proposed MAS object tracking system with (Two (2) tracking agents	0.52
Proposed MAS object tracking system with (Ten (10) tracking agents	0.51
Proposed MAS object tracking system with (Twenty (20) tracking agents	0.51

Table 5.6 shows the object tracking result on simulation video A3 (see Section 5.2.1.) The A3 video is a video of a moving ball with constant speed and direction with full occlusion in the middle of the video.

Table 5.6: Tracking result on simulation video A3 dataset by using a different number of "CAMShiftAgents"

Object Tracking System	SFDA
MATLAB CAMShift	0.09
Proposed MAS object tracking system with (Two (2) tracking agents	0.32
Proposed MAS object tracking system with (Ten (10) tracking agents	0.36
Proposed MAS object tracking system with (Twenty (20) tracking agents	0.33

Generally, the result in Table 5.6 is not better than the result of KF and PF tracker in Table 5.2 except when using the multi CAMShift agent object tracking system with 10 tracking agents. Unlike KF and PF tracker that uses the spatial information to track the target object, the CAMShift tracker in MATLAB solely depends on the colour of the target object to track. The CAMShift in MATLAB tracker can successfully track the object until the object is fully occluded (see Figure D.6). After the occlusion, the tracker stops at the location where the object is fully occluded and cannot track the object after it reappears.

The CAMShift agent has integrated Kalman filter. Therefore, the tracker can continue to track even after the occlusion.

For the multi CAMShift agent object tracking system with two tracking agents, the SFDA is 0.32, which is 3.5 times better than the result produced by the CAMShift in MATLAB (SFDA = 0.09). The setting by using two agents is sufficient to track the object in the simulation video A3 dataset because of the simplicity of the appearance of the targeted track object (see Figure 5.3).

In Table 5.6, the result of the multi CAMShift agent object tracking system when using ten tracking agents is better than the result when using two tracking agent. However, the result of the multi CAMShift agent object tracking system when using 20 tracking agents is not better than the result when using ten tracking agents. Therefore, adding more tracking agents into the multi CAMShift agent object tracking system does not necessarily improve the tracking result.

The multi CAMShift agent object tracking system has integrated Kalman filtering to estimate the location of the target object. Therefore the target object can be successfully tracked when it reappears after long occlusion. Because of the ball in the video move at a constant speed, it is easy for Kalman filter to predict the ball movement.

In Table 5.7, the tracking result on PETS 2009 view001 S1 is presented. As mention in the section 5.4.1, MATLAB CAMShift tracker can only track the targeted track object until frame number 50 (Figure 5.12) when partial occlusion happened to the object. Thus, the SFDA for the MATLAB CAMShift tracker is only 0.31, sharing the same result with the multi CAMShift agent object tracking system with two tracking agents.

Table 5.7: Tracking result on PETS 2009 view 001 S1 datasets by using a different number of "CAMShiftAgents"

Object Tracking System	SFDA
MATLAB CAMShift	0.31
Proposed MAS object tracking system with (Two (2) tracking agents	0.31
Proposed MAS object tracking system with (Ten (10) tracking agents	0.54
Proposed MAS object tracking system with (Twenty (20) tracking agents	0.20

In PETS 2009 view001 S1 video, the appearance of the targeted track object is more complicated than a single colour ball in the A1 video. Therefore, when the multi-agent system cannot find the optimum value for the array upper boundary and lower boundary in CAMShift tracker, the tracking result became poor. More object-tracking agents are needed for the multi CAMShift agent object tracking system to find the optimum parameters. In Table 5.7, the multi CAMShift agent object tracking system with ten tracking agents produces the best tracking result. The result of CAMShift agent tracker that uses 20 agents is shown in Figure E.1 until E8 in Appendix E. The result in Figure E.6 shows that the tracker could continue to track the object after the occlusion in Frame 50. However, during the second occlusion at Frame 98 (see Figure E.7) the tracker begins to track the wrong object. The error in the tracking in the frame 98 can be seen in the graph in Figure F.2 (in Appendix F). At frame 90 the track centroid in horizontal direction starts to recline whiles the ground truth in Figure F.1 shows the opposite.

When using 20 CAMShift agents, each agent is assigned an array upper boundary and a lower boundary value. The valid value is between 1 (inclusive) to 256 (inclusive). Therefore, when using too many tracking agents, the range of the array upper boundary and a lower boundary of each agent is very small as shown in Table 5.4. Therefore, none of the agents can produce good tracking result.

Lastly, Table 5.8 shows the tracking result on CAVIAR atrium dataset. Generally, the ranking of the trackers is identical to the previous result in Table 5.7, but each tracker shows slightly better performance. The result in Table 5.8 is better because the scenario in CAVIAR atrium dataset is less crowded than PETS 2009 view001 S1 video. The result from MATLAB CAMShift and the multi CAMShift agent object tracking system with two tracking agent are the same (SFDA=0.38). The result from the multi CAMShift agent object tracking system with ten tracking agents is the best while the system with 20 agents is the worst.

Table 5.8: Tracking result on CAVIAR atrium dataset

Object Tracking System	SFDA
MATLAB CAMShift	0.38
Proposed MAS object tracking system with (Two (2) tracking agents	0.38
Proposed MAS object tracking system with (Ten (10) tracking agents	0.63
Proposed MAS object tracking system with (Twenty (20) tracking agents	0.28

The frame by frame result of the CAMShift agent tracking system that uses ten tracking agents are shown in Figure E.9 until Figure E.16.

5.5 Agents Communication in JADE

One of the objectives of this research is to combine the spatial-temporal model tracker and appearance model tracker using a multi-agent system to handle occlusion problem in object tracking. The Section 5.4.1 of this chapter has shown that the developed system can handle occlusion problem in object tracking. This section discusses the communication of our multi-agent object tracking system. It is important in every multi-agent system

development report to show the interaction process among the agents to prove the system runs according to the design. The interaction and message passing between the agents are shown in Figure 5.14.

The system is divided into four parts, the object tracking manager agent (*video01*), the CAMShift object-tracking agents (*cam01*, *cam02, cam03, … cam$_N$),* where N can be 2 or 10 or 20), the evaluator agent and the Directory Facilitator (*df*) agent. The object tracking manager agent will act as services for loading the surveillance video into an Array List of pictures and then detect the initial location of the tracking subject. Then, the object tracking manager agent (*video01*) will request for the ID of available CAMShift object-tracking agents that are available in the system.

A request performed by the *video01* agent to *df* agent is shown in Figure 5.15. In JADE, the message passing process between agents is based on FIPA (Foundation for Intelligent Physical Agents) standard. In the request, a message consists of the sender id, receiver id and content. The content of the request in our system is shown in Figure 5.16.

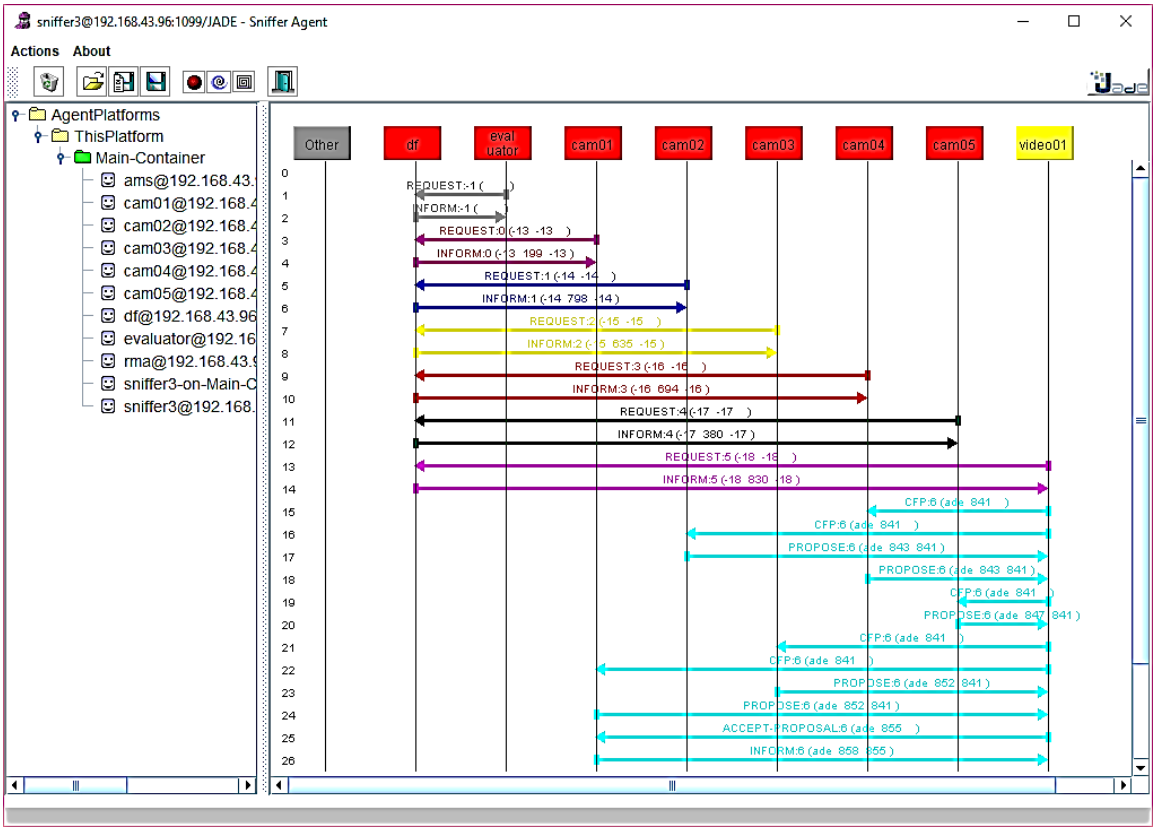


Figure 5.14: Interaction between *CAMShift* agents, *df* agents and *video01* agent.

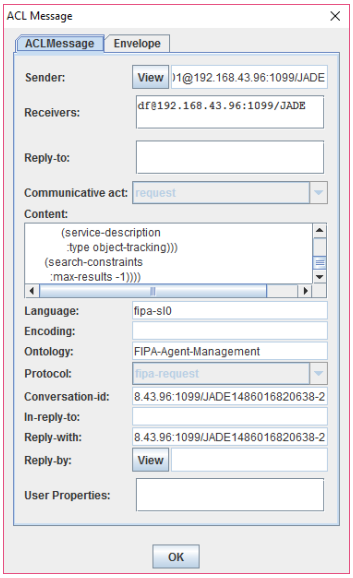


Figure 5.15: Message from Video agent to Directory Facilitator agent requesting for agent Id that provides object tracking service.

```
((action
   (agent-identifier
    :name df@192.168.43.96:1099/JADE
    :addresses (sequence http://LAPTOP-BA67H0PK:7778/acc))
   (search
    (df-agent-description
     :services
      (set
       (service-description
        :type object-tracking)))
    (search-constraints
     :max-results -1))))
```

Figure 5.16: Content of *Request* message from *video01* agent to *df* agent.

```
((result
   (action
    (agent-identifier
     :name df@192.168.43.96:1099/JADE
     :addresses (sequence http://LAPTOP-BA67H0PK:7778/acc))
    (search
     (df-agent-description
      :services
       (set
        (service-description
         :type object-tracking)))
     (search-constraints
      :max-results -1)))
   (sequence
    (df-agent-description
     :name
      (agent-identifier
       :name cam02@192.168.43.96:1099/JADE
       :addresses (sequence http://LAPTOP-BA67H0PK:7778/acc))
     :services
      (set
       (service-description
        :name camShiftTracker
        :type object-tracking)))
    (df-agent-description
     :name
      (agent-identifier
       :name cam01@192.168.43.96:1099/JADE
       :addresses (sequence http://LAPTOP-BA67H0PK:7778/acc))
     :services
      (set
       (service-description
        :name camShiftTracker
        :type object-tracking))))))
```

Figure 5.17: Content of *Inform* message from *df* agent to *video01* agent.

After a request is sent, the *video01* agent will wait for an *Inform* message from *df* agent. The content of the *Inform* message is shown in Figure 5.17. In the *Inform* message, the *video01* agent will know the name of the available agents which provide the object tracking service (i.e. *cam01* and *cam02*).

Next, the *video01* agent will use the names in the *Inform* message to send Call for Proposal (CFP) message to both object-tracking agents (*cam01* and *cam02*). Take note that the number of object-tracking agents in the system can either be 2, 10 or 20.

Object-tracking agents will use the Array List of pictures provided by the *video01* agent to produce a tracking result, and the result will be evaluated. The evaluation result from each object-tracking agent is then returned to the *video01* agent as proposal message. The *video01* agent will then evaluate the proposal and choose the best proposal as tracking result and send an accept proposal message to the best tracking agent. The tracking process ended when the best object tracking agent return *Inform* message to the video01 agent as shown in Figure 5.14.

5.6 Summary

At the beginning of this chapter, a set of simulation videos was design and generated to test the tracking capability of three popular trackers observed from a review of previous works. Experiments are conducted using the Kalman filter, Particle filter and Mean-shift tracker.

Sequence Frame Detection Accuracy was used to evaluate the tracking performance of each tracker. The results confirmed with most of the previous work from the literature review except that PF tracker in our experiment surprisingly failed to detect objects with arbitrary movement. The finding from the preliminary experiment also helped to identify

some of the limitations in the mean-shift tracker and become the motivation of this research.

The result from the multi CAMShift agent object tracking system is benchmarked with the result obtained from MATLAB CAMShift function. The multi CAMShift agent object tracking system show improvement over the MATLAB CAMShift object tracking system when it is used to track an object with full occlusion (simulation video A3). The SFDA achieved by the multi CAMShift agent object tracking system is 0.32, which is 3.5 times better than the result of the tracking is than MATLAB CAMShift where the SFDA is only 0.09. The multi CAMShift agent object tracking system also shows improvement when compared to MATLAB on tracking human (increment of 23% in SFDA in PETS 2009 S1 video and increment of 25% in SFDA in CAVIAR atrium video). The optimum setting when using multi CAMShift agent object tracking system is by using ten object tracking agents.

Lastly, the communication protocol in our multi-agent system is tested and verified using the sniffer agent tools in JADE to ensure the message flow is running properly.

CHAPTER 6
CONCLUSION AND LIMITATION

As mentioned in the introduction of this research, the main goal of our work is to design a method to select the inclusive lower boundary array and the inclusive upper boundary array for the colour histogram in CAMShift tracker. The proposed design with significant contribution to the academia (see Section 6.1) is built and tested with various video dataset to validate its accuracy and robustness. Although some limitations are discovered (see Section 6.2) but overall the objectives of this research are met with the proposed works.

6.1 Contributions

In this research, the focus of the research is on object detection and tracking in the crowded surveillance video. The object tracking result of the proposed method shows improvement when compared to the standard methods while keeping their fast performance during runtime. The summarised contribution follows here:

- A multi CAMShift agents object tracking system has been designed and implemented to automatically select the optimal values for the inclusive lower boundary array and the inclusive upper boundary array for the process of generating the colour histogram for CAMShift tracker to handle object occlusion. This research is the first research that improves CAMShift tracking result by using a system to select the optimum value for the inclusive lower boundary array and the inclusive upper boundary array. The requirement for the values varies from one situation to another due to the appearance of the targeted object, and its environment is different from one video to another. The multi CAMShift agent

object tracking system is suitable for situations, when no user input is possible and when the computational time is limited. When the time is limited, the user cannot incrementally train the trackers from every single image. So, the multi CAMShift agent object tracking system provides a solution to select these values automatically. The experiment result of the multi CAMShift agent object tracking system shows improvement when compared to CAMShift in MATLAB on tracking human (increment of 23% in SFDA in PETS 2009 S1 video and increment of 25% in SFDA in CAVIAR atrium video).

- Kalman filter has been successfully fused with the CAMShift to create the "CAMShiftAgent" in the multi CAMShift agent object tracking system. The "CAMShiftAgent" agent that combines the spatial-temporal approach and the appearance model approach to handle the full occlusion problem in object tracking. The framework of the multi CAMShift agent object tracking system is designed and successfully implemented on the JADE platform. The multi CAMShift agents object tracking system can track an object after a full occlusion in the experiment that uses the simulation video. The result (SFDA=0.32) obtained by the multi CAMShift agents object tracking system is 3.5 times better than the result of the CAMShift tracker MATLAB (SFDA=0.09). CAMShift in MATLAB does not use spatial-temporal to track the object.
- In the final part of the thesis, an experiment is planned and tested on the multi CAMShift agent object tracking system. The simulation video dataset, is designed to identify the strength and weaknesses of object tracking system. These simulated videos are ideal to provide a good indication of which algorithms work well in a given scenario because the videos are computer generated and free from noise such

as shadow. Using the simulation videos, the strength and the limitation of the multi CAMShift agent object tracking system are highlighted. When video A1 (moving ball with no occlusion) is used to test the proposed system, it achieves identical result with MATLAB, but when video A3 is used, the multi CAMShift agent object tracking system achieve the result which is 3.5 times better than MATLAB. The simulation video A3 confirmed the ability of the multi CAMShift agent object tracking system in tracking an object even after the object is fully occluded.

6.2 Future Work

This research has contributed to several areas of object tracking. However, there are still several areas future work could address. This research can be extended by engaging in investigations focusing on the following suggestions.

- When a tracked object exit a scene in a surveillance video. A mechanism could be added to the system to indicate the end of the tracking process when the target has exited the scene. Otherwise, like in our system, the tracker will look for another similar object in the scene to continue tracking. In our result analysis, the tracking was ended manually.
- More video dataset could be added as new dataset are made available in the future to the continuously evaluate the multi CAMShift agent object tracking system.
- Other tracking methods can be developed as agents and then add to the multi-agent environment to allow the multi CAMShift agent object tracking system to compete. This would create more variety of tracking agents in the environment and hopefully create a system that could track an object in videos all kinds of complex scenes.

REFERENCES

Akbari, O. Z. (2010). A Survey of Agent-Oriented Software Engineering Paradigm: Towards Its Industrial Acceptance. *Journal of Computer Engineering Research, 1*(12), 14-28.

Ali, A., & Mirza, S. M. (2006). Object Tracking Using Correlation, Kalman Filter and Fast Means Shift Algorithms. *2nd International Conference on Emerging Technologies.* Peshawar, Pakistan.

Ali, A., & Terada, K. (2009). A Framework for Human Tracking Using Kalman Filter and Fast Mean Shift Algorithms. *IEEE 12th International Conference on Computer Vision Workshops, 2009* (pp. 1028-1033). Kyoto: IEEE.

Ali, I., & Dailey, M. N. (2012). Multiple Human Tracking in High-Density Crowds. *Image and Vision Computing, 30*(12), 966–977.

Babu, R. V., & Makur, A. (2007). Kernel-Based Spatial-Color Modeling for Fast Moving Object Tracking. *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing). 1,* I.901-I.904. Honolulu: IEEE.

Baltzakis, H. (30 November, 2004). *Kalman/Particle Filters Tutorial.* Retrieved 2 August, 2017, from Institute of Computer Science of the Foundation for Research and Technology: https://www.ics.forth.gr/mobile/tutorial_kalman_particles.ppt

Batista, J. P. (2004). Tracking Pedestrians Under Occlusion Using Multiple Cameras. In A. Campilho, & M. Kamel, *Image Analysis and Recognition* (pp. 552-562). Berlin: Springer Berlin Heidelberg.

Baum, T., Izhaki, I., Rivlin, E., & Katzir, G. (2014). Active Tracking and Pursuit Under Different Levels of Occlusion: A Two-Layer Approach. *Machine Vision and Applications, 25*(1), 173-184.

Bedruz, R. A., Sybingco, E., Bandala, A., Quiros, A. R., Uy, A. C., & Dadios, E. (2017). Real-Time Vehicle Detection and Tracking Using a Mean-Shift Based Blob Analysis and Tracking Approach. *International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management,* (pp. 1-5). Manila: IEEE.

Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing Multi-Agent Systems With JADE.* West Sussex: John Wiley & Sons.

Benavidez, P., & Jamshidi, M. (2011). Mobile Robot Navigation and Target Tracking System. *6th International Conference on System of Systems Engineering, 2011* (pp. 299-304). Albuquerque, NM: IEEE.

Bernhardt, S. (2012). *Mean-Shift Video Tracking.* Retrieved April, 2012, from Matlab Central: https://www.mathworks.com/matlabcentral/fileexchange/35520-mean-shift-video-tracking

Besada, J. A., Garc´ıa, J., Portillo, J., Molina, J. M., Varona, A., & Gonzalez, G. (2005). Airport Surface Surveillance Based on Video Images. *IEEE Transactions on Aerospace and Electronic Systems, 41*(3), 1075-1082.

Beymer, D., & Konolige, K. (1999). Real-Time Tracking of Multiple People Using Continuous Detection. *IEEE Frame Rate Workshop,* 53.

Bradski, G. R. (1998). Computer Vision Face Tracking for Use in a Perceptual User Interface. *Intel Technology Journal, 2*, 1-15.

Branca, A., Attolico, G., & Distante, A. (2002). Cast Shadow Removing in Foreground Segmentation. *16th International Conference on Pattern Recognition, 2002. 1*, pp. 214-217. Quebec City: IEEE.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 203-236.

Camuñas-Mesa, L. A., Serrano-Gotarredona, T., Ieng, S.-H., Benosman, R., & Linares-Barranco, B. (2017). Event-Driven Stereo Visual Tracking Algorithm to Solve Object Occlusion. *IEEE Transactions on Neural Networks and Learning Systems*, 1-15.

Capi, G., Toda, H., & Nagasaki, T. (2010). A Vision Based Robot Navigation and Human Tracking for Social Robotics. *Robotic and Sensors Environments, 2010 IEEE International Workshop on*, (pp. 1-6). Phoenix, AZ.

Castanedo, F., García, J., Patricio, M. A., & Molina, J. M. (2011). A Multi-Agent Architecture Based on the BDI Model for Data Fusion in Visual Sensor Networks. *Journal of Intelligent & Robotic Systems, 62*(3-4), 299-328.

Čehovin, L., Leonardis, A., & Kristan, M. (2016). Visual Object Tracking Performance Measures Revisited. *Image Processing IEEE Transactions on, 25*, 1261-1274.

Cesa-Bianchi', N., & Lugosi', G. (2006). *Prediction, Learning, and Games.* New York: Cambridge University Press.

Chen, K., Song, K., Kyoungho, C., & Guo, Y. (2013). Optimized Meanshift Target Reference Model Based on Improved Pixel Weighting in Visual Tracking. *Journal of Electronics (China). 30*, 283-289.

Cho, J., Jin, S., Pham, X., & Jeon, J. (2006). Object Tracking Circuit Using Particle Filter With Multiple Features. *In SICE-ICASE, 2006. International Joint Conference*, (pp. 1431-1436).

Comaniciu, D., & Ramesh, V. (2000). Mean Shift and Optimal Prediction for Efficient Object Tracking. *Image Processing, 2000. Proceedings 2000 International Conference on (Volume 3)* (pp. 70-73). Vancouver, BC: IEEE.

Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-Based Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*(5), 564-577.

Computational Vision Group, Reading University. (14 Oct, 2007). *PETS 2007 Benchmark Data*. Retrieved from Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance: http://www.cvg.reading.ac.uk/PETS2007/data.html

Conte, D., Foggia, P., Percannella, G., & Vento, M. (2010). Performance Evaluation of a People Tracking System on PETS 2009 Database. *Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance, 2010* (pp. 119-126). IEEE.

Coşkun, M., & Ünal, S. (2016). Implementation of Tracking of a Moving Object based on Camshift Approach With a UAV. *Procedia Technology, 22*, 556-561.

Cropley, J. (1 2, 2016). *Top Video Surveillance Trends for 2016.* Retrieved 14 Feb, 2017, from IHS Technology: https://technology.ihs.com/520142

Cucchiara, R., Grana, C., Piccardi, M., Prati, A., & Sirotti, S. (August, 2001). Improving Shadow Suppression in Moving Object Detection With HSV Color Information. *Intelligent Transportation Systems, 2001. Proceedings. 2001* (pp. 334-339). Oakland, CA: IEEE.

Deguchi, K., Kawanaka, O., & Okatani, T. (2004). Object Tracking by the Mean-Shift of Regional Color Distribution Combined With the Particle-Filter Algorithms. *17th International Conference on Pattern Recognition, 2004. 3*, pp. 506-509. IEEE.

DeLoach, S. A. (2001). Analysis and Design Using MaSE and agentTool . *12th Midwest Artificial Intelligence and Cognitive Science Conferece* (pp. 1-7). Oxford: Miami University.

Dockstader, S., & Tekalp, A. (2001). Multiple Camera Tracking of Interacting and Occluded Human Motion. *Proceedings of the IEEE, 89*(10), 1441-1455.

Duan, Z., Cai, Z., & Yu, J. (2009). Occlusion Detection and Recovery in Video Object Tracking Based on Adaptive Particle Filters. *Control and Decision Conference, 2009. CCDC '09.* (pp. 466-469). Guilin: IEEE.

E. Cuevas, D. Z. (10 August, 2005). *Kalman Filter for Vision Tracking.* Retrieved 2017 October, 2017, from Dissertationen Online Der Freien Universität Berlin:

http://www.diss.fu-berlin.de/docs/servlets/MCRFileNodeServlet/FUDOCS_ derivate_000000000473/2005_12.pdf

Elgammal, A., Duraiswami, R., Harwood, D., & Davis, L. S. (2002). Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance. *Proceedings of IEEE, 90*(7), 1151-1163.

FarrokhNejad, R. (12 Mar, 2007). *2D Target Tracking Using Kalman Filter*. Retrieved Mar, 2012, from MATLAB Central: http://www.mathworks.com/matlabcentral/fileexchange/14243-2d-target-tracking-using-kalman-filter

Fekir, A., & Benamrane, N. (2015). Multi Agent System for Boundary Detection and Object Tracking in Image Sequence Based on Active Contours. *Multiagent and Grid Systems, 11*(2), 81-93.

Fisher, R. (2001). *CAVIAR Video Sequence Ground Truth.* Retrieved 31 1, 2017, from CAVIAR: Context Aware Vision Using Image-based Active Recognition: https://homepages.inf.ed.ac.uk/rbf/CAVIAR/gt.htm

Fisher, R. B. (2005). *CAVIAR Test Case Scenarios*. Retrieved from CAVIAR: Context Aware Vision Using Image-based Active Recognition: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/caviar.htm

Gascueña, J. M., Fernández-Caballero, A., López, M. T., & Delgado, A. E. (2011). Knowledge Modeling Through Computational Agents: Application to Surveillance Systems. *Expert Systems, 28*(4), 306-323.

Ghandehari, E., Saadatjoo, F., & Chahooki, Z. (2015). Method Integration: An Approach to Develop Agent Oriented Methodologies. *Journal of Artificial Intelligence and Data Mining*, *3*(1), 59-76.

Godil, A., Bostelman, R., Shackleford, W., Hong, T., & Shneier, M. (2014). *Performance Metrics for Evaluating Object and Human Detection and Tracking Systems.* Retrieved 1 September, 2017, from semanticscholar.org: https://pdfs.semanticscholar.org/d051/ba0a904e4b4c45a2af145aa29b8490bbbc5c.pdf

Greenhill, D., Renno, J., Orwell, J., & Jones, G. A. (March, 2008). Occlusion Analysis: Learning and Utilising Depth Maps in Object Tracking. *Image Vision Comput, 26*(3), 430-441.

Guha, P., Mukerjee, A., & Subramanian, V. K. (2011). Formulation, Detection and Application of Occlusion States (Oc-7) in the Context of Multiple Object Tracking. *Advanced Video and Signal-Based Surveillance, 2011 8th IEEE International Conference on* (pp. 191-196). Klagenfurt: IEEE.

Guha, P., Mukerjee, A., & Venkatesh, K. S. (2005). Efficient Occlusion Handling for Multiple Agent Tracking by Reasoning With Surveillance Event Primitives. *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, (pp. 49-56).

Guo, L., Li, L., Zhao, Y., & Zhao, Z. (2016). Pedestrian Tracking Based on CAMShift With Kalman Prediction for Autonomous Vehicles. *International Journal of Advanced Robotic Systems*, *13*(3), 120-128.

Han, Z., Zhang, R., Wen, L., Xie, X., & Li, Z. (2016). Moving Object Tracking Method Based on Improved Camshift Algorithm. *International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration* (p. 92). Wuhan: IEEE.

Han, Z., Zhang, R., Wen, L., Xie, X., & Li, Z. (2016). Moving Object Tracking Method Based on Improved Camshift Algorithm. *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration* (pp. 91-95). Wuhan: IEEE.

Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W4: Real-time Surveillance of People and Their Activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 809-830.

Harville, M., Gordon, G., & Woodfill, J. (2001). Adaptive Video Background Modeling Using Color and Depth. *Image Processing, 2001. Proceedings. 2001 International Conference on*, *3*, pp. 90-93. Thessaloniki.

Hedayati, M., Wan Zaki, W. M., & Hussain, A. (2010). Real-Time Background Subtraction for Video Surveillance: From Research to Reality. *2010 6th International Colloquium on Signal Processing & Its Applications*, (pp. 1-6). Mallaca City.

Huang, K., Tao, D., Yuan, Y., Li, X., & Tan, T. (2011). Biologically Inspired Features for Scene Classification in Video Surveillance. *Systems, Man, and Cybernetics, Part B: Cybernetics, 41*(1), 307-313.

IEEE Computer Society. (2009). *Performance Evaluation of Tracking and Surveillance 2009*. Retrieved from PETS 2009: Benchmark data: http://www.cvg.rdg.ac.uk/PETS 2009/a.html

Isard, M., & MacCormick, J. (2001). BraMBLe: A Bayesian Multiple-blob Tracker. *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, 2*, pp. 34-41.

Ivanov, Y., & Bobick, A. (1999). Recognition of Multi-agent Interaction in Video Surveillance. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* (pp. 169-176). Kerkyra: IEEE.

Javed, O., Shafique, K., & Shah, M. (2002). A Hierarchical Approach to Robust Background Subtraction Using Color and Gradient Information. *Motion and Video Computing, 2002, Workshop on* (pp. 22-27). IEEE.

Juan, T., & Sterling, L. (2003). The ROADMAP Meta-Model for Intelligent Adaptive Multi-Agent Systems in Open Environments. *Lecture Notes in Computer Science, 2935*, 53-68.

Juan, T., Pearce, A., & Sterling, L. (2002). ROADMAP: Extending the Gaia Methodology for Complex Open Systems. *In Proceedings of the first international joint conference on Autonomous agents and multiagent systems* (pp. 3-10). Bologna: ACM.

Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active Contour Model. *International Journal of Computer Vision, 1*(4), 321-331.

Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., et al. (2009). Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(2), 319–336.

Kim, I. S., Choi, H. S., Yi, K. M., Choi, J. Y., & Kong, S. G. (2010). Intelligent Visual Surveillance - A Survey. *International Journal of Control, Automation, and Systems, 8*(5), 926-939.

Kleeman, L. (1996). *Understanding and Applying Kalman Filtering.* Retrieved 1 August, 2017, from Biorobotics Laboratory, Carnegie Mellon University: http://biorobotics.ri.cmu.edu/papers/sbp_papers/integrated3/kleeman_kalman_basics.pdf

Kou, W., Wang, Y., Guo, J., Zhang, H., & Liu, S. (2017). A Design of Detection and Tracking System Based on Dynamic Multi-object. *Technology, Networking, Electronic and Automation Control Conference, 2017 IEEE 2nd Information* (pp. 1-6). Chengdu: IEEE.

Lee, B. Y., Liew, L. H., Cheah, W. S., & Wang, Y. C. (2012). Simulation Videos for Understanding Occlusion Effects on Kernel Based Object Tracking. In S.-S. Yeo, Y. Pan, Y. S. Lee, & H. B. Chang, *Computer Science and its Applications*, 203, 139-147. Netherlands: Springer Netherlands.

Lee, B. Y., Liew, L. H., Cheah, W. S., & Wang, Y. C. (2014). Occlusion Handling in Videos Object Tracking: A Survey. *8th International Symposium of the Digital Earth* (pp. 1-6). Kuching: IOP Publishing.

Lee, D., Kim, J., & Sohn, J. (2011). Modified Mean-Shift for Head Tracking. *8th International Conference on Ubiquitous Robots and Ambient Intelligence 2011* (pp. 848-849). Incheon: IEEE.

Li, J., Zhou, X., Chen, S., Chan, S., & Ju, Z. (2017). Robust Object Tracking via Structure Learning and Patch Refinement in Handling Occlusion. *Intelligent Robotics and Applications. 2017* (pp. 449-459). Wuhan: Springer, Cham.

Li, W., & Li, X. (2016). Multiple Object Tracking Based on Modified Algorithm of GMMCP Tracker. *IEEE International Conference on Signal and Image Processing* (pp. 11-15). Beijing: IEEE.

Li, Y., Bloch, I., & Shen, W. (2015). Handling Inter-object Occlusion for Multi-object Tracking Based on Attraction Force Constraint. *Lecture Notes in Computer Science, vol 9164. Spring. 9164,* pp. 520-527. Niagara Falls: Springer, Cham.

Li, Y., Cao, X., Liu, J., & Zhang, B. (2017). CNN-Detector-Based Multiple Homogeneous Objects Tracking Under Stochastic Wide-Range Occlusions. *2017 IEEE Global Conference on Signal and Information Processing* (pp. 808-812). Montreal: IEEE.

Li, Y., Wang, Y., Qi, Y., & Li, H. (2010). Multi-Agent Based Particle Filter for Moving Object Tracking. *Computer Application and System Modeling, 2010 International Conference on.* Taiyuan.

Liang, N. G. (2012). An Improved Object Tracking Method Based on Particle Filter. *In Consumer Electronics, Communications and Networks, 2012 2nd International Conference on,* (pp. 3107-3110).

Lien, S.-F., Hsia, K.-H., & Su, J.-P. (2015). Moving Target Tracking Based on Camshift Approach. *Applied Mathematics & Information Sciences*, *7*(1), 395-401.

Liu, C., Hu, C., & Aggarwal, J. (2011). Eigenshape Kernel Based Mean Shift for Human Tracking. *IEEE International Conference on Computer Vision Workshops (ICCV Workshops) 2011* (pp. 1809-1816). Barcelona: IEEE.

Lu, M. V. (2013). Reusing Negotiation Strategy for Multi-agent System Development. *Master thesis, Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak.*

Luo, Y., Sterling, L., & Taveter, K. (2007). Modelling a Smart Music Player With a Hybrid Agent-oriented Methodology. *The 15th International Requirements Engineering Conference* (pp. 281-286). Delhi, India: IEEE.

Ma, Y., & Chen, Q. (2010). Depth Assisted Occlusion Handling in Video Object Tracking. In *Advances in Visual Computing* (pp. 449-460). Berlin: Springer Berlin Heidelberg.

Maccormick, J., & Blake, A. (2000). A Probabilistic Exclusion Principle for Tracking Multiple Objects. *International Journal of Computer Vision*, *39*(1), 57-71.

Marchesotti, L., Piva, S., & Regazzoni, C. (2003). An Agent-based Approach for Tracking People in Indoor Complex Environments. *Proceedings of 12th International Conference on Image Analysis and Processing* (pp. 99-102). Mantova: IEEE.

Marti-Puig, P., Rodríguez, S., Paz, J. F., Reig-Bolaño, R., Rubio, M. P., & Bajo, J. (2012). Stereo Video Surveillance Multi-agent System: New Solutions for Human Motion Analysis. *Journal of Mathematical Imaging and Vision, 42*(2-3), 176-195.

Mathur, G., & Bundele, M. (2016). Research on Intelligent Video Surveillance Techniques for Suspicious Activity Detection Critical Review. *International Conference on Recent Advances and Innovations in Engineering* (pp. 1-8). Jaipur: IEEE.

MathWorks. (12 Nov, 2016). *MATLAB vs. Python: Top Reasons to Choose MATLAB.* Retrieved 10 Jan, 2017, from MATLAB: https://www.mathworks.com/products/matlab/matlab-vs-python.html

Meng, J., Yuan, J., Tan, Y.-P., & Wang, G. (2015). Fast Object Instance Search in Videos From One Example. *IEEE International Conference on Image Processing 2015* (pp. 4381-4385). Quebec City: IEEE.

Milan, A. (2016). *Anton Milan - Multi-Target Tracking Ground Truth and Detections.* Retrieved 31 1, 2017, from Anton Milan: http://www.cvg.reading.ac.uk/PETS 2009/

Mirabi, M., & Javadi, S. (2012). People Tracking in Outdoor Environment Using Kalman Filter. *2012 Third International Conference on Intelligent Systems Modelling and Simulation* (pp. 303-307). IEEE.

Mozerov, M., Amato, A., Roca, X., & González, J. (2009). Solving the Multi Object Occlusion Problem in a Multiple Camera Tracking System. *Pattern Recognition and Image Analysis, 19*(1), 165-171.

Myint, H., Wong, P., Dooley, L., & Hopgood, A. (2016). Tracking a Table Tennis Ball for Umpiring Purposes Using a Multi-agent System. *The Proceedings of 20th International Conference on Image Processing, Computer Vision, & Pattern Recognition* (pp. 119-125). Las Vegas: CSREA Press.

Nghiem, A. T., Bremond, F., Thonnat, M., & Valentin, V. (2007). ETISEO, Performance Evaluation for Video Surveillance Systems. *IEEE Conference on Advanced Video and Signal Based Surveillance 2007* (pp. 476-481). Washington: Association for Computing Machinery.

Nie, W., Liu, A., & Su, Y. (2012). Multiple Person Tracking by Spatiotemporal Tracklet Association. *IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*, (pp. 481-486). Beijing.

Ojha, S., & Sakhare, S. (2015). Image Processing Techniques for Object Tracking in Video Surveillance - A Survey. *International Conference on Pervasive Computing* (pp. 1-6). Pune: IEEE.

OpenCV. (Jun, 2017). *Meanshift and Camshift*. Retrieved 30 Jun, 2017, from OpenCV: http://docs.opencv.org/trunk/db/df8/tutorial_py_meanshift.html

Padgham, M., & Winikoff, L. (2005). *Prometheus: A Practical Agent Oriented Methodology*. Hershey, USA: Idea Group.

Pal, D. H., & Kakade, S. M. (2016). Dynamic Hand Gesture Recognition Using KINECT Sensor. *International Conference on Global Trends in Signal Processing, Information Computing and Communication, 2016* (pp. 448-453). Jalgaon: IEEE.

Paris, S. (2007). *Particle Filter Color Tracker*. Retrieved Mar, 2014, from MATLAB Central: http://www.mathworks.com/matlabcentral/fileexchange/17960-particle-filter-color-tracker

Pérez, Ó., García, J., Berlanga, A., & Molina, J. M. (2005). Adjustment of Surveillance Video Systems by a Performance Evaluation Function. In Mira, J. & Álvarez, J. (Eds), *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach. IWINAC 2005. Lecture Notes in Computer Science. 3562*, pp. 499-508. Berlin, Heidelberg: Springer.

Pulford, G. W. (2005). Taxonomy of Multiple Target Tracking Methods. *Radar, Sonar and Navigation, IEE Proceedings, 152*(5), 291-304.

Radu, S. (2013). *An Adaptive Negotiation Multi-Agent System for e-Commerce Applications. PhD thesis.* Bucharest: Faculty of Automatic Control and Computers, University Politehnica of Bucharest.

Raman, R., Sa, P., & Majhi, B. (2012). Occlusion Prediction Algorithms for Multi-camera Network. *Sixth International Conference on Distributed Smart Cameras* (pp. 1-6). Hong Kong: IEEE.

Rao, A., & Georgef, M. (1995). BDI Agents: From Theory to Practice. *1st International Conference on MultiAgent* (pp. 312–319). San Francisco, Calif, USA: The MIT Press.

Rasid, L. N., & Saudi, S. A. (2010). Versatile Object Tracking Standard Database for Security Surveillance. *10th International Conference on Information Sciences Signal Processing and their Applications* (pp. 782-785). Kuala Lumpur: IEEE.

Rasid, L. N., & Suandi, S. A. (2010). Versatile Object Tracking Standard Database for Security Surveillance. *Tenth International Conference on Information Science, Signal Processing and Their Applications* (pp. 10-13). Kuala Lumpur: IEEE.

Rathnayaka, P., Baek, S.-H., & Park, S.-Y. (2017). An Efficient Calibration Method for a Stereo Camera System With Heterogeneous Lenses Using an Embedded Checkerboard Pattern. *Journal of Sensors, 1*, 1-12.

Rautaray, S. S., & Agrawal, A. (2012). Design of Gesture Recognition System for Dynamic User Interface. *IEEE International Conference on Technology Enhanced Education.* Kerala: IEEE.

Ren, J., & Hao, J. (2012). Mean Shift Tracking Algorithm Combined With Kalman Filter. *5th International Congress on Image and Signal Processing* (pp. 727-730). IEEE.

Salhi, A., & Yengui Jammoussi, A. (2012). Object Tracking System Using Camshift, Meanshift. *International Journal of Electronics and Communication Engineering, 64*, 421-426.

Sánchez, S., Rodríguez, S., De la Prieta, F., De Paz, J. F., & Bajo, J. (2015). Multi-agent System for Tracking and Classification of Moving Objects. *Conference Proceedings of Distributed Computing and Artificial Intelligence, 12th International Conference* (pp. 63-74). Cham: Springer International Publishing.

Sattar, J., & Dudek, G. (2006). On the Performance of Color Tracking Algorithms for Underwater Robots Under Varying Lighting and Visibility. *Proceedings of 2006 IEEE International Conference on Robotics and Automation 2006* (pp. 3550-3555). Orlando: IEEE.

Scott, D. (1992). *Multivariate Density Estimation.* Houston, Texas: Wiley.

Shilpa, S. M. (2016). Masking Selected Region of Moving Object Using RGB Masking Technique. *2nd International Conference on Applied and Theoretical Computing and Communication Technology 2016* (pp. 376-380). Bangalore: IEEE.

Shu, G., Dehghan, A., Oreifej, O., Hand, E., & Shah, M. (2012). Part-based Multiple-Person Tracking With Partial Occlusion Handling. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1815-1821). Providence, RI, USA: IEEE.

Sterling, L. S., & Taveter, K. (2009). *The Art of Agent-oriented Modelling.* Cambridge, London, England.: The MIT Press.

Suarez, J., & Murphy, R. R. (2012). Hand Gesture Recognition With Depth Images: A Review. *The 21st IEEE International Symposium on Robot and Human Interactive Communication* (pp. 9-13). Paris: IEEE.

Tang, D., & Zhang, Y.-J. (2011). Combining Mean-shift and Particle Filter for Object Tracking. *Sixth International Conference on Image and Graphics* (pp. 771-776). Hefei, Anhui: IEEE.

Tao, H. (2006). *Object Tracking and Kalman Filtering.* Retrieved 24 12, 2016, from Course Web Pages: https://classes.soe.ucsc.edu/cmpe264/Fall06/Lec15.pdf

Tao, H., & Yu, Y. (2012). Finger Tracking and Gesture Recognition With Kinect, (pp. 214-218). Chengdu.

Taveter, K., & Wagner, G. (2005). Towards Radical Agent-oriented Software Engineering Processes Based on AOR Modelling. *Agent-oriented methodologies, 10*, 277-316.

Taylor, G. R., Chosak, A. J., & Brewe, P. C. (2007). OVVV: Using Virtual Worlds to Design and Evaluate Surveillance Systems. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (pp. 1-8).

Ulusoy, M. (1 Jan, 2017). *Understanding Kalman Filters, Part 4: An Optimal State Estimator Algorithm.* Retrieved 1 October, 2017, from Videos and Webinars, Mathworks: https://www.mathworks.com/videos/understanding-kalman-filters-part-4-optimal-state-estimator-algorithm--1493129749201.html

van Lon, R., & Holvoet, T. (October, 2016). *When Do Agents Outperform Centralized Algorithms? A Systematic Empirical Evaluation in Logistics.* Retrieved from Lirias.kuleuven: https://lirias.kuleuven.be/handle/123456789/553845

Wang, H., Huo, L., & Zhang, J. (2011). Target Tracking Algorithm Based on Dynamic Template and Kalman Filter. *Communication Software and Networks* (pp. 330-333). IEEE.

Wen, Z.-Q., & Cai, Z.-X. (2006). Mean Shift Algorithm and Its Application in Tracking of Objects. *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics* (pp. 4024-4028). Dalian: IEEE.

Wooldridge, M. (2002). *An Introduction to Multiagent System.* West Sussex: John Wiley & Sons Ltd.

Wooldridge, M., & Ciancarini, P. (2001). Agent-oriented Software Engineering: The State of the Art. *Proceedings of the first international workshop on agent-oriented software engineering* (pp. 1-28). New York: Berlin Springer.

Wu, C.-W., Zhong, M.-T., Tsao, Y., Yang, S.-W., Chen, Y.-K., & Chien, S.-Y. (2017). Track-clustering Error Evaluation for Track-based Multi-camera Tracking System Employing Human Re-identification. *Computer Vision and Pattern Recognition Workshops (CVPRW) 2017 IEEE Conference on* (pp. 1416-1424). Honolulu: IEEE.

Yao, C., BoJunxing, Zhang, C., & LiuQiang. (2017). Online Visual Tracking Based on Subspace Representation With Continuous Occlusion Modeling. *Multimedia Systems*, 357–368.

Yao, R., Xia, S., Shen, F., Zhou, Y., & Niu, Q. (May, 2016). Exploiting Spatial Structure From Parts for Adaptive Kernelized Correlation Filter Tracker. *IEEE Signal Processing Letters, 23*(5), 658 - 662.

Yilmaz, A., Javed, O., & Shah, M. (December, 2006). Object Tracking: A survey. *ACM Computing Surveys (CSUR), 38*(4), 1-45.

Yin, M., Bo, Y., Zhao, G., & Zou, W. (2013). Occluded Object Tracking Based on Bayesian Decision Theory and Particle Filtering. *32nd Chinese Control Conference* (pp. 3834 - 3838). Xi'an: IEEE.

Yun, W.-h., Kim, D., & Lee, J. (2010). Person Following With Obstacle Avoidance Based on Multi-layered Mean Shift and Force Field Method. *IEEE International Conference on Systems Man and Cybernetics 2010* (pp. 3813 - 3816). Istanbul: IEEE.

Zhang, T., Yang, Z., Zhang, X., & Shen, Y. (2016). A Vision System for Multi-rotor Aircraft to Track Moving Object. *Guidance, Navigation and Control Conference, IEEE Chinese* (pp. 401-406). Nanjing, China: IEEE.

Zhang, X., Zhang, X., Wang, Y., & Yu, H. (2017). Extended Social Force Model-based Mean Shift for Pedestrian Tracking Under Obstacle Avoidance. *IET Computer Vision, 11*(1), 1-9.

Zhang, Z., & Wong, K. H. (2014). Pyramid-based Visual Tracking Using Sparsity Represented Mean Transform. *Computer Vision and Pattern Recognition 2014 IEEE Conference on* (pp. 1226-1233). Columbus: IEEE.

Zhao, H., Xiang, K., Cao, S., & Wang, X. (2016). Robust Visual Tracking Via Camshift and Structural Local Sparse Appearance Model. *Journal of Visual Communication and Image Representation, 34*, 176-186.

Zhao, J., Qiao, W., & Men, G.-Z. (2009). An Approach Based on Mean Shift and Kalman Filter for Target Tracking Under Occlusion. *International Conference on Machine Learning and Cybernetics (Volume 4)* (pp. 2058-2062). Baoding: IEEE.

Zhu, Z., Ji, Q., Fujimura, K., & Lee, K. (2002). Combining Kalman Filtering and Mean Shift for Real Time Eye Tracking Under Active IR Illumination. *Proceedings of 16th International Conference on Pattern Recognition (Volume 4)* (pp. 318-321). IEEE.

APPENDICES

Appendix A:
Kalman Filter Object Tracking Result Frame by Frame

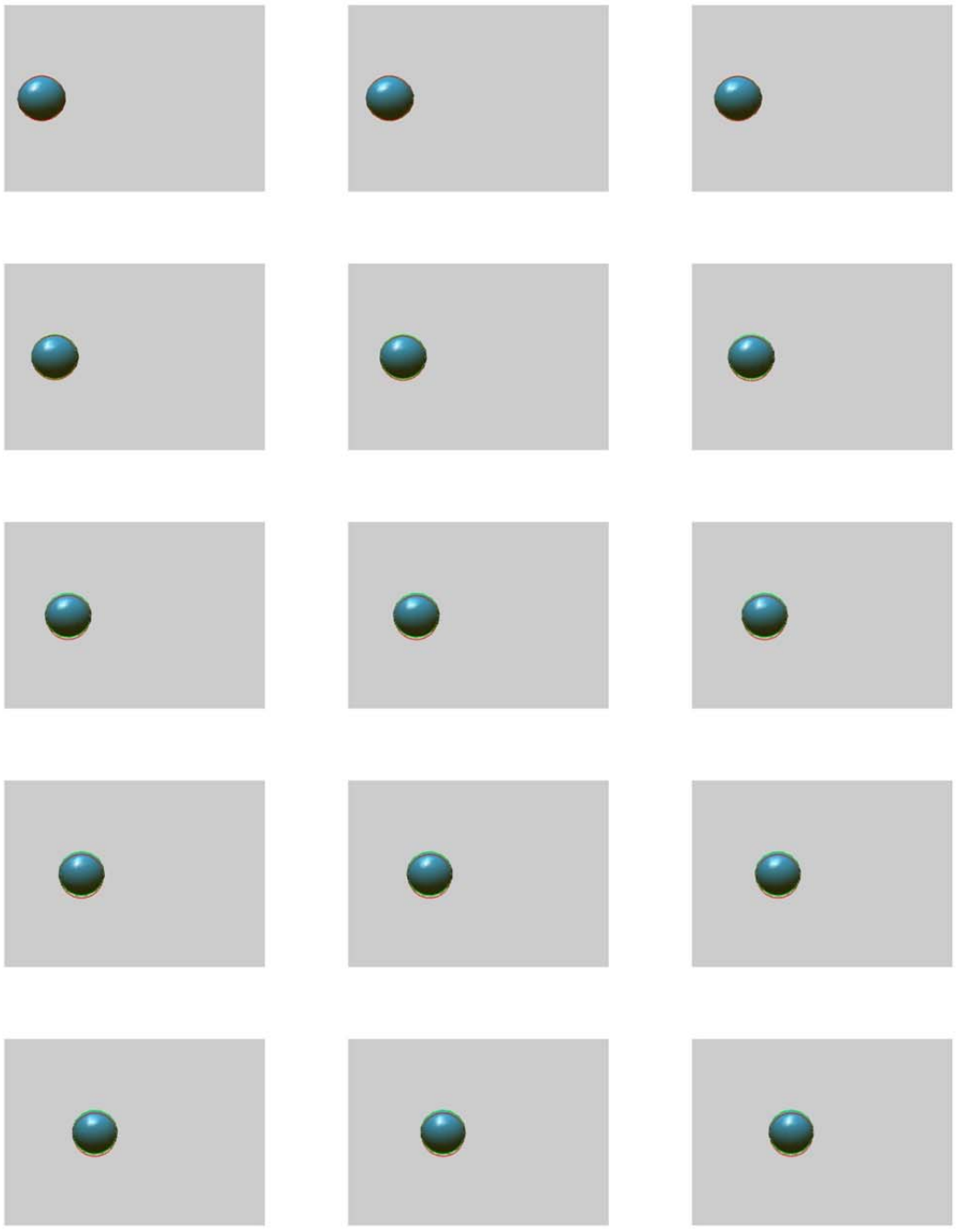


Figure A.1: Kalman filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A1.

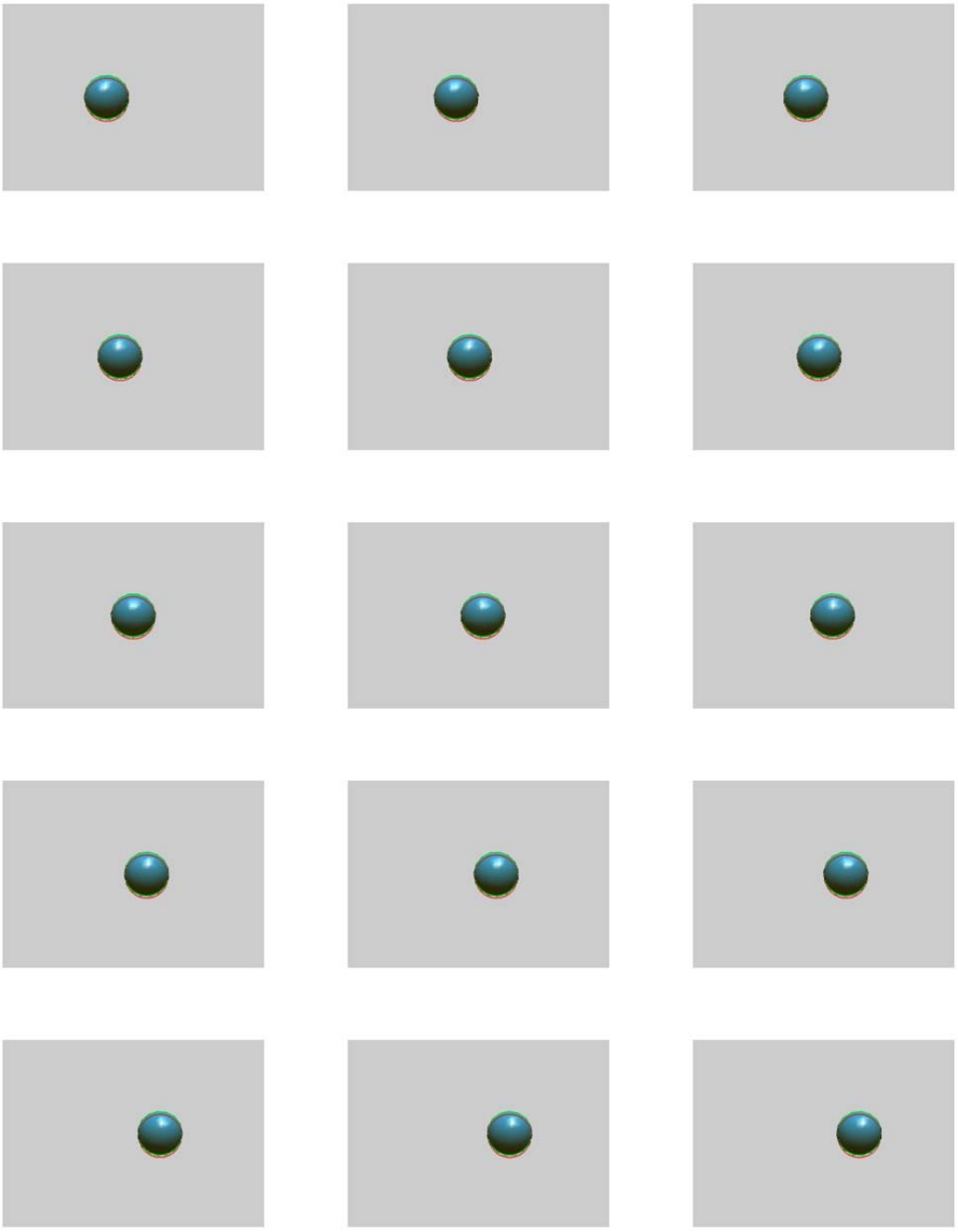


Figure A.2: Kalman filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A1.

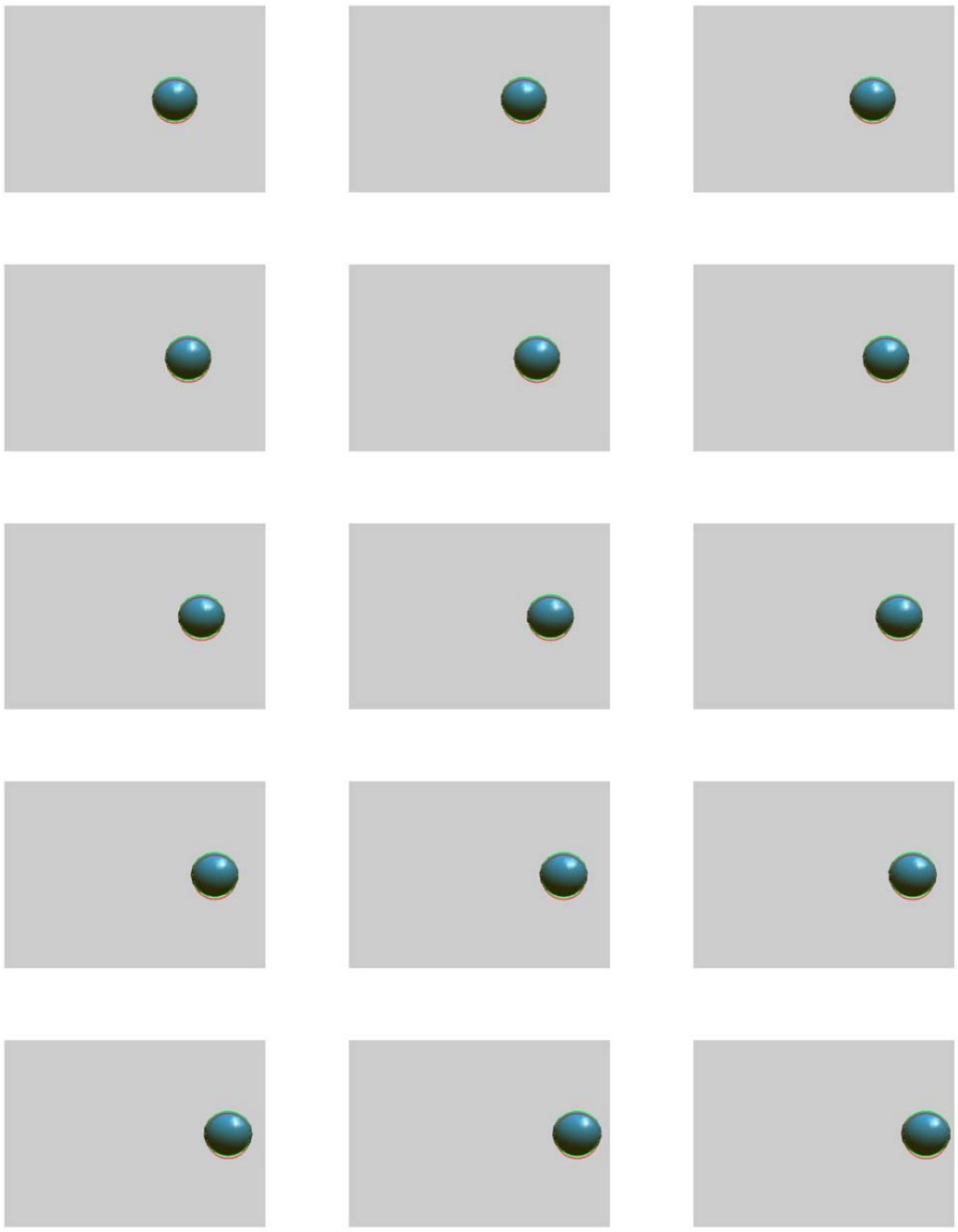


Figure A.3: Kalman filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A1.

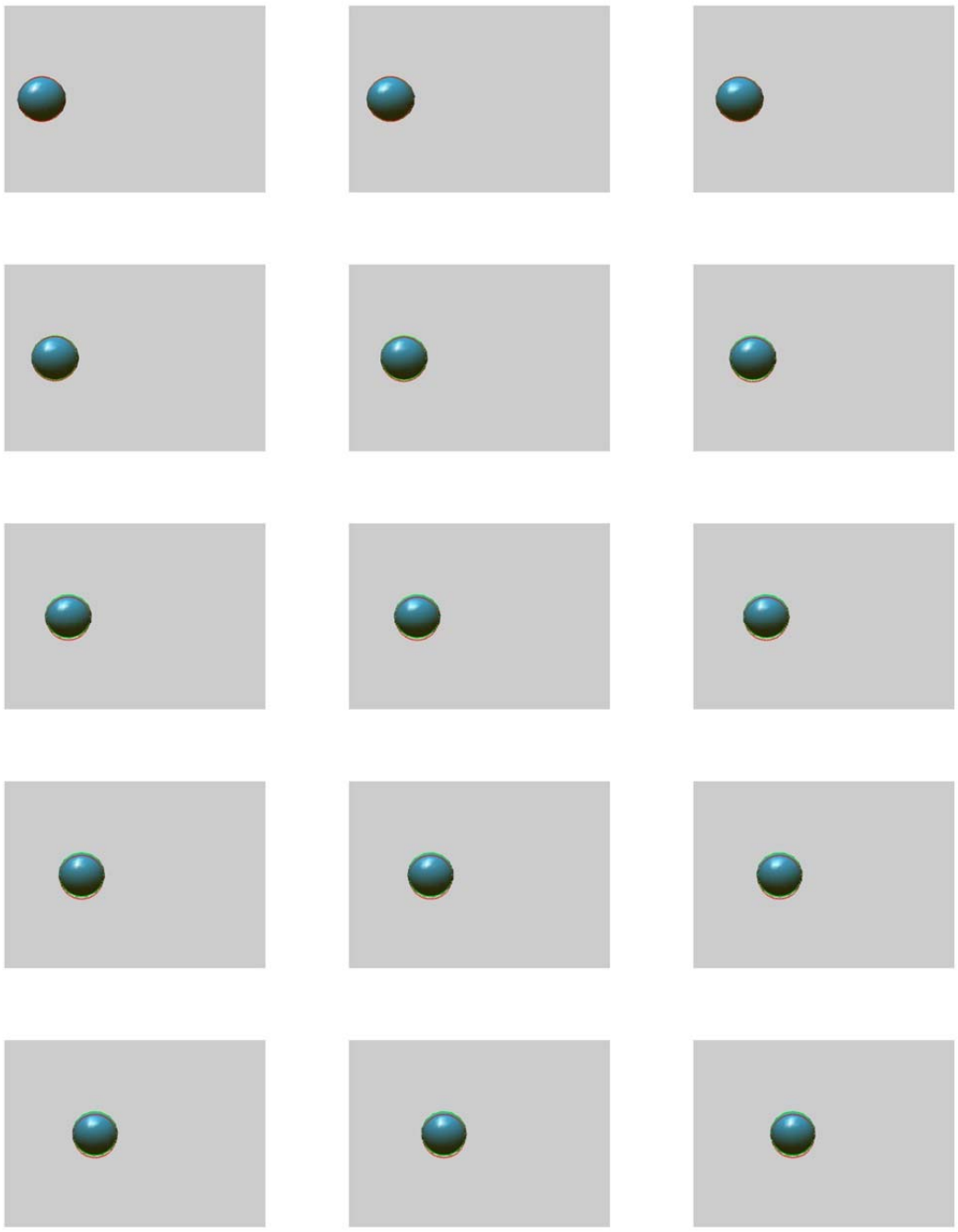


Figure A.4: Kalman filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A2.

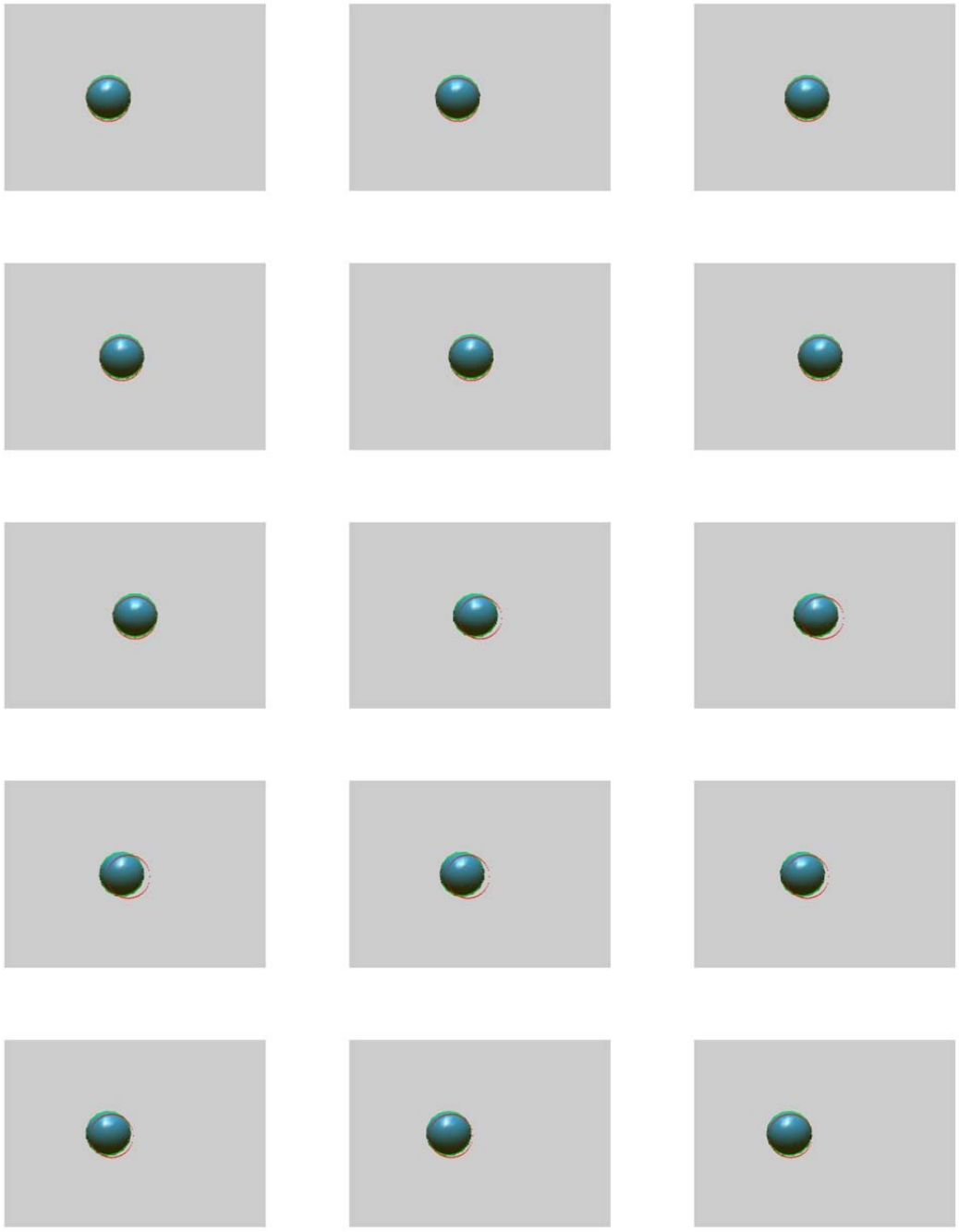


Figure A.5: Kalman filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A2.

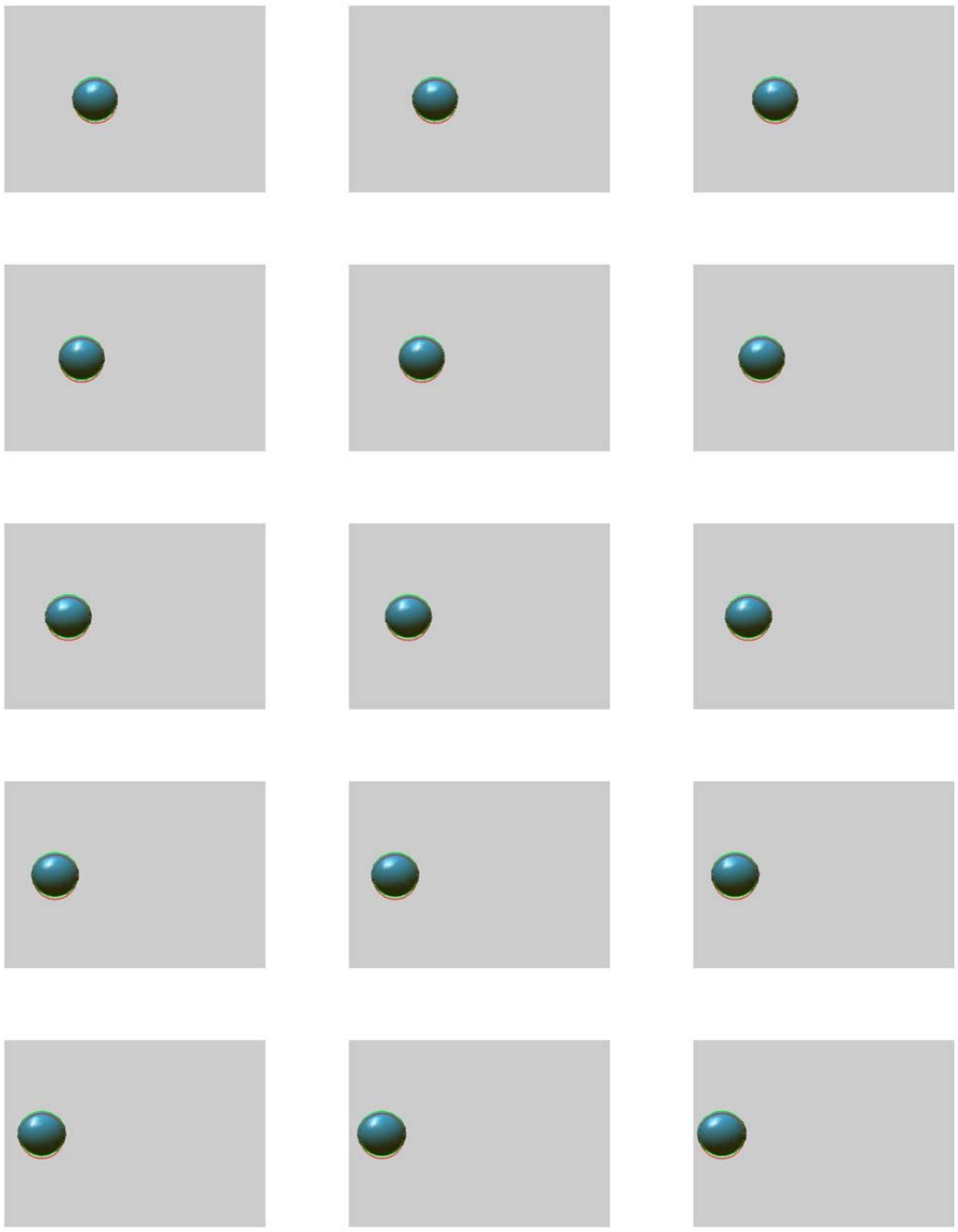


Figure A.6: Kalman filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A2.

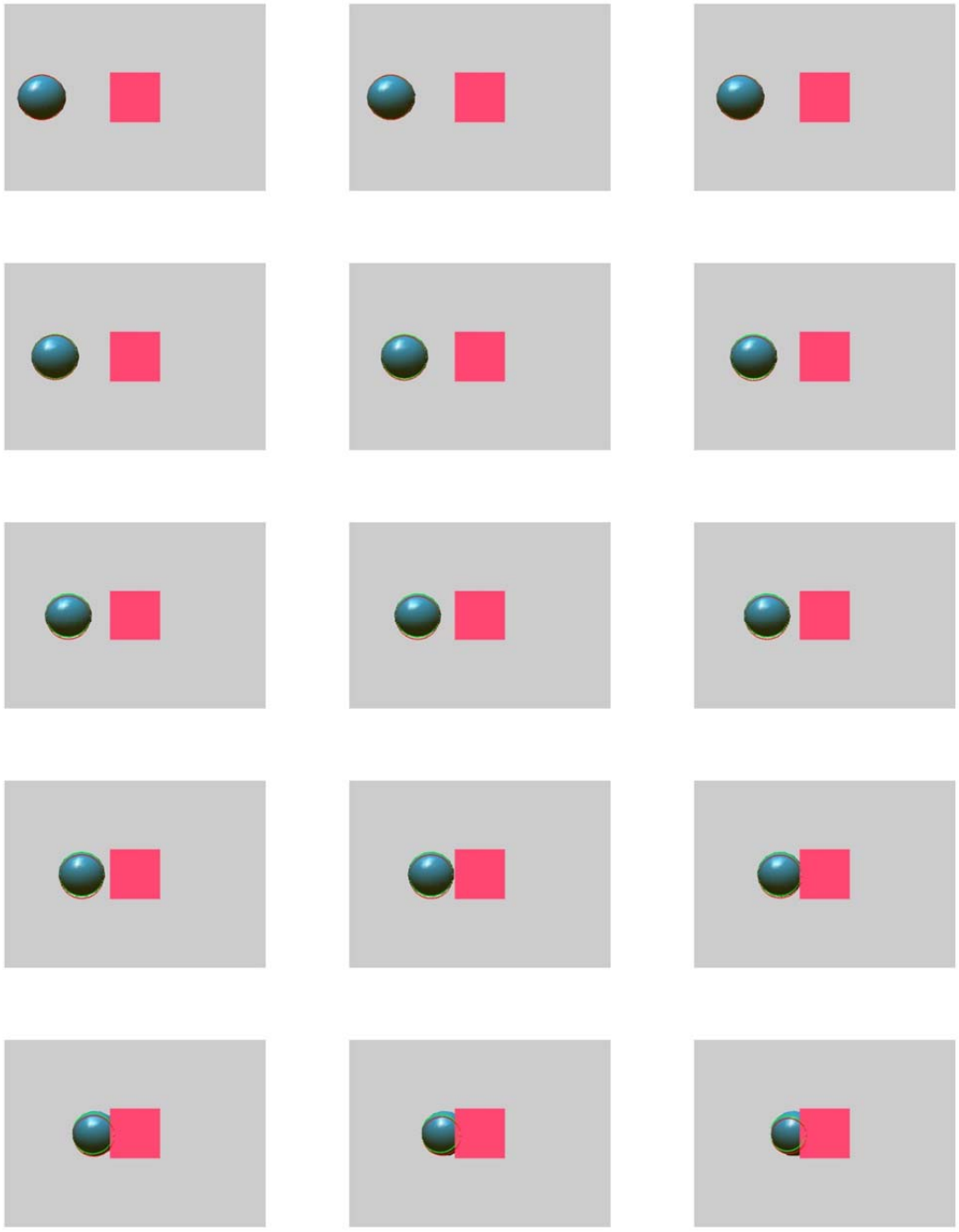


Figure A.7: Kalman filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A3.

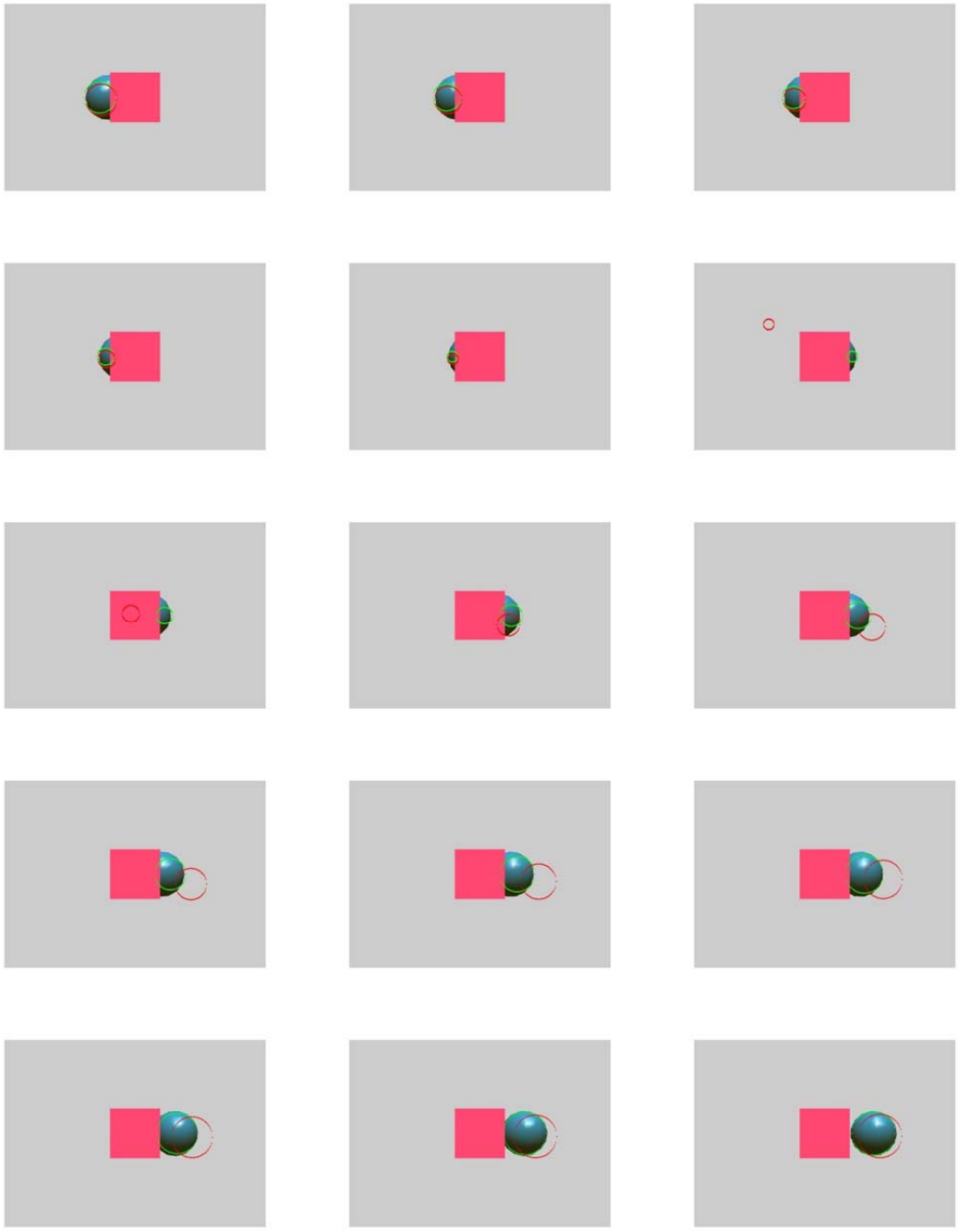


Figure A.8: Kalman filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A3.

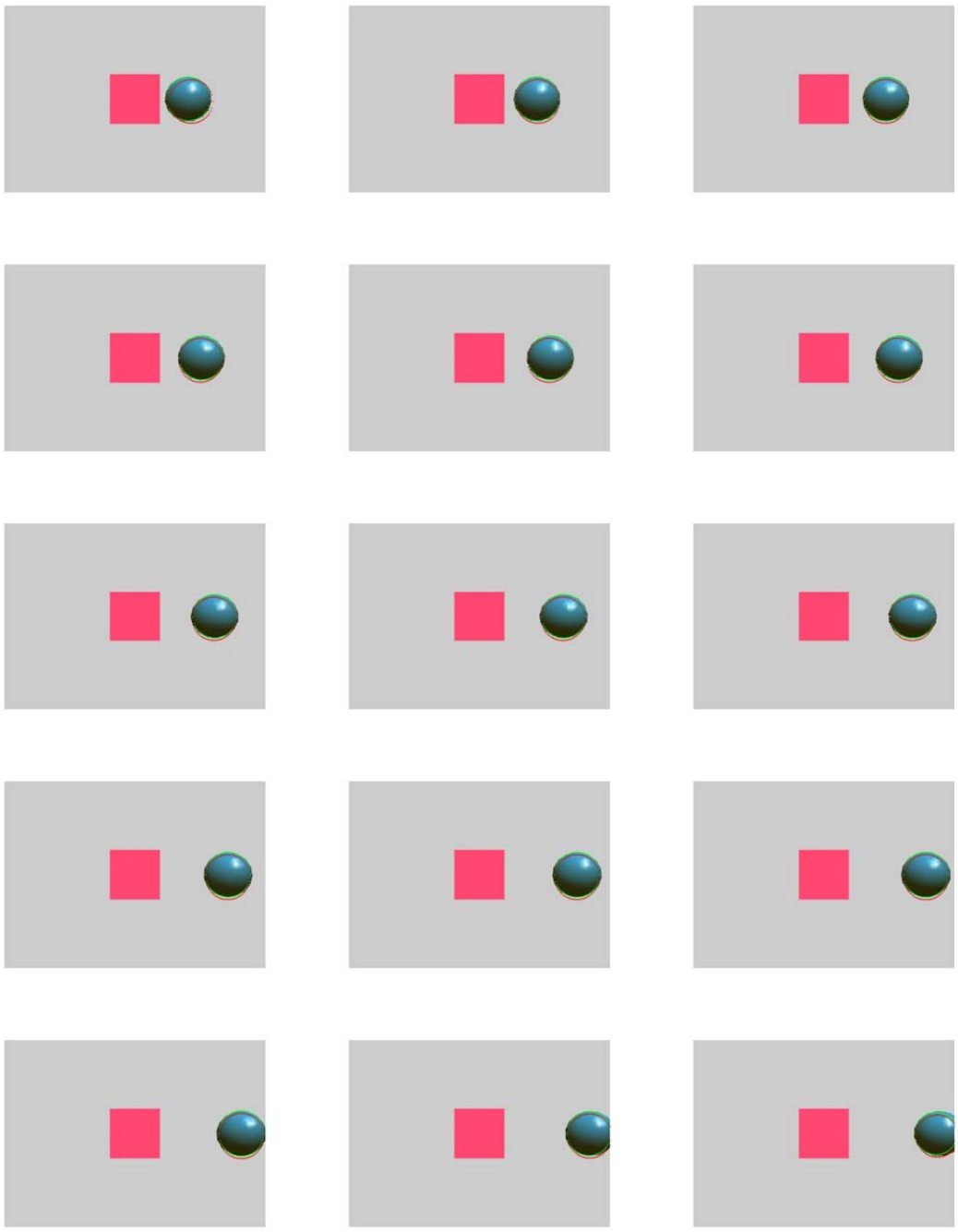


Figure A.9: Kalman filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A3.

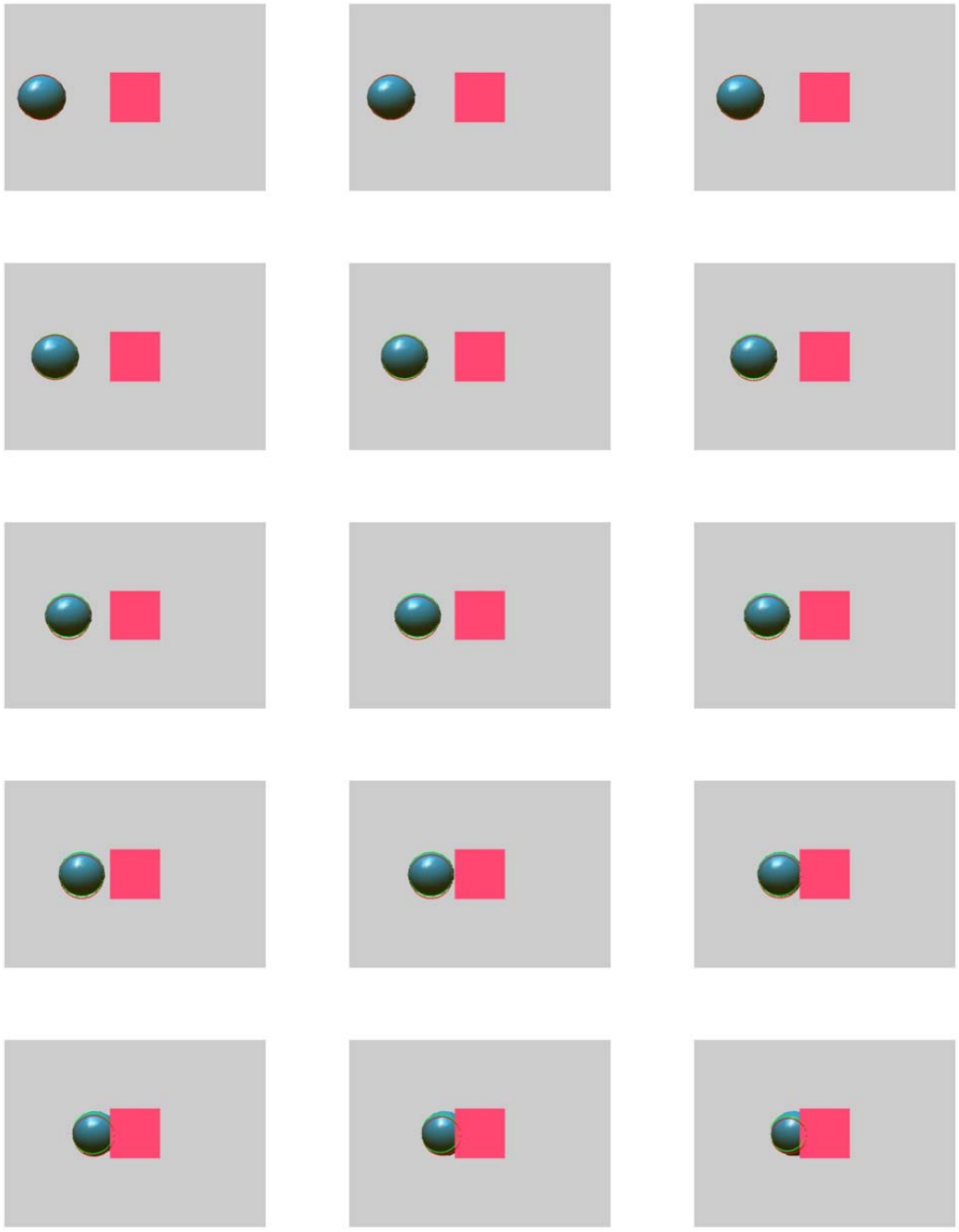


Figure A.10: Kalman filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A4.



Figure A.11: Kalman filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A4.

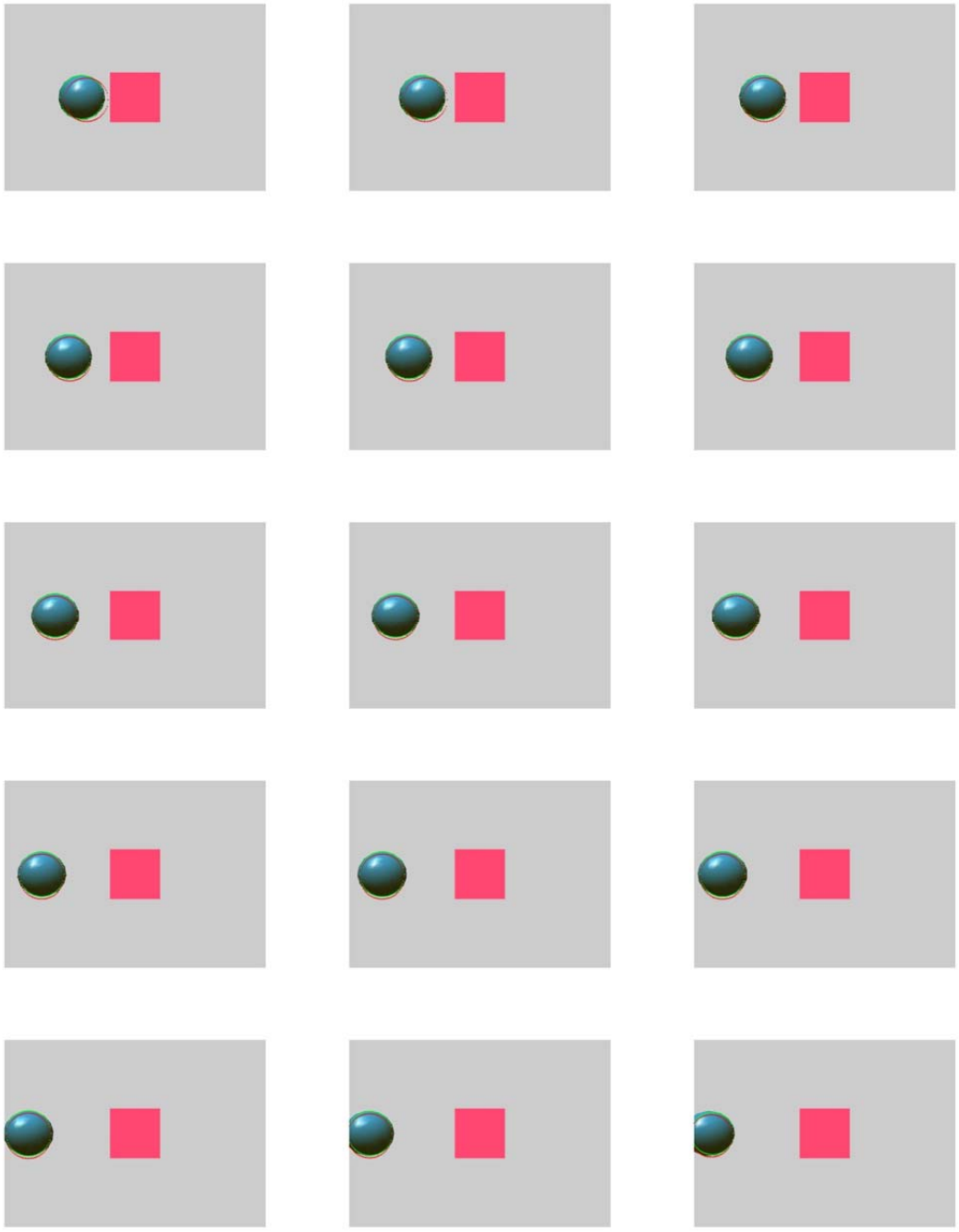


Figure A.12: Kalman filter object tracking result when tracking a moving ball in Frame 30 to Frame 45 (From top left to top right and then to bottom) in Video sequence A4.

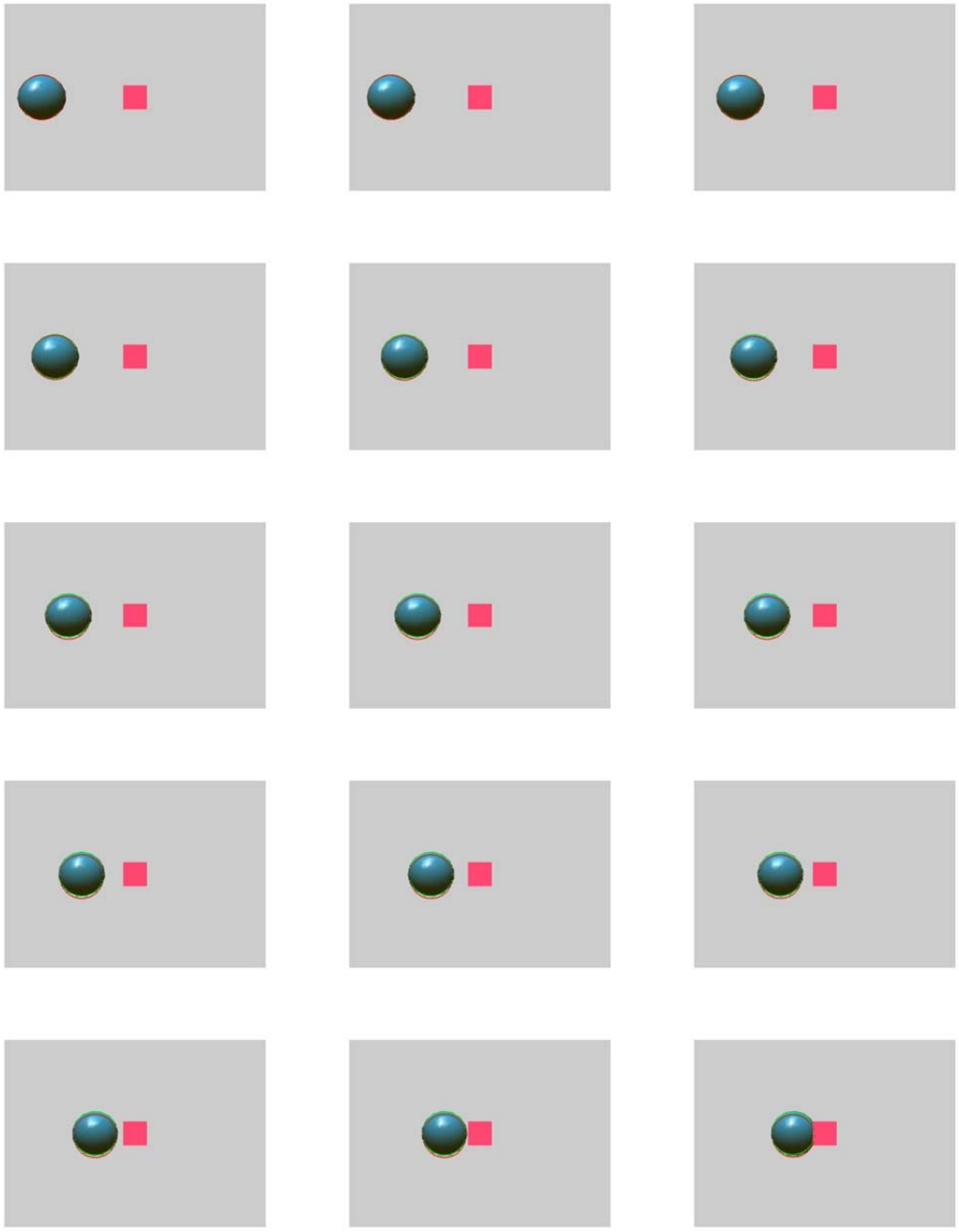


Figure A.13: Kalman filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A5.

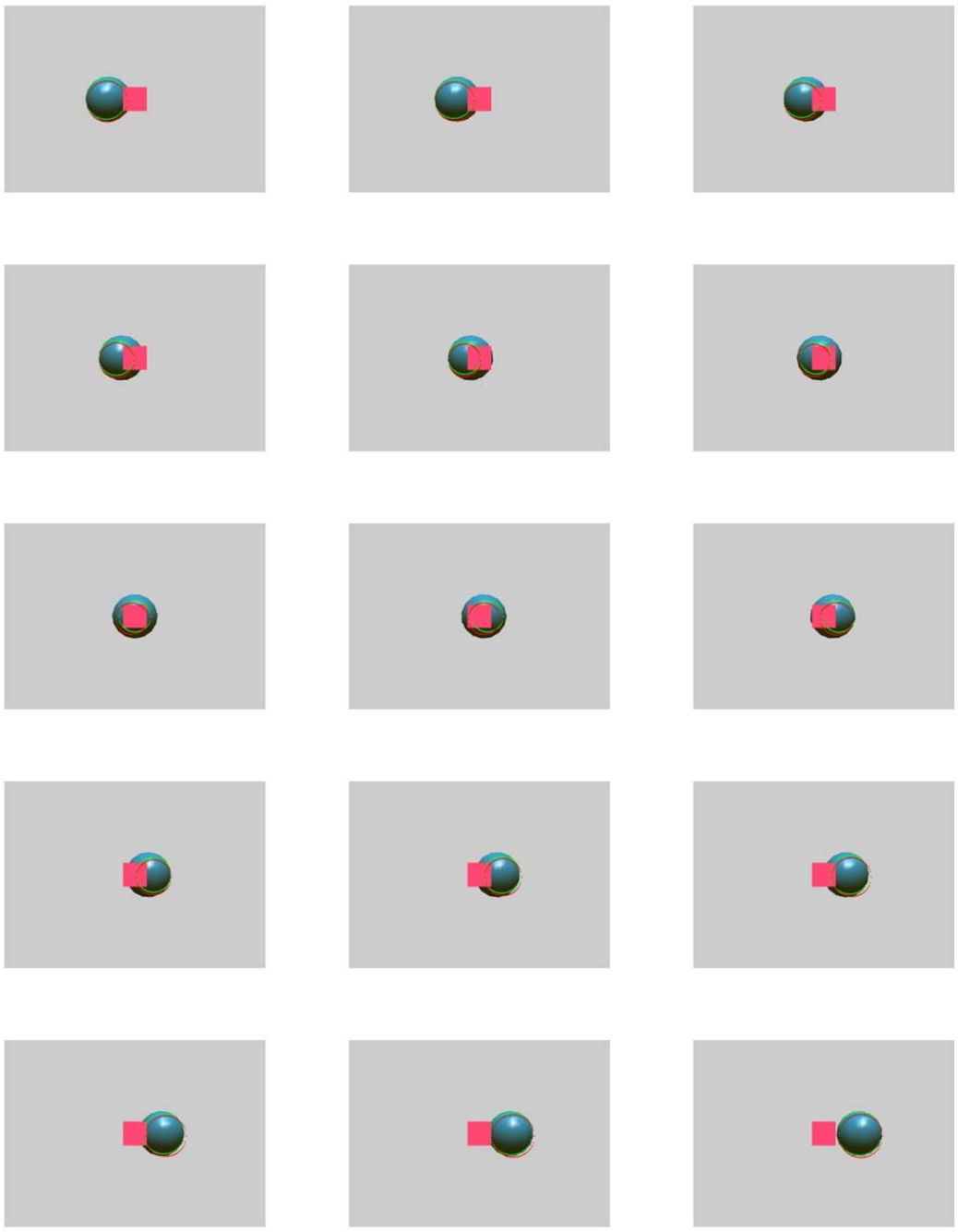


Figure A.14: Kalman filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A5.

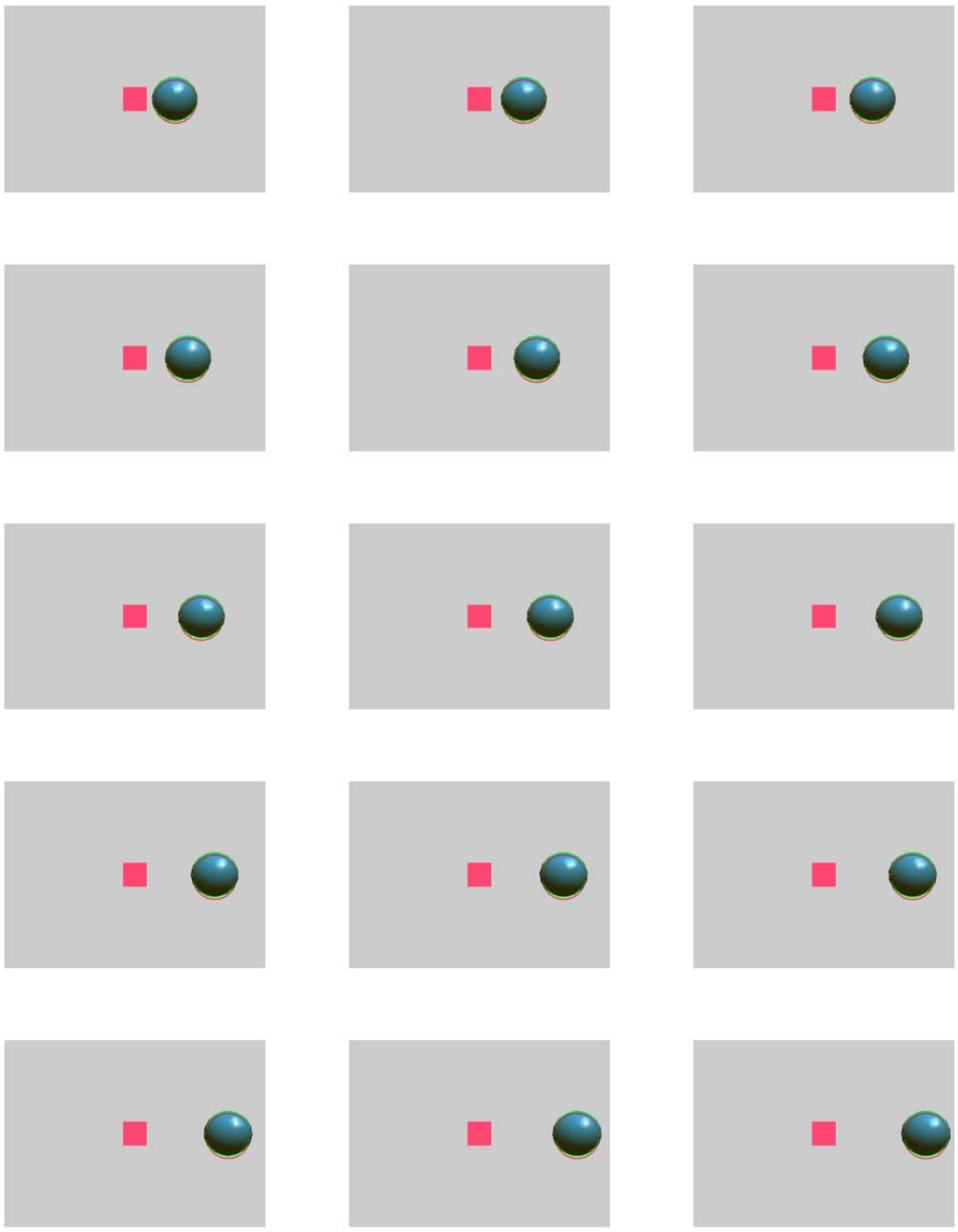


Figure A.15: Kalman filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A5.

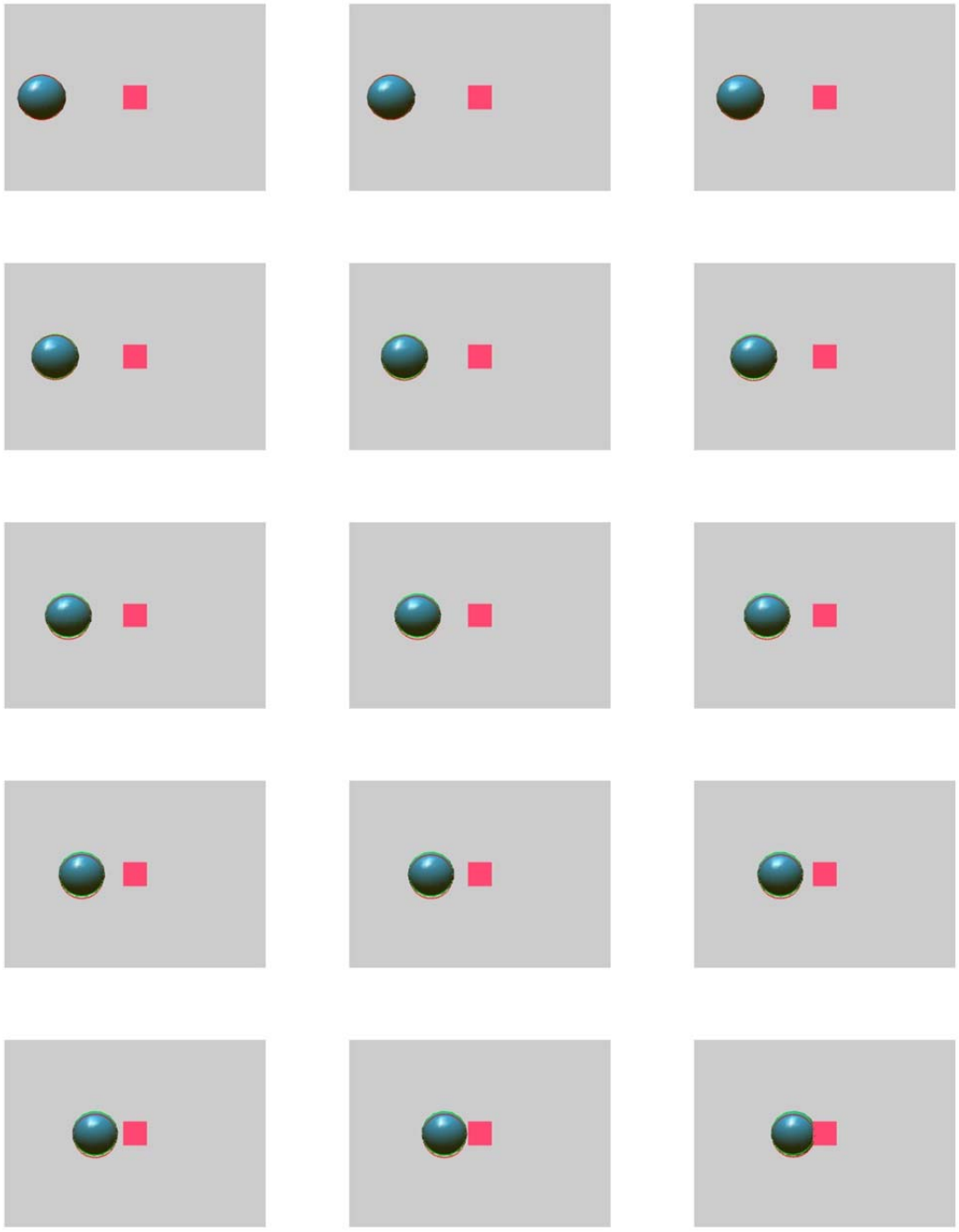


Figure A.16: Kalman filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A6.

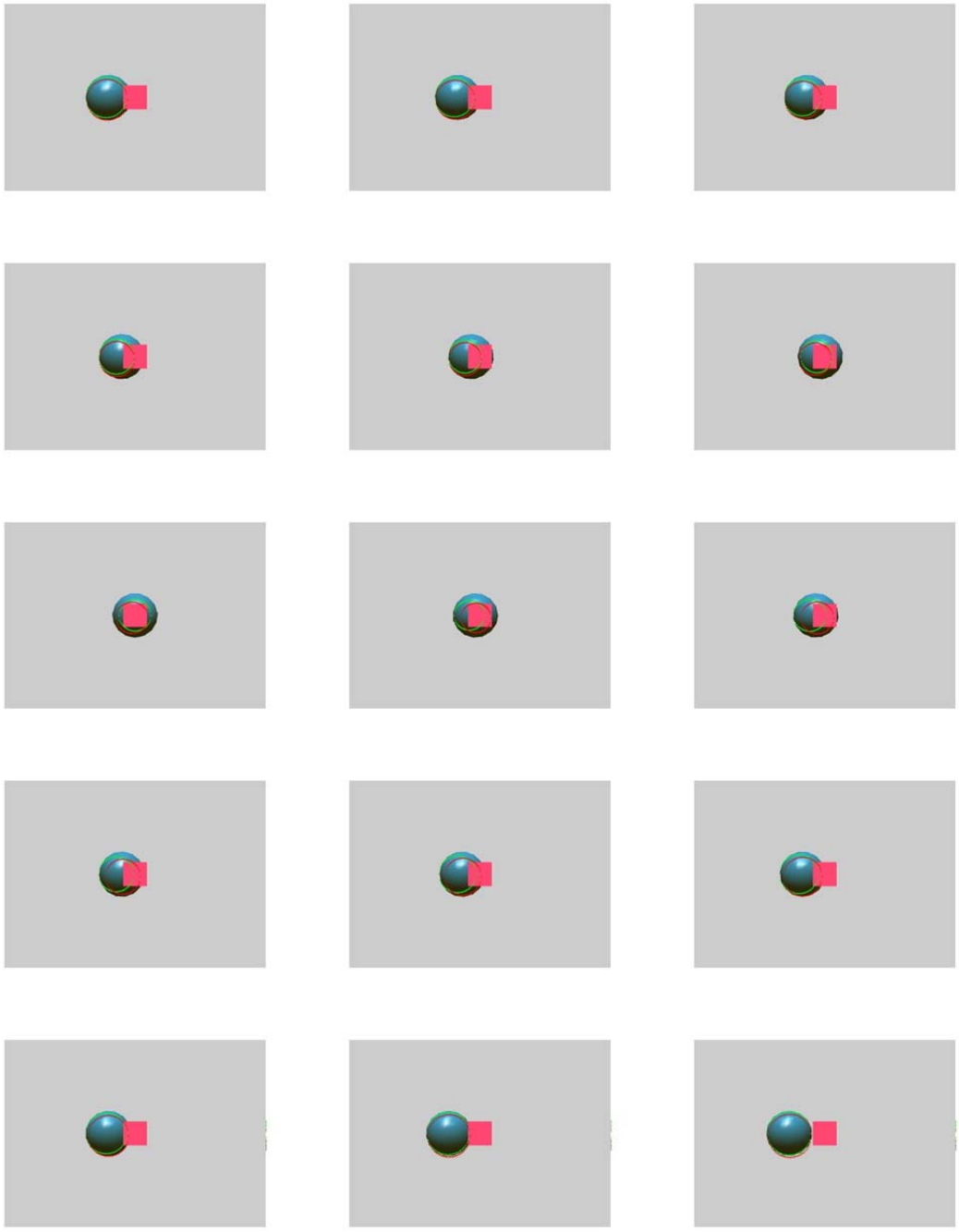


Figure A.17: Kalman filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A6.

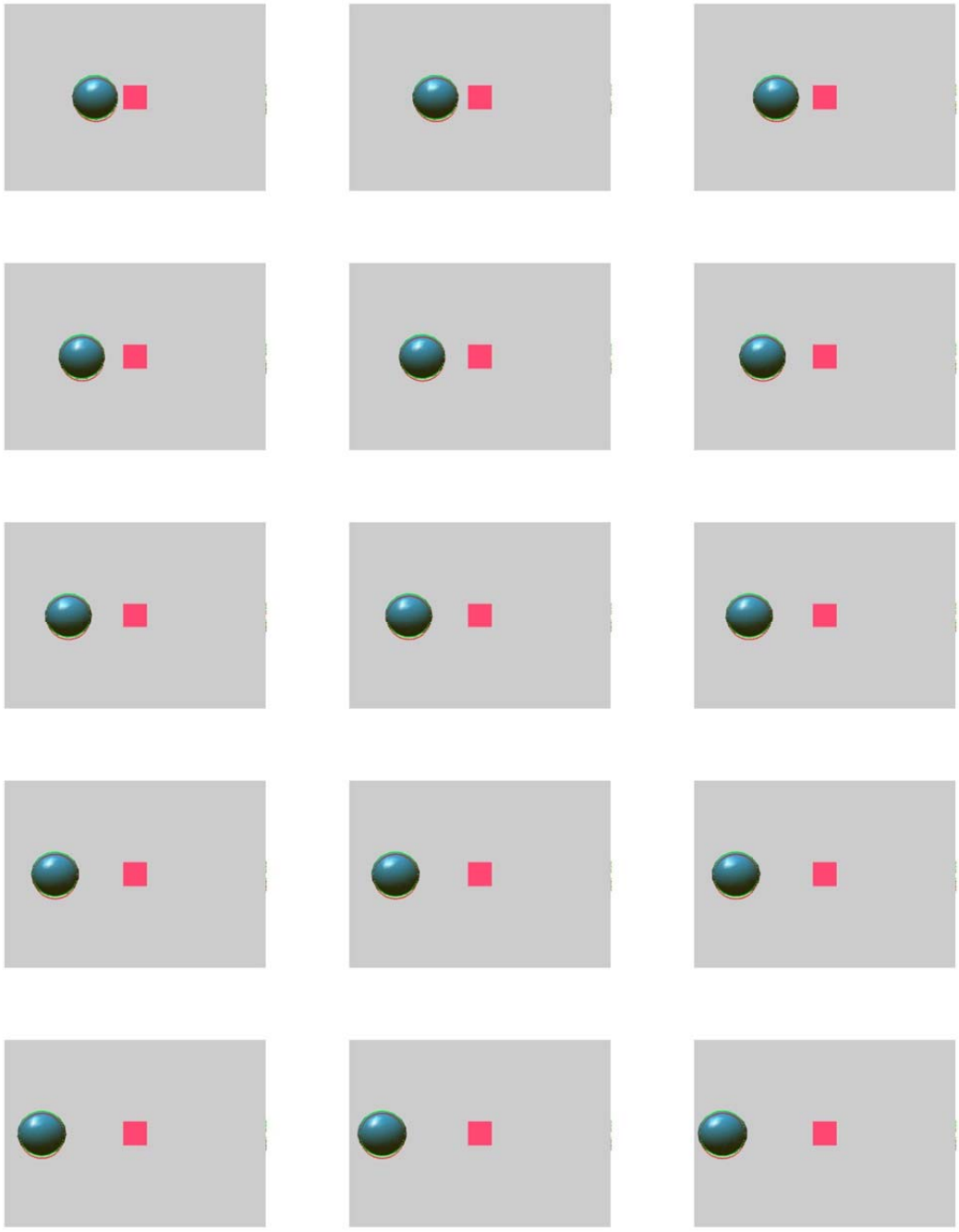


Figure A.18: Kalman filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A6.



Figure A.19: Kalman filter object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence PETS 2009



Figure A.20: Kalman filter object tracking result when tracking a pedestrian in Frame 15 to Frame 30 (From top left to top right and then to bottom) in Video sequence PETS 2009



Figure A.21: Kalman filter object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence PETS 2009



Figure A.22: Kalman filter object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence PETS 2009



Figure A.23: Kalman filter object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence PETS 2009



Figure A.24: Kalman filter object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence PETS 2009



Figure A.25: Kalman filter object tracking result when tracking a pedestrian in Frame 91 to Frame 105 (From top left to top right and then to bottom) in Video sequence PETS 2009



Figure A.26: Kalman filter object tracking result when tracking a pedestrian in Frame 106 to Frame 120 (From top left to top right and then to bottom) in Video sequence PETS 2009

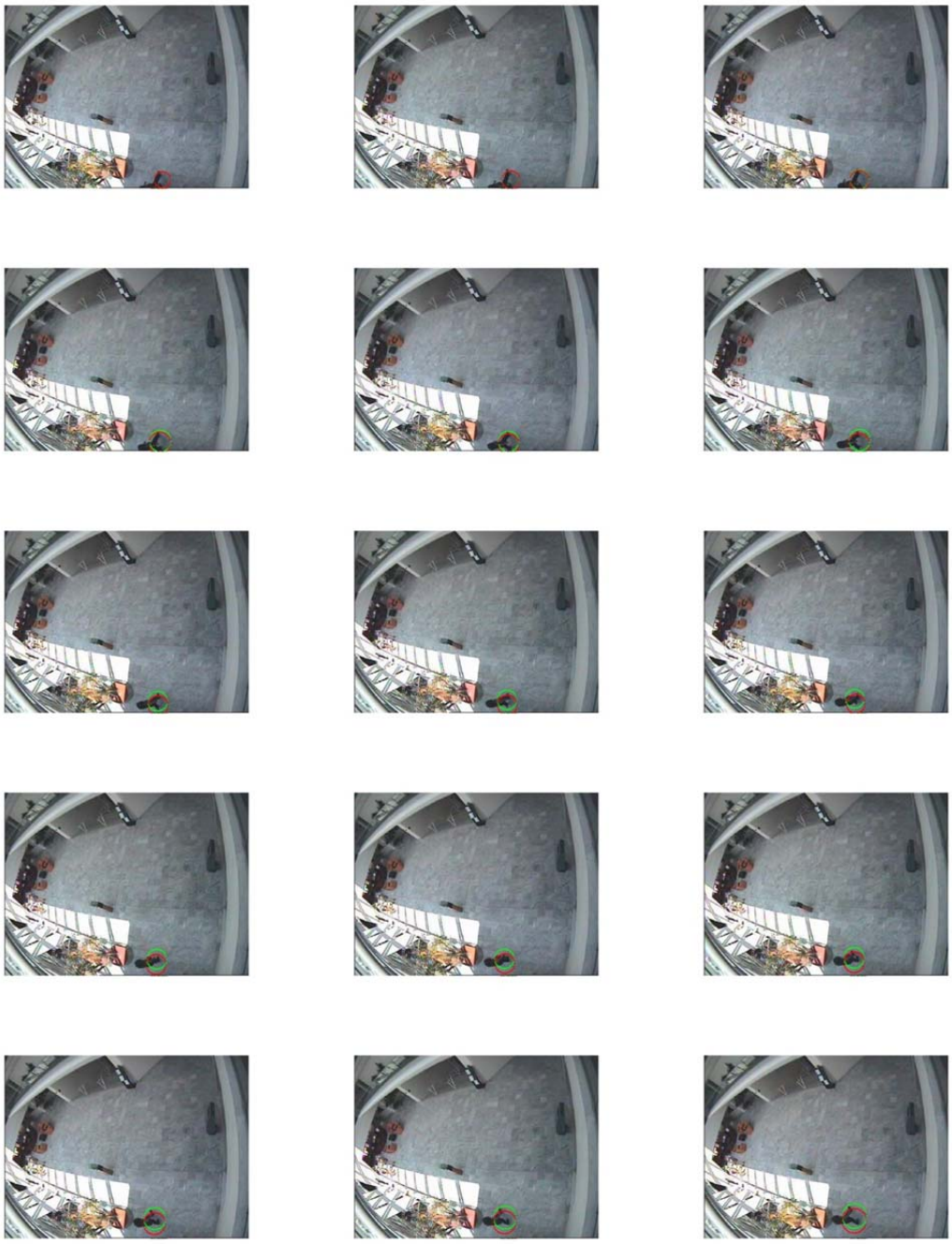


Figure A.27: Kalman filter object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence CAVIAR

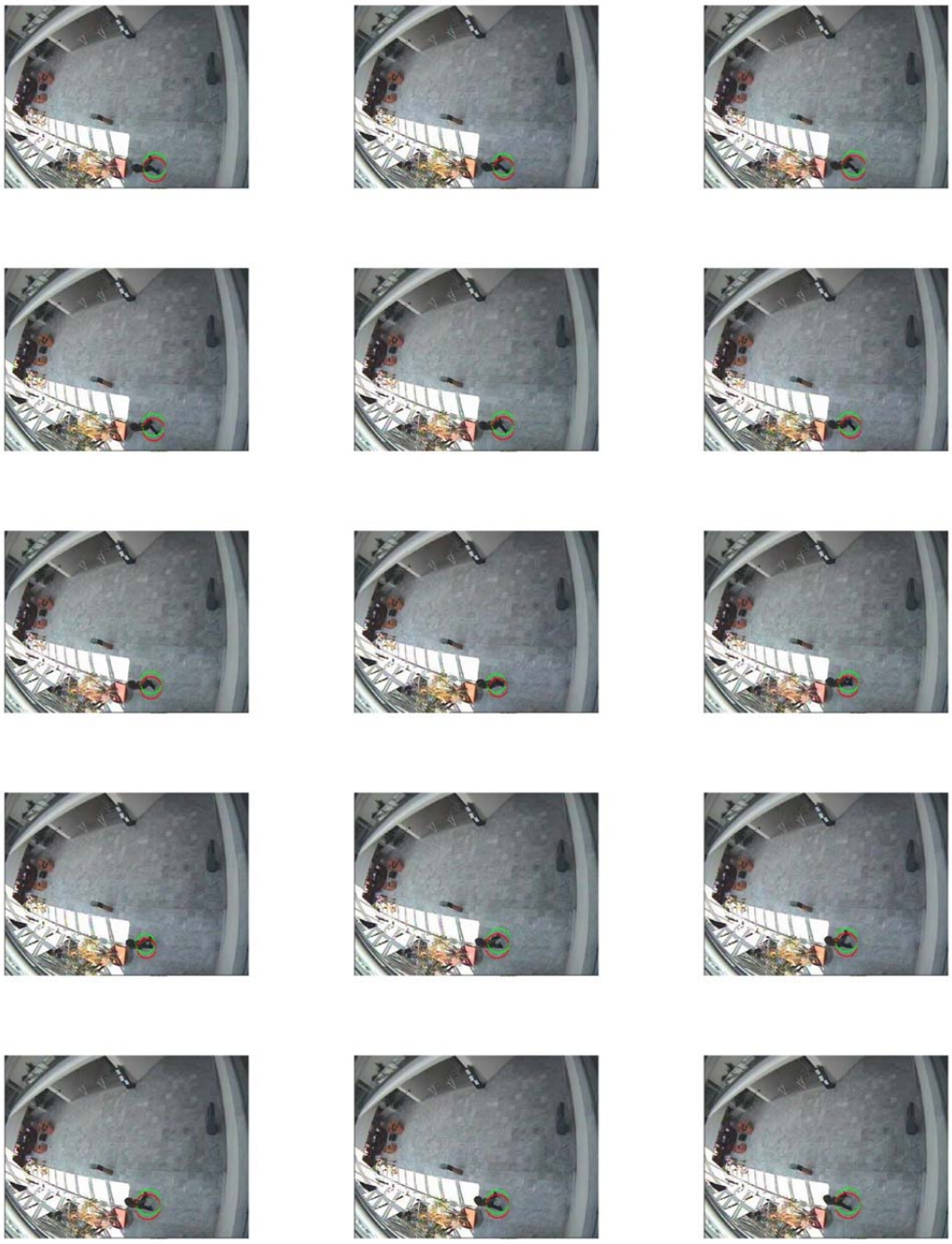


Figure A.28: Kalman filter object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence CAVIAR

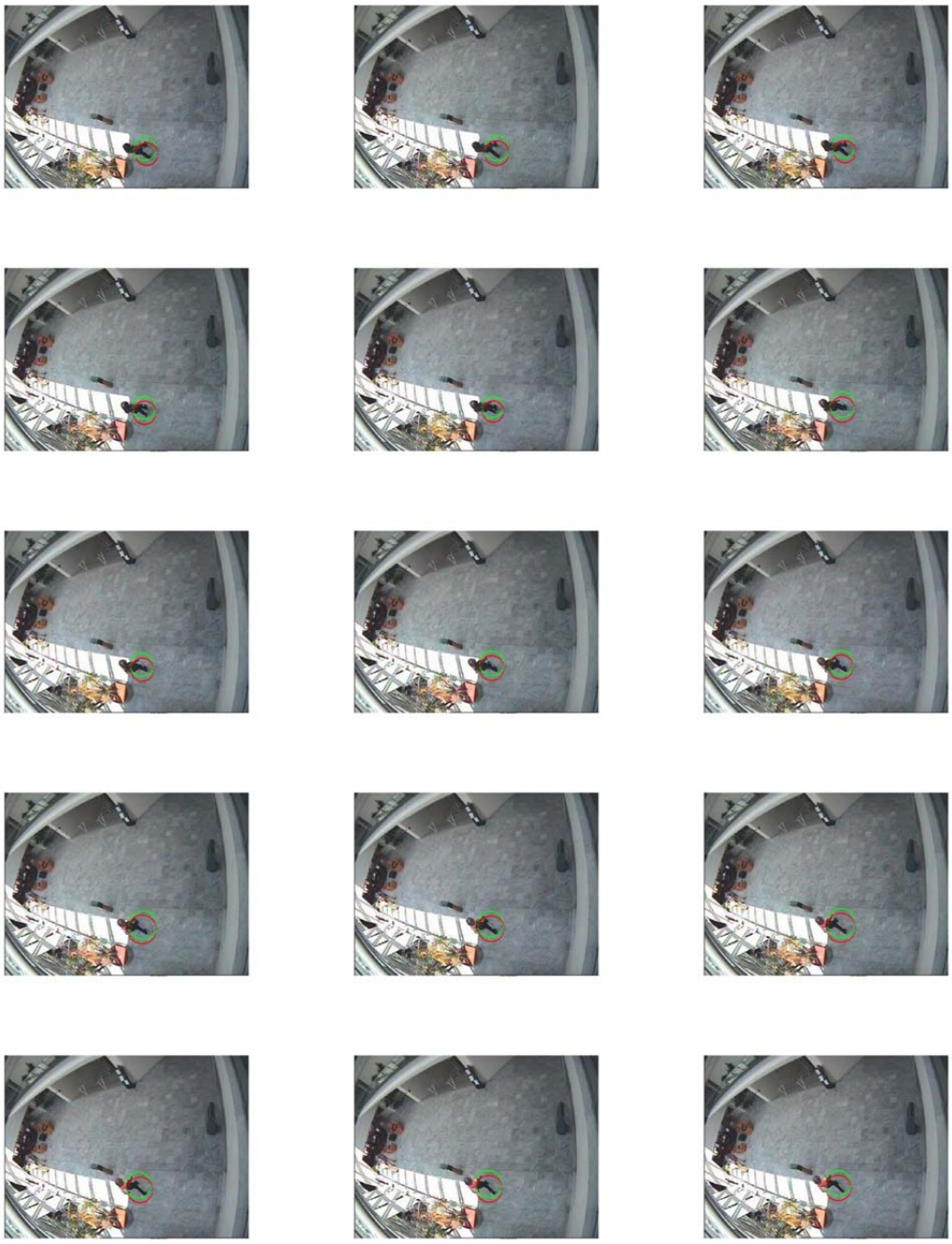


Figure A.29: Kalman filter object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence CAVIAR

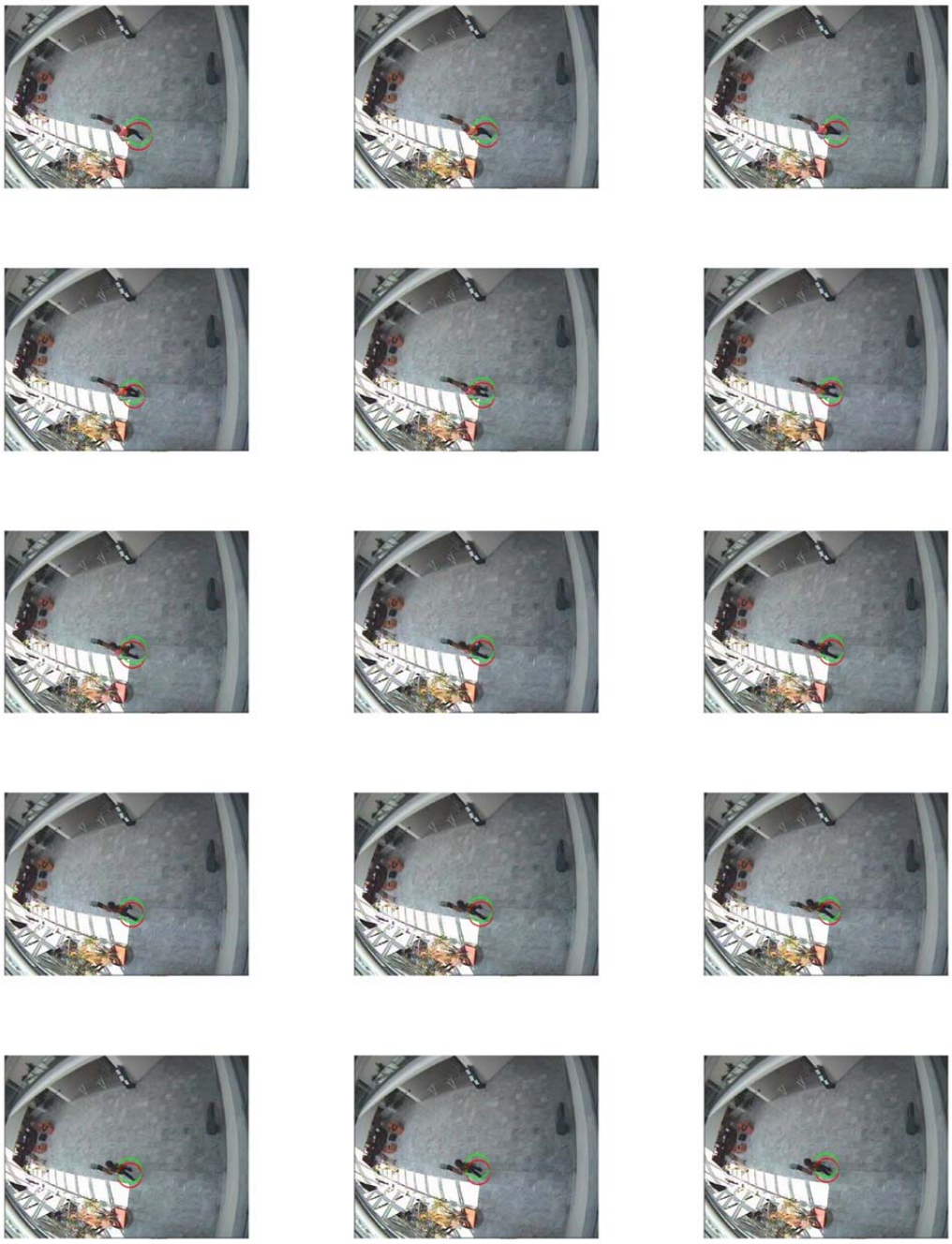


Figure A.30: Kalman filter object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence CAVIAR

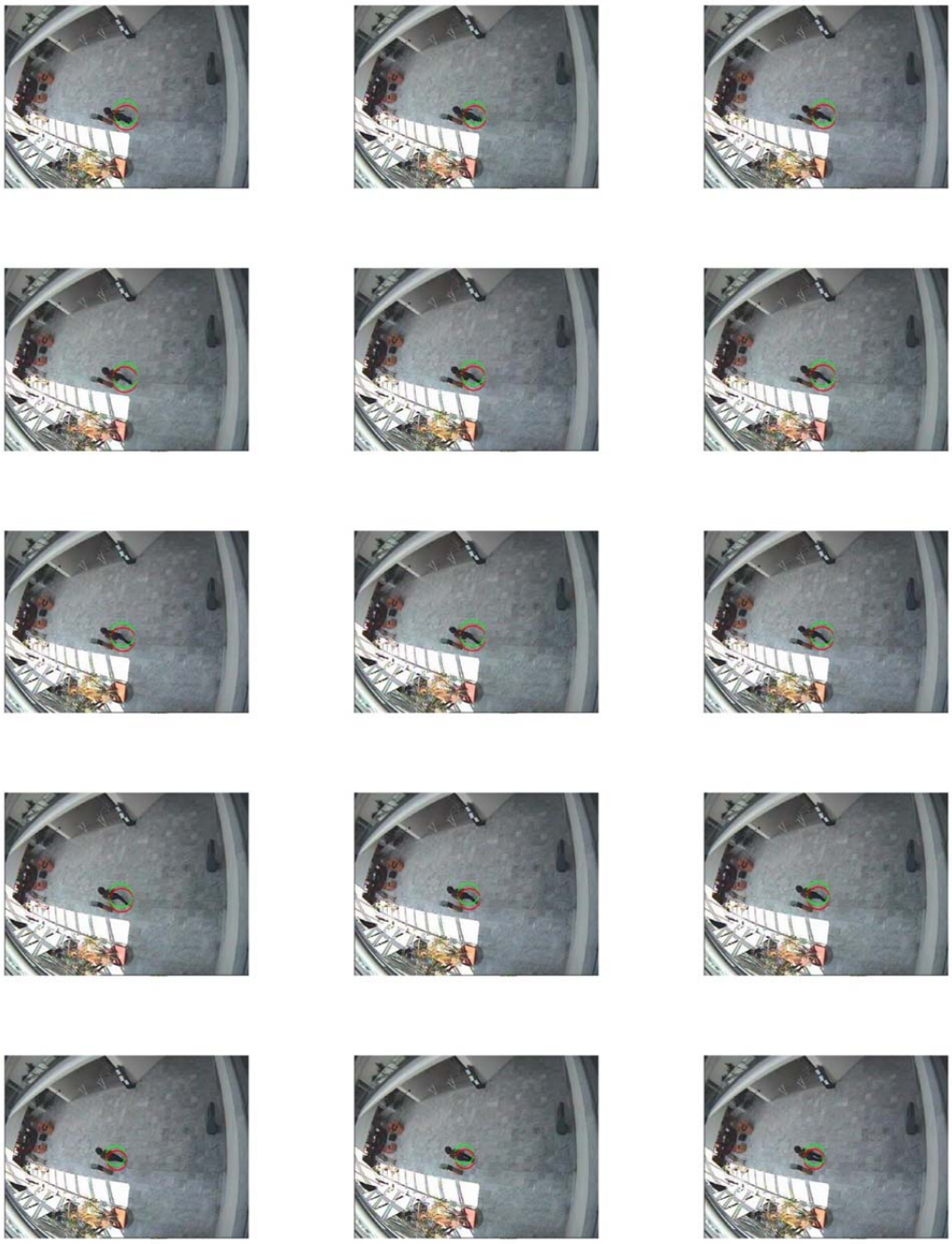


Figure A.31: Kalman filter object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence CAVIAR

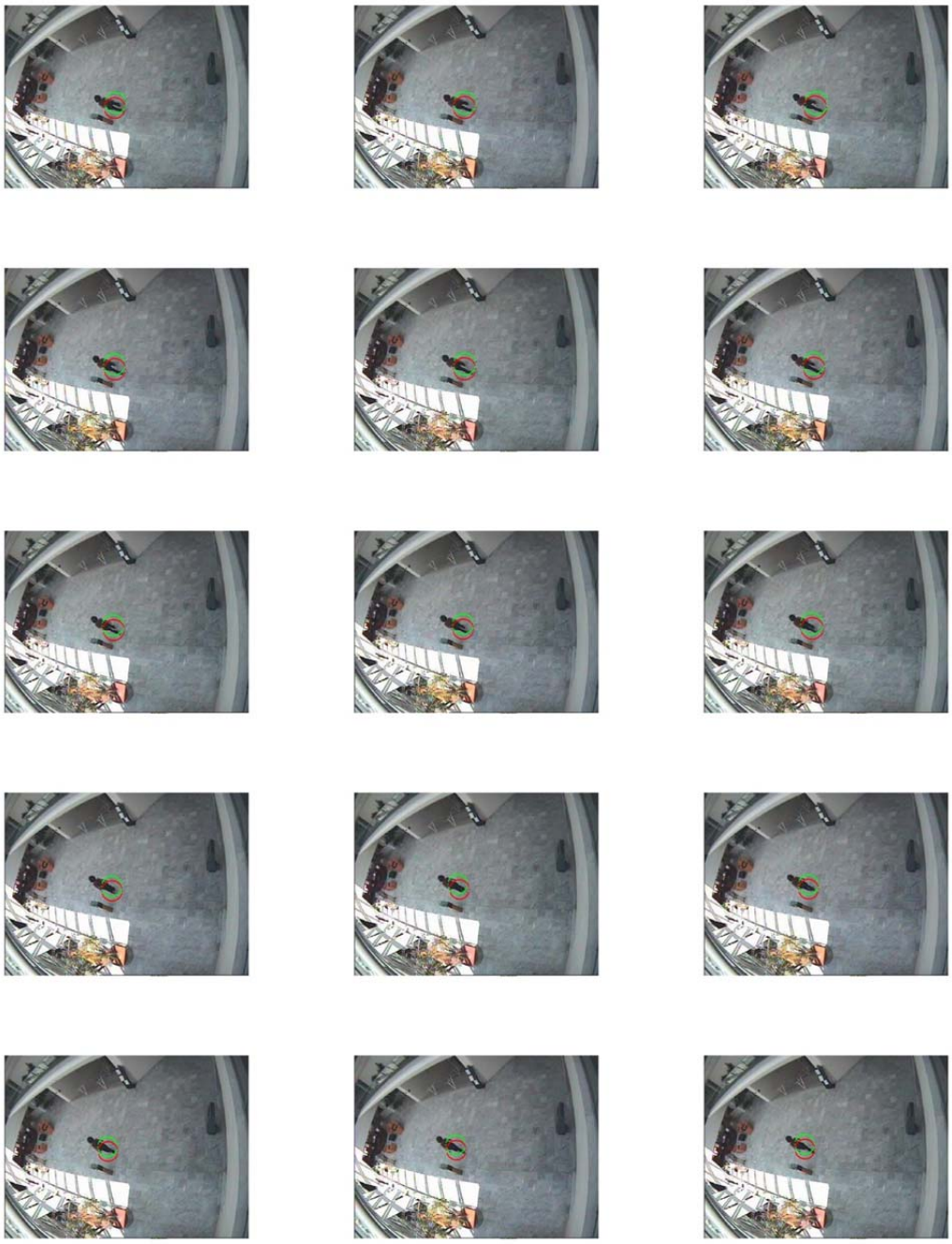


Figure A.32: Kalman filter object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence CAVIAR

Appendix B:
Particle Filter Object Tracking Result Frame by Frame

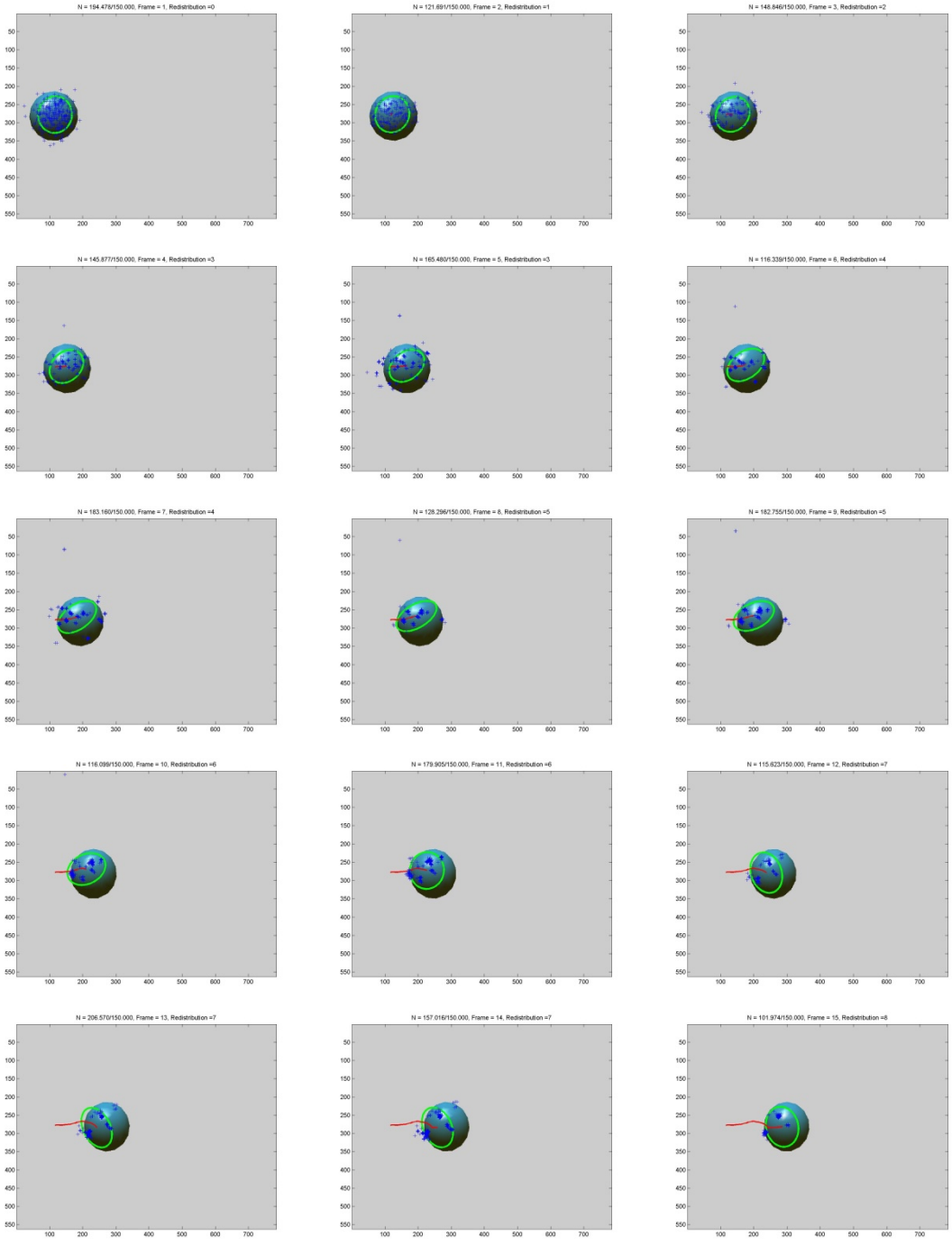


Figure B.1: Particle filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A1.

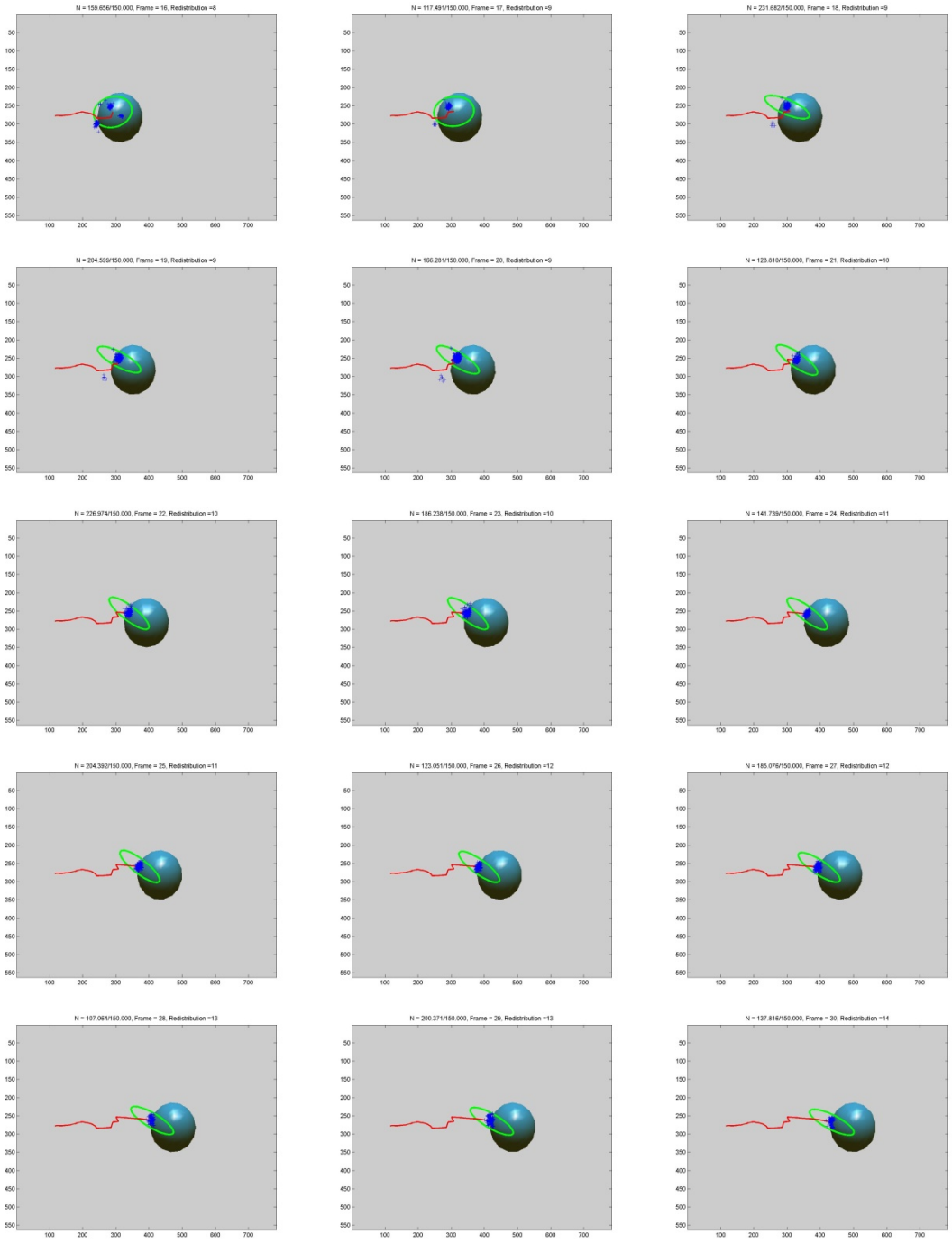


Figure B.2: Particle filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A1.

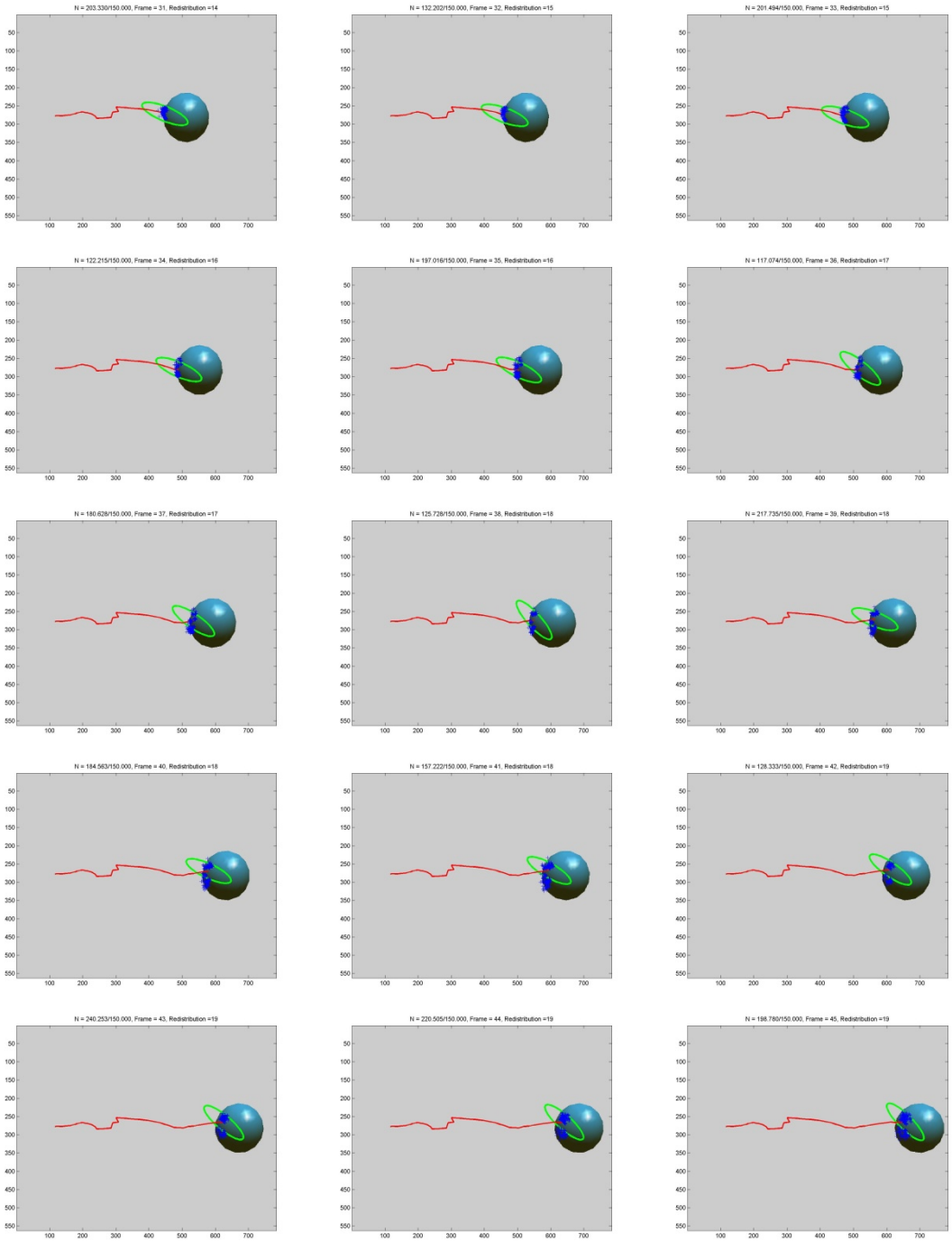


Figure B.3: Particle filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A1.

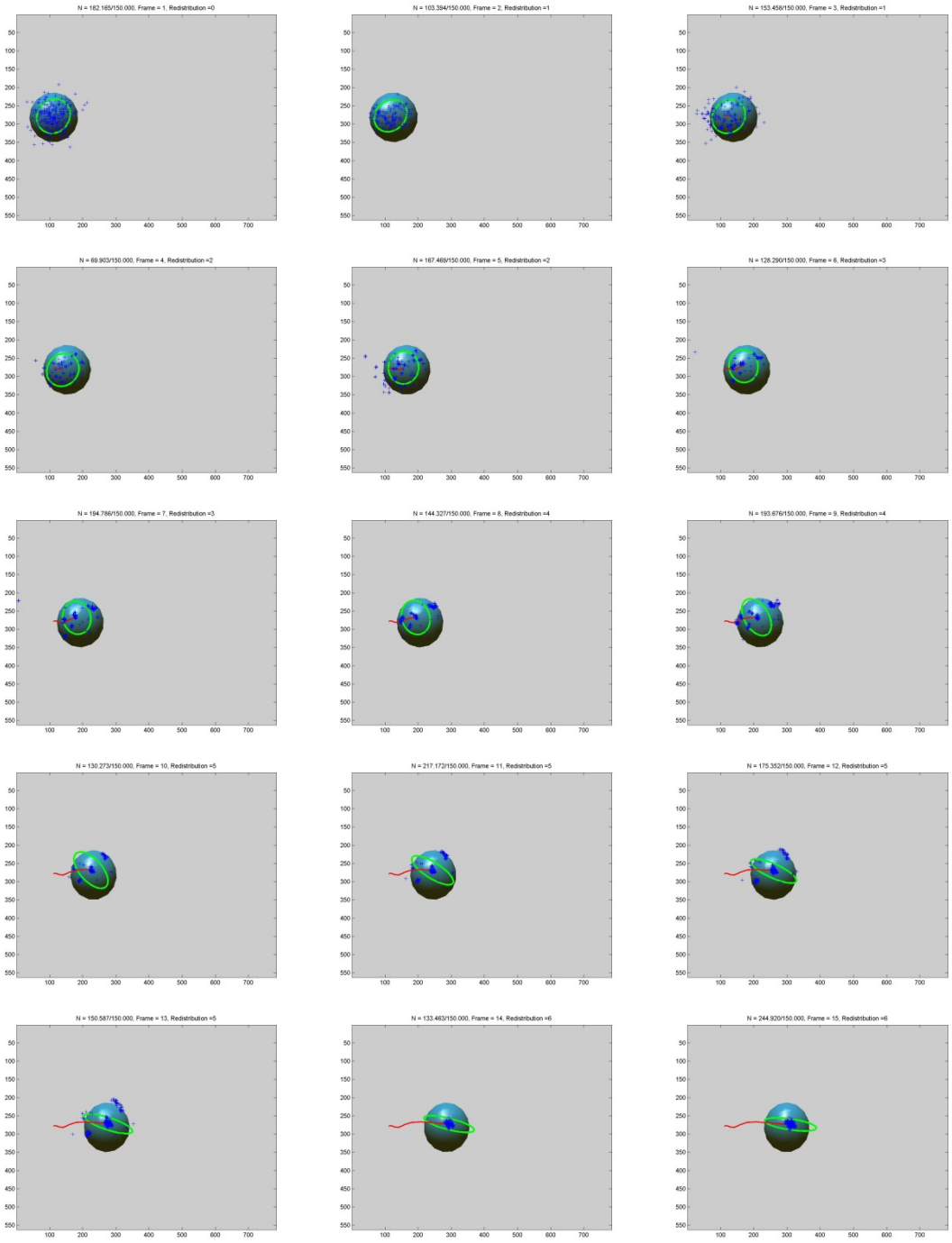


Figure B.4: Particle filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A2.

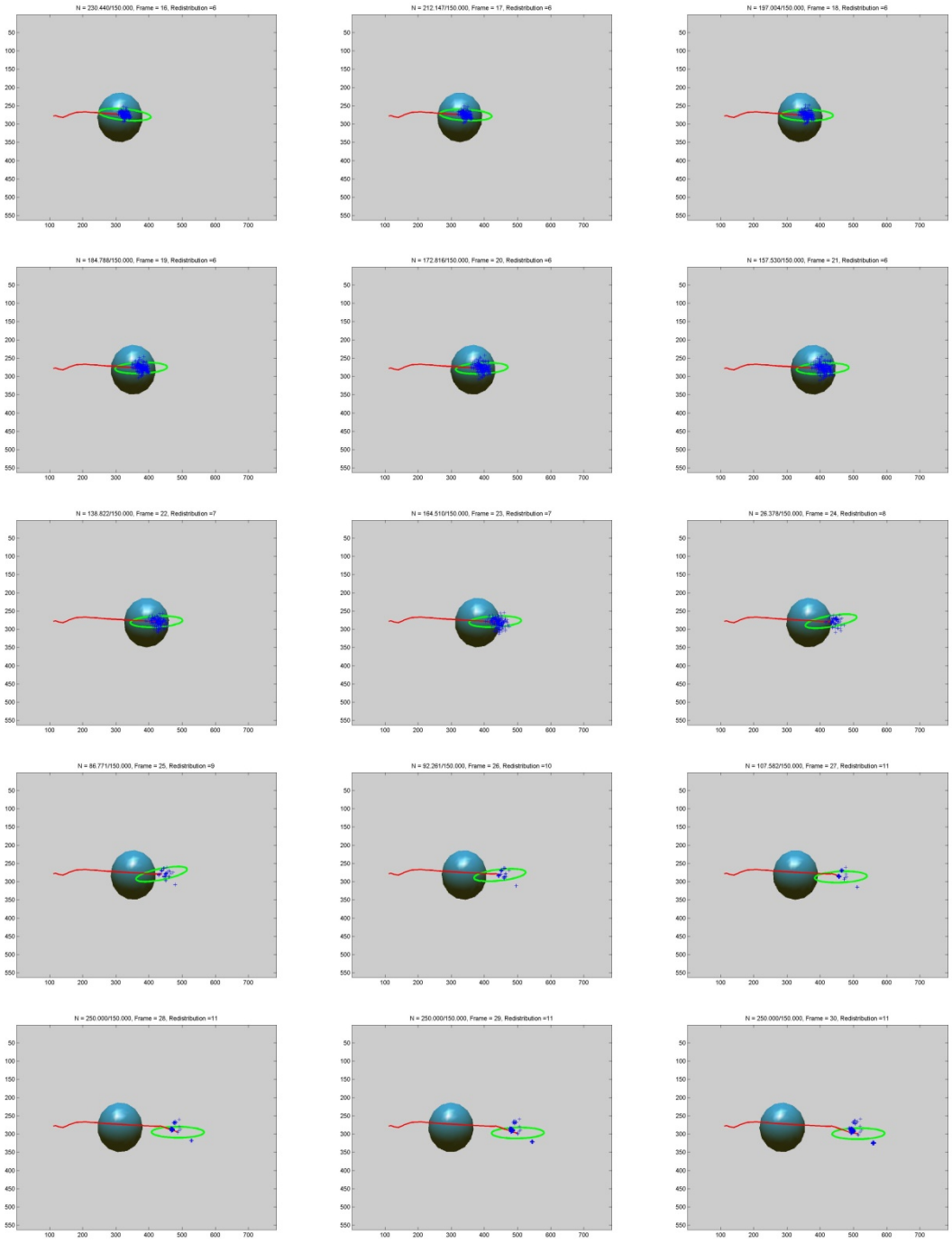


Figure B.5: Particle filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A2.

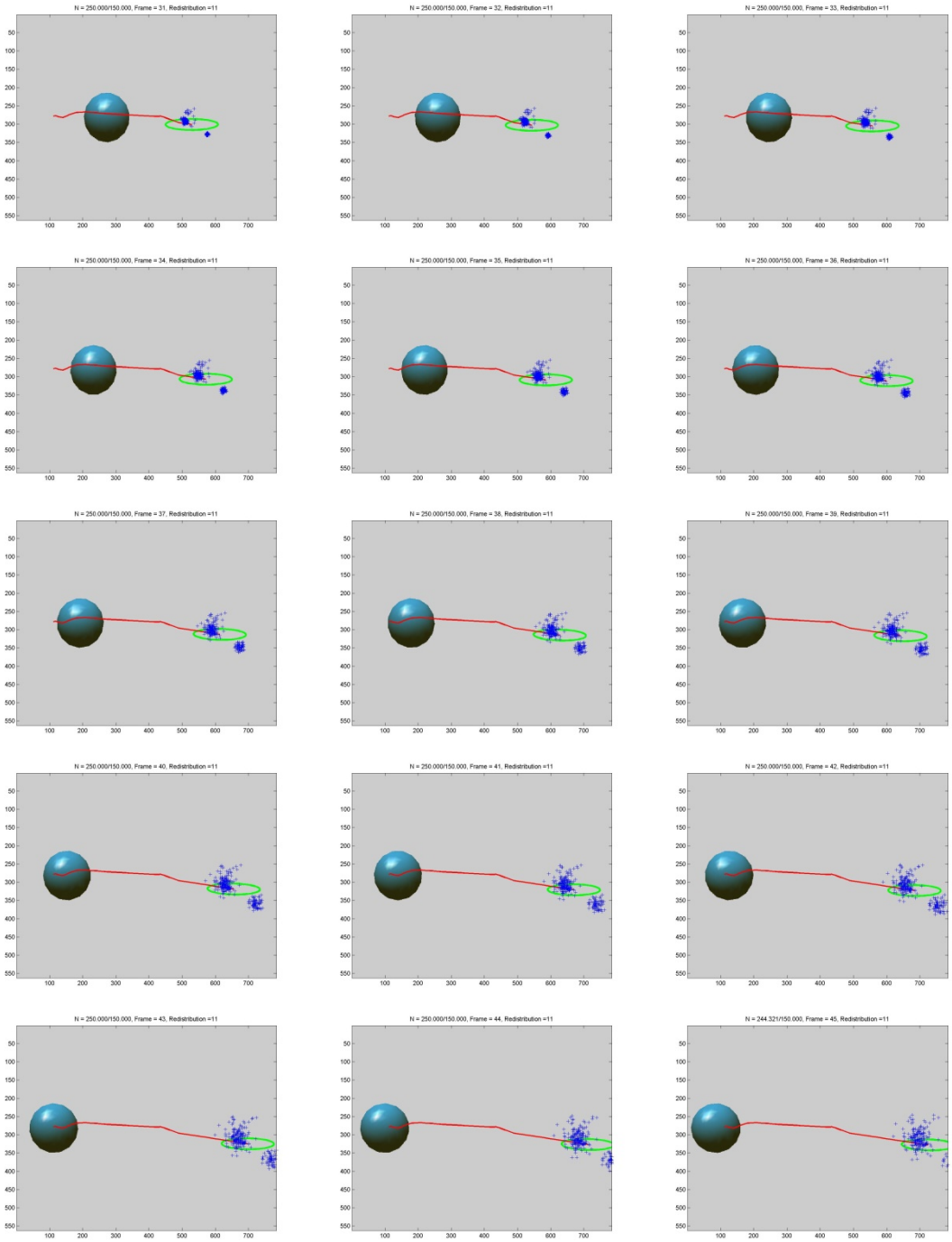


Figure B.6: Particle filter object tracking result when tracking a moving ball in Frame 31 to Frame 15 (From top left to top right and then to bottom) in Video sequence A2.

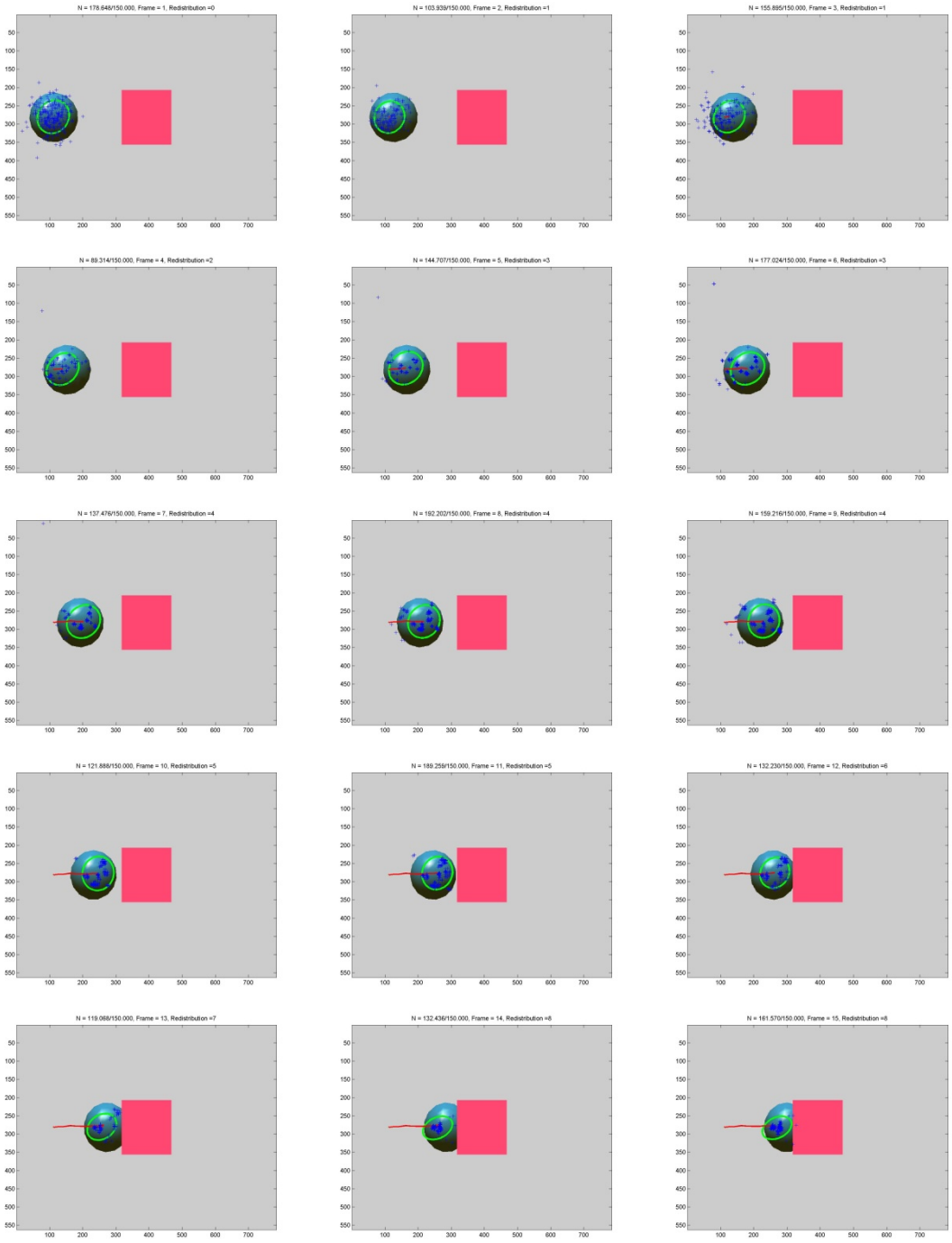


Figure B.7: Particle filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A3.

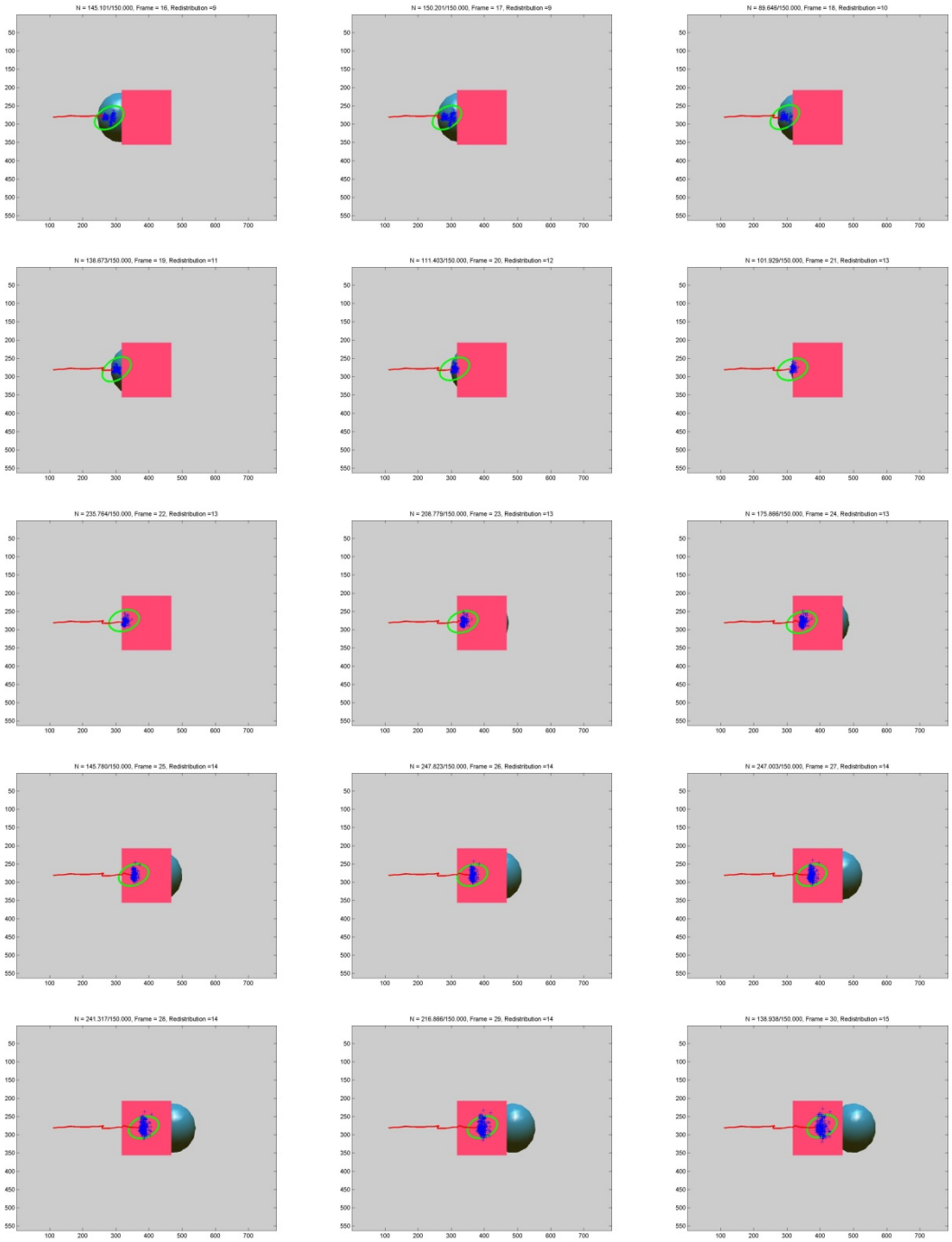


Figure B.8: Particle filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A3.

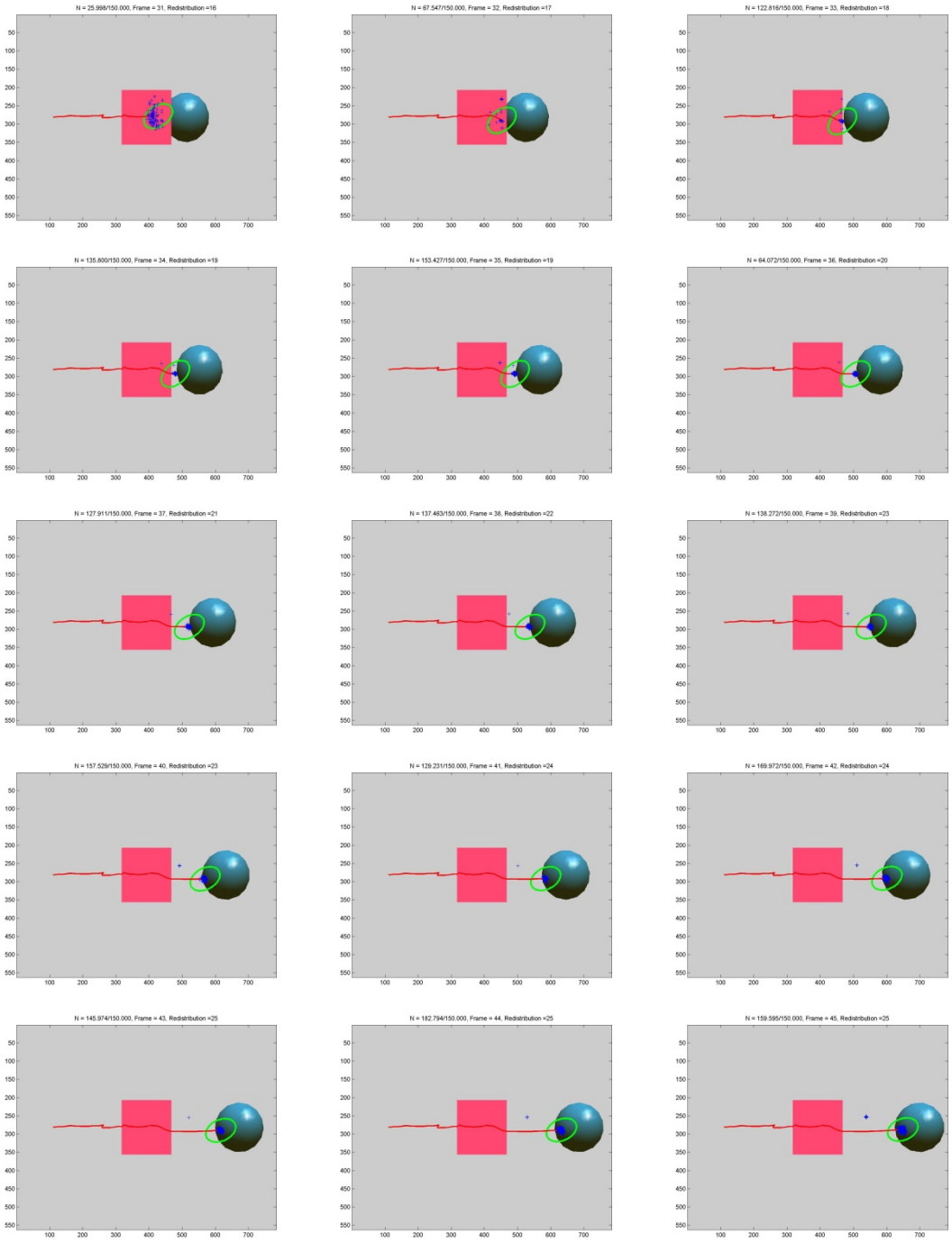


Figure B.9: Particle filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A3.

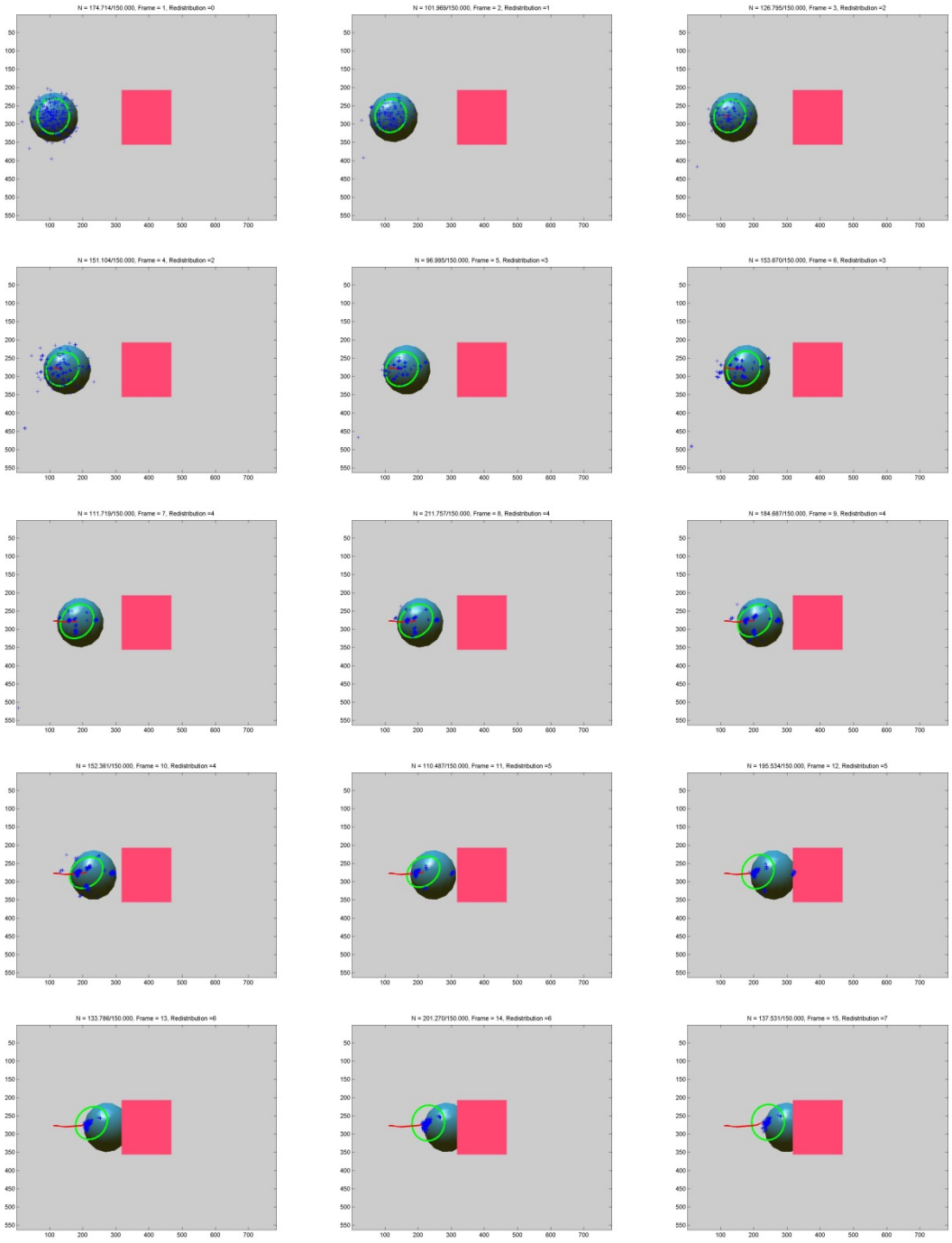


Figure B.10: Particle filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A4.

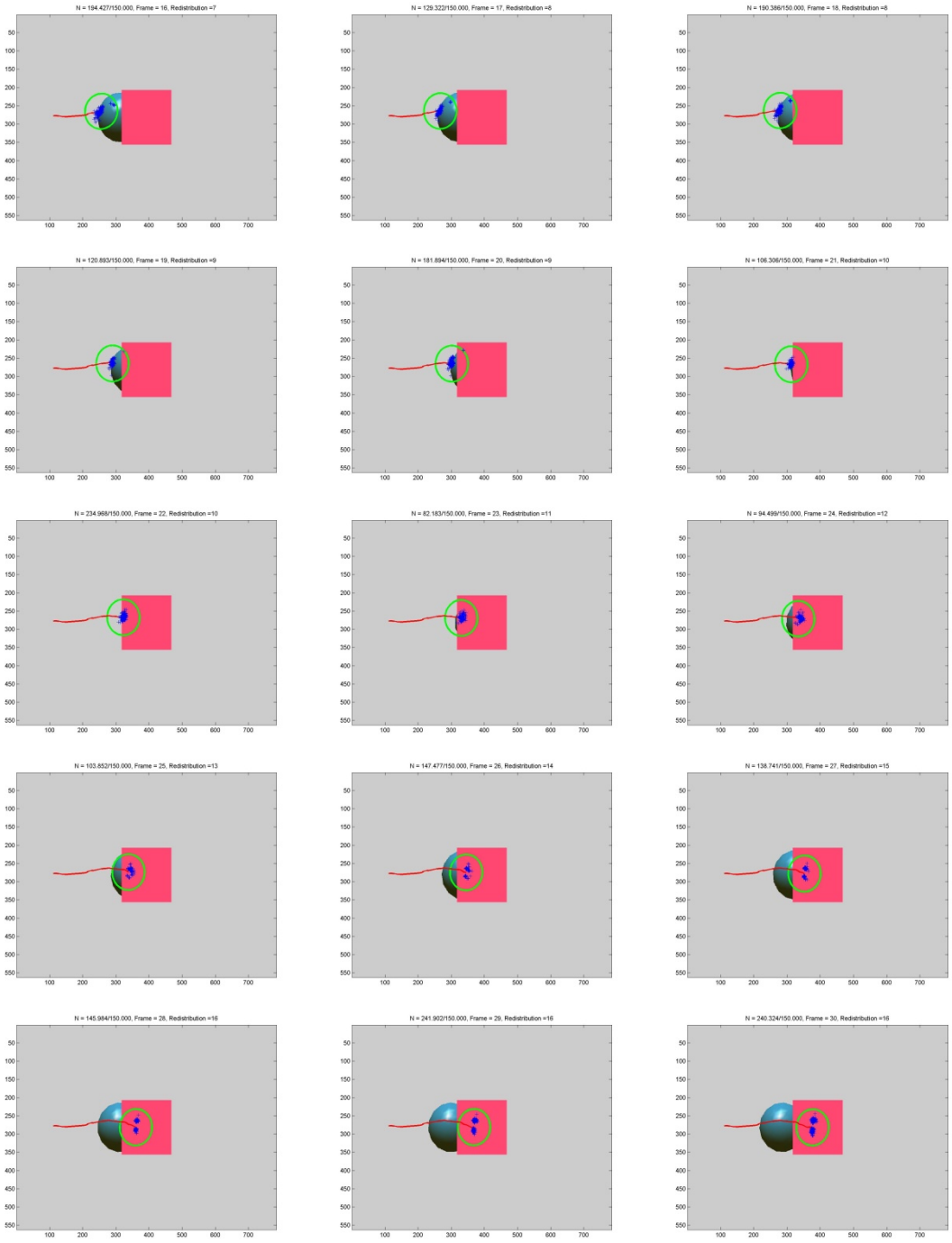


Figure B.11: Particle filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A4.

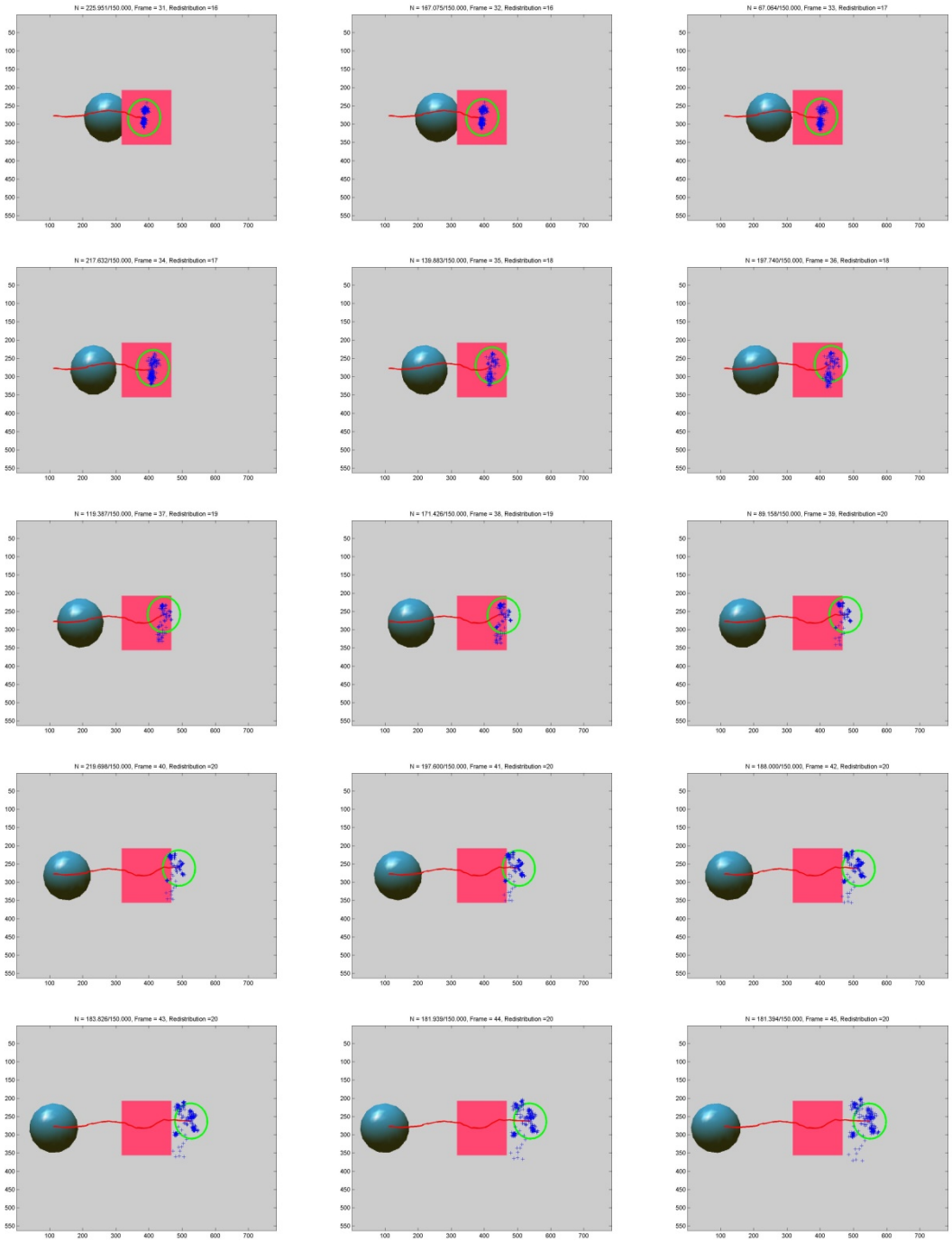


Figure B.12: Particle filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A4.

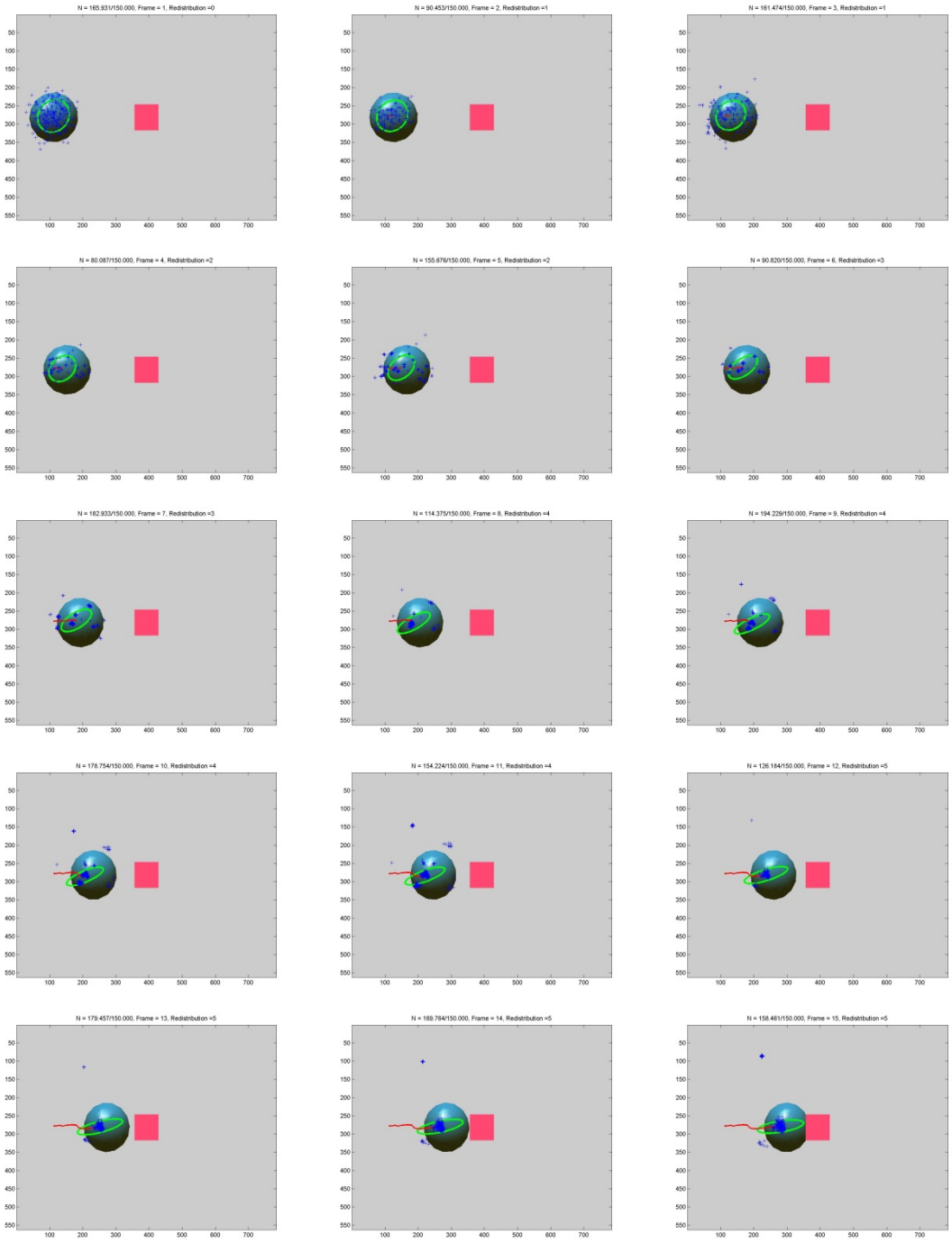


Figure B.13: Particle filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A5.

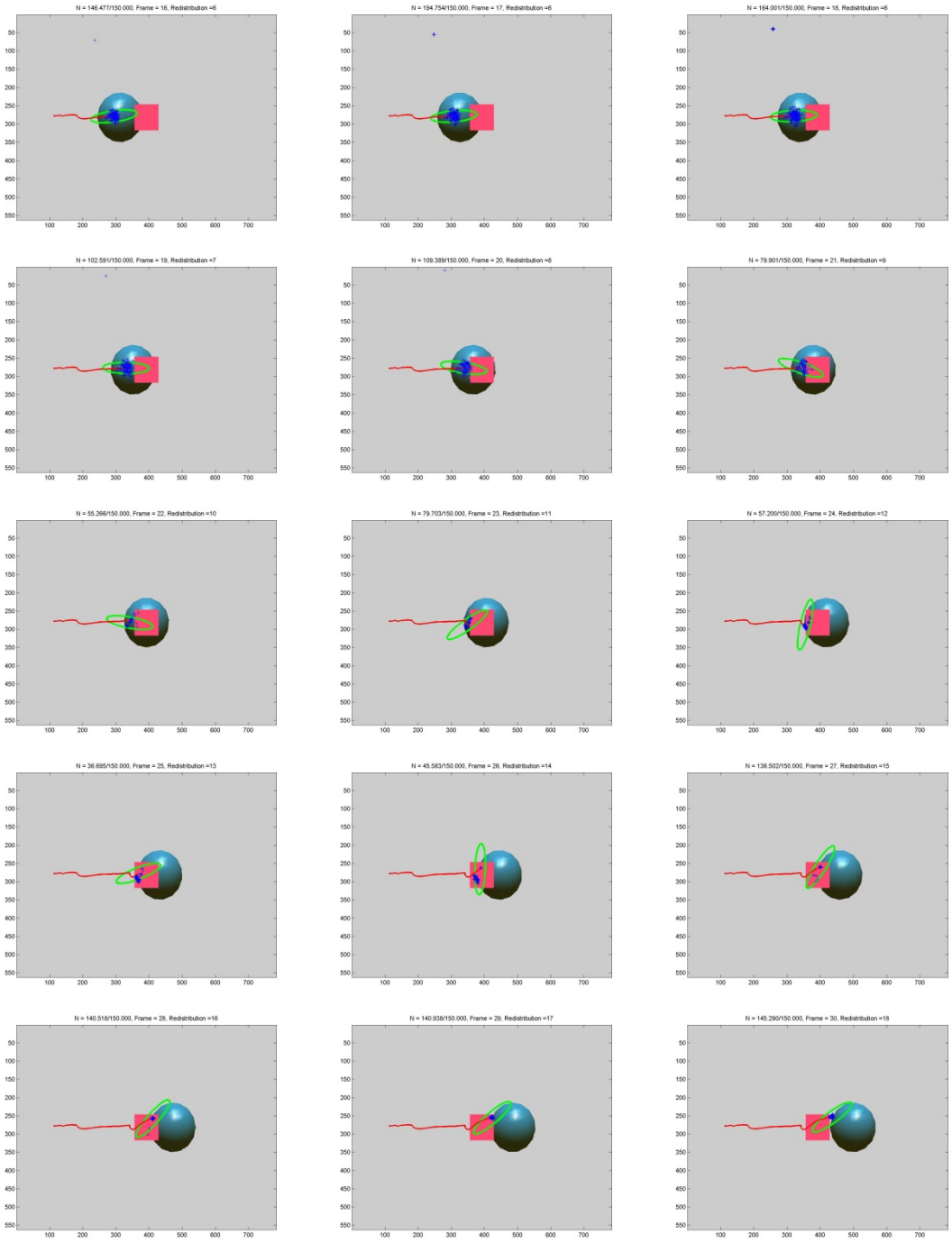


Figure B.14: Particle filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A5.

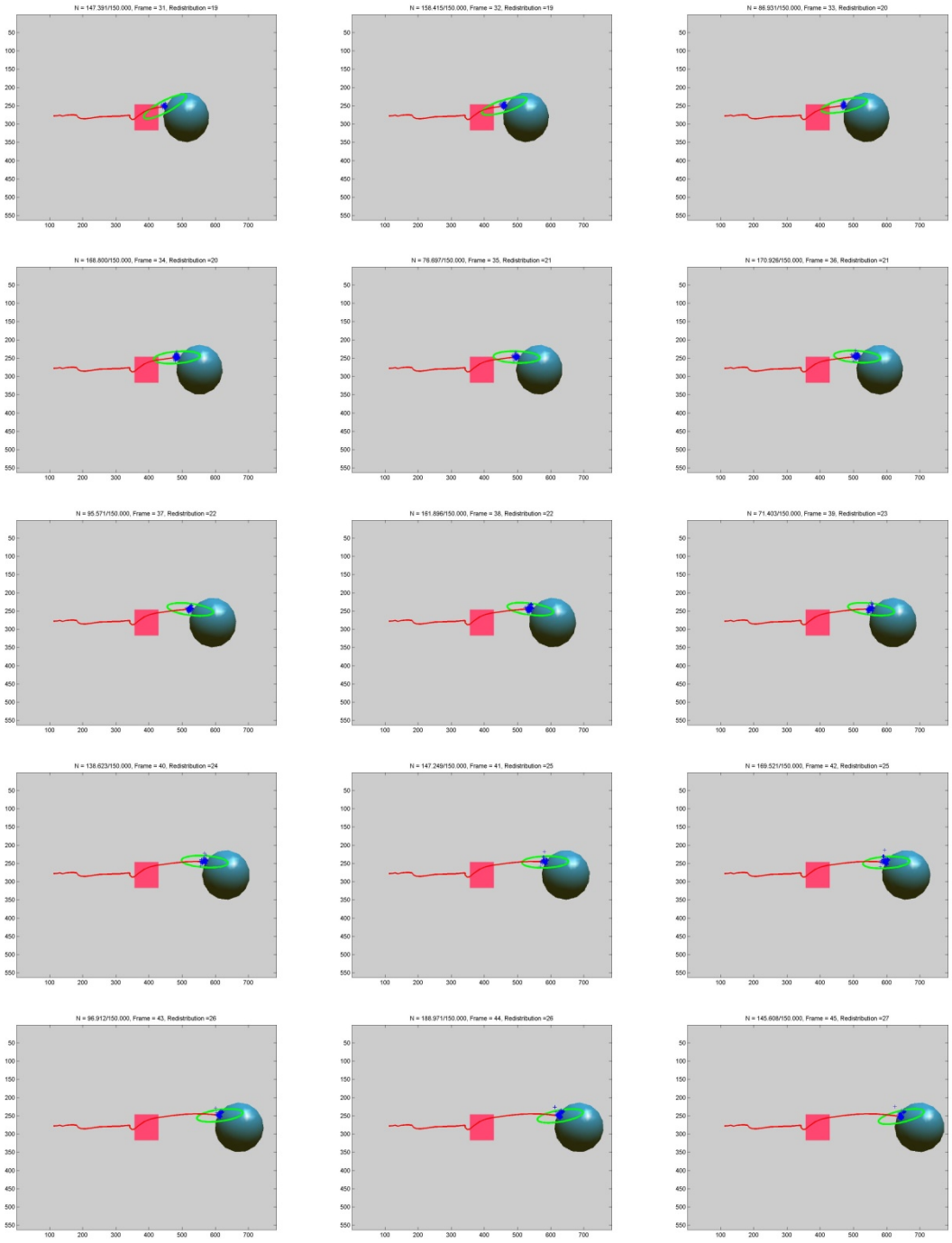


Figure B.15: Particle filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A5.

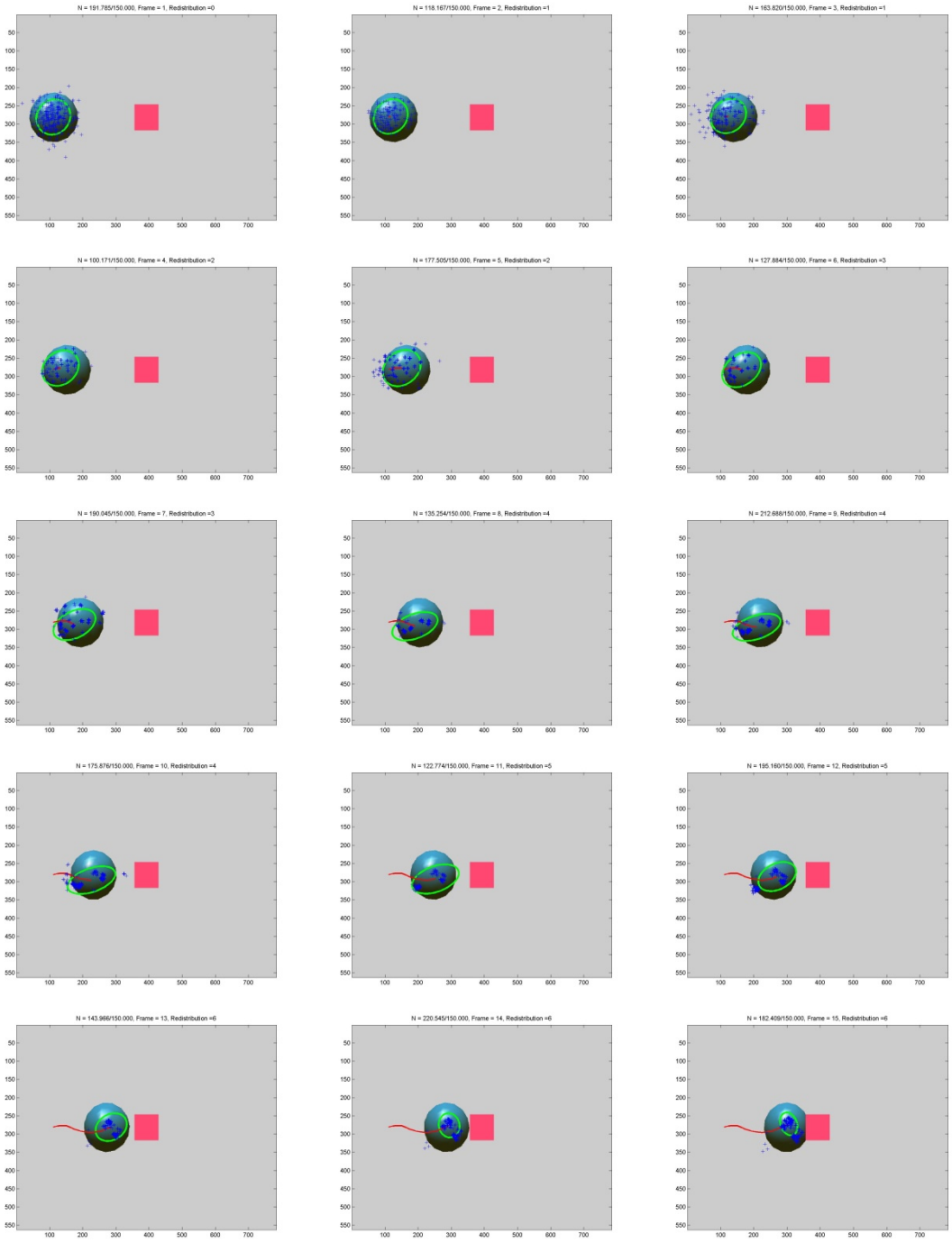


Figure B.16: Particle filter object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A6.

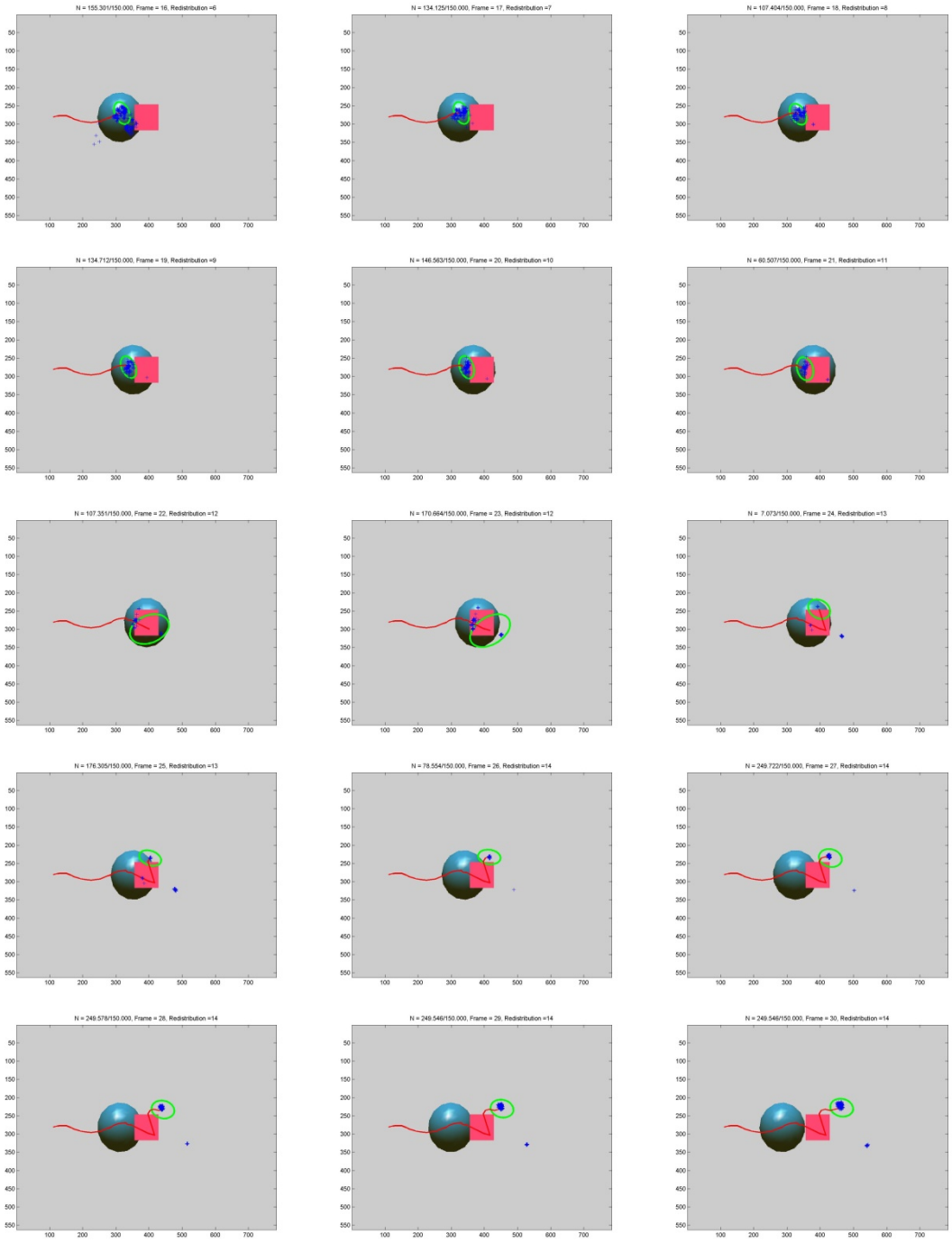


Figure B.17: Particle filter object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A6.

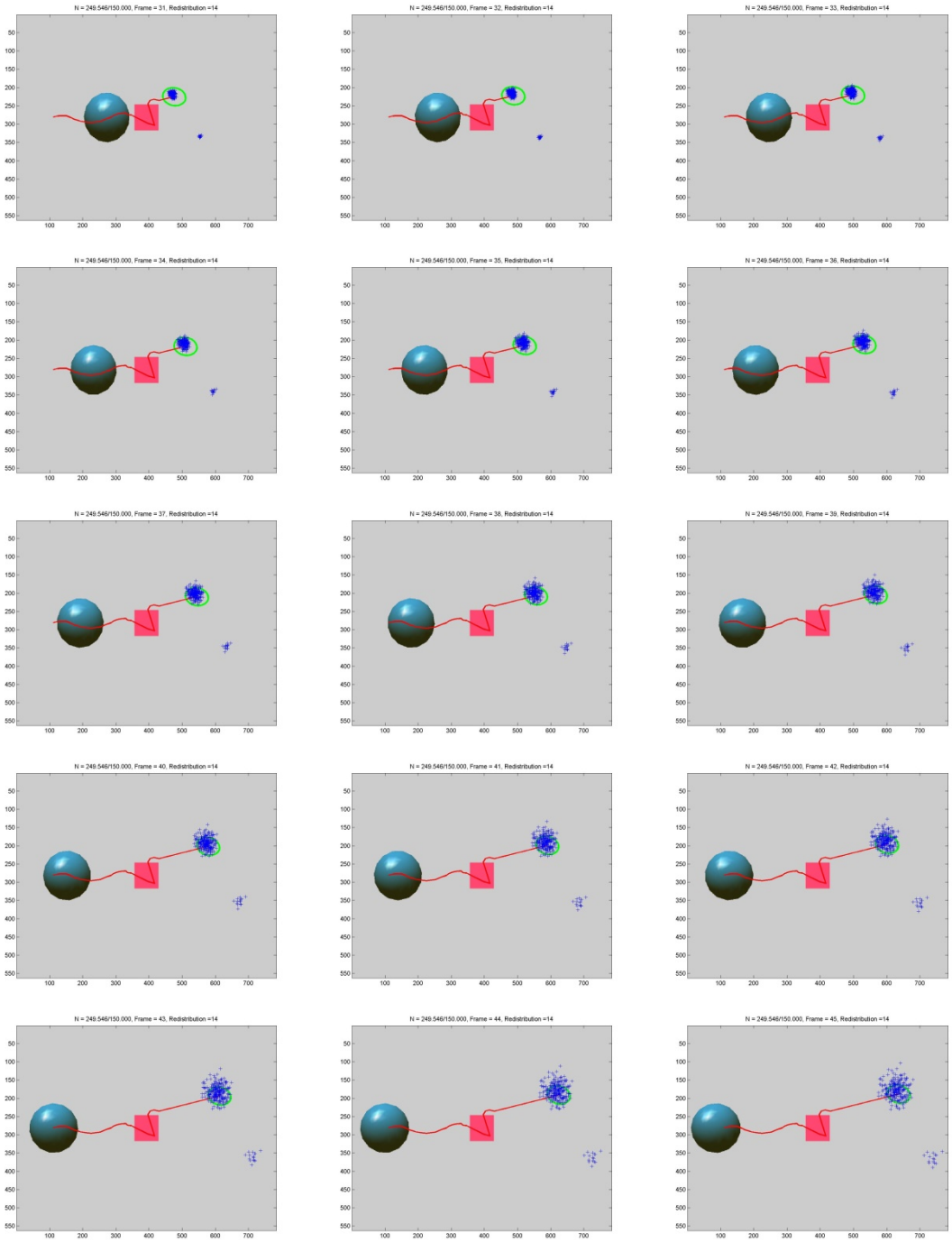


Figure B.18: Particle filter object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A6.



Figure B.19: Particle filter object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure B.20: Particle filter object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure B.21: Particle filter object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure B.22: Particle filter object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure B.23: Particle filter object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure B.24: Particle filter object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure B.25: Particle filter object tracking result when tracking a pedestrian in Frame 91 to Frame 105 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure B.26: Particle filter object tracking result when tracking a pedestrian in Frame 106 to Frame 120 (From top left to top right and then to bottom) in Video sequence PETS 2009.

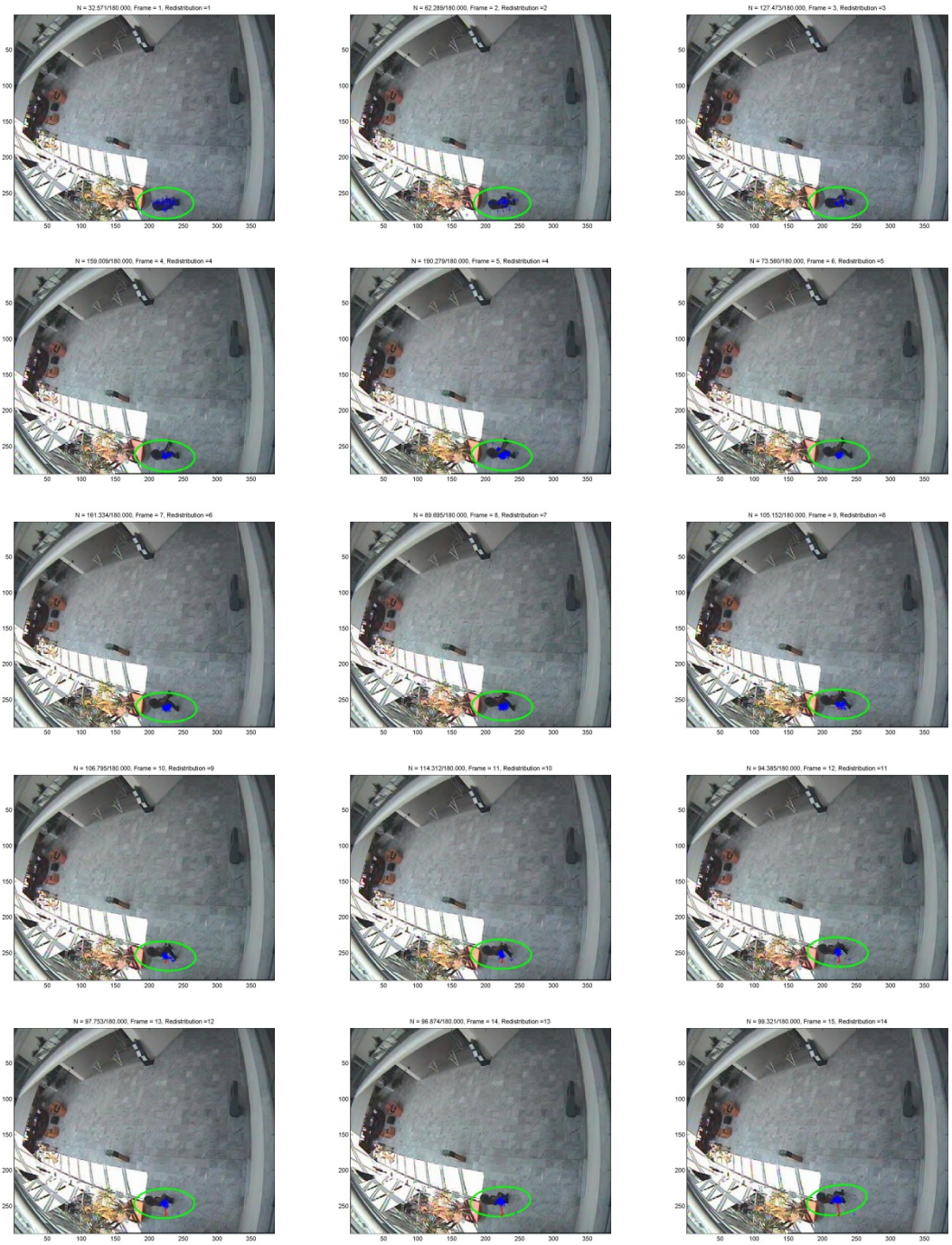


Figure B.27: Particle filter object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence CAVIAR.

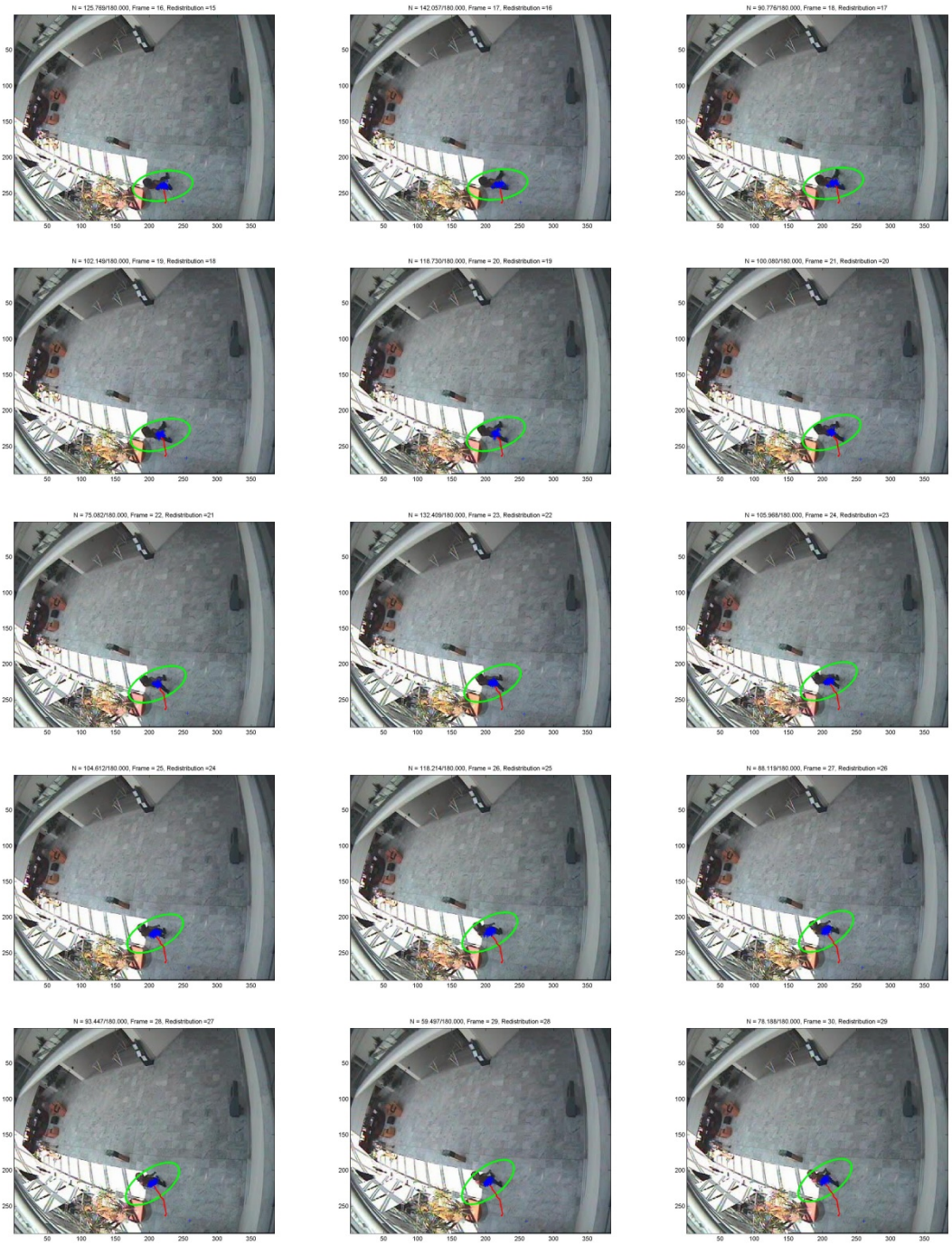


Figure B.28: Particle filter object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence CAVIAR.

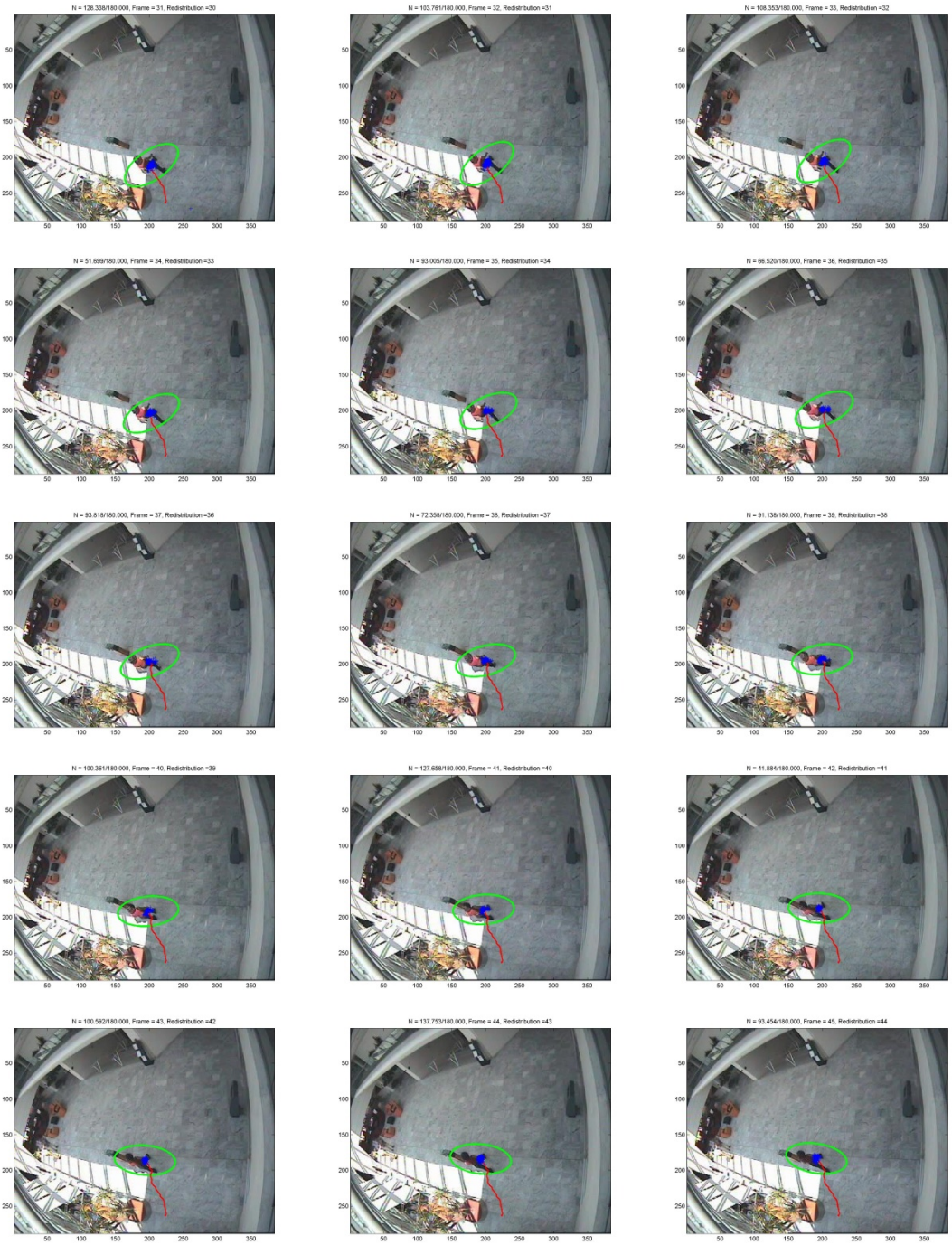


Figure B.29: Particle filter object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence CAVIAR.

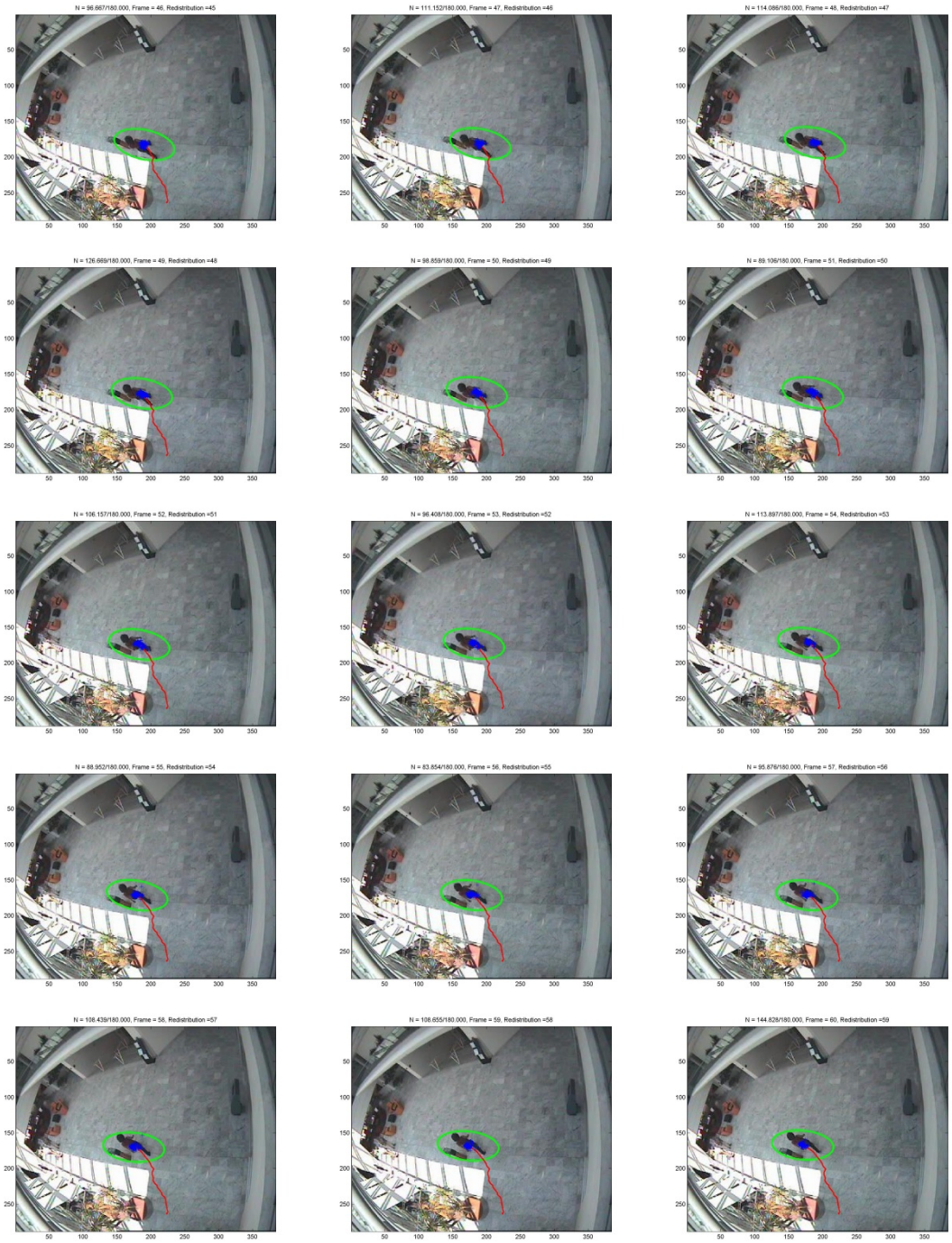


Figure B.30: Particle filter object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence CAVIAR.

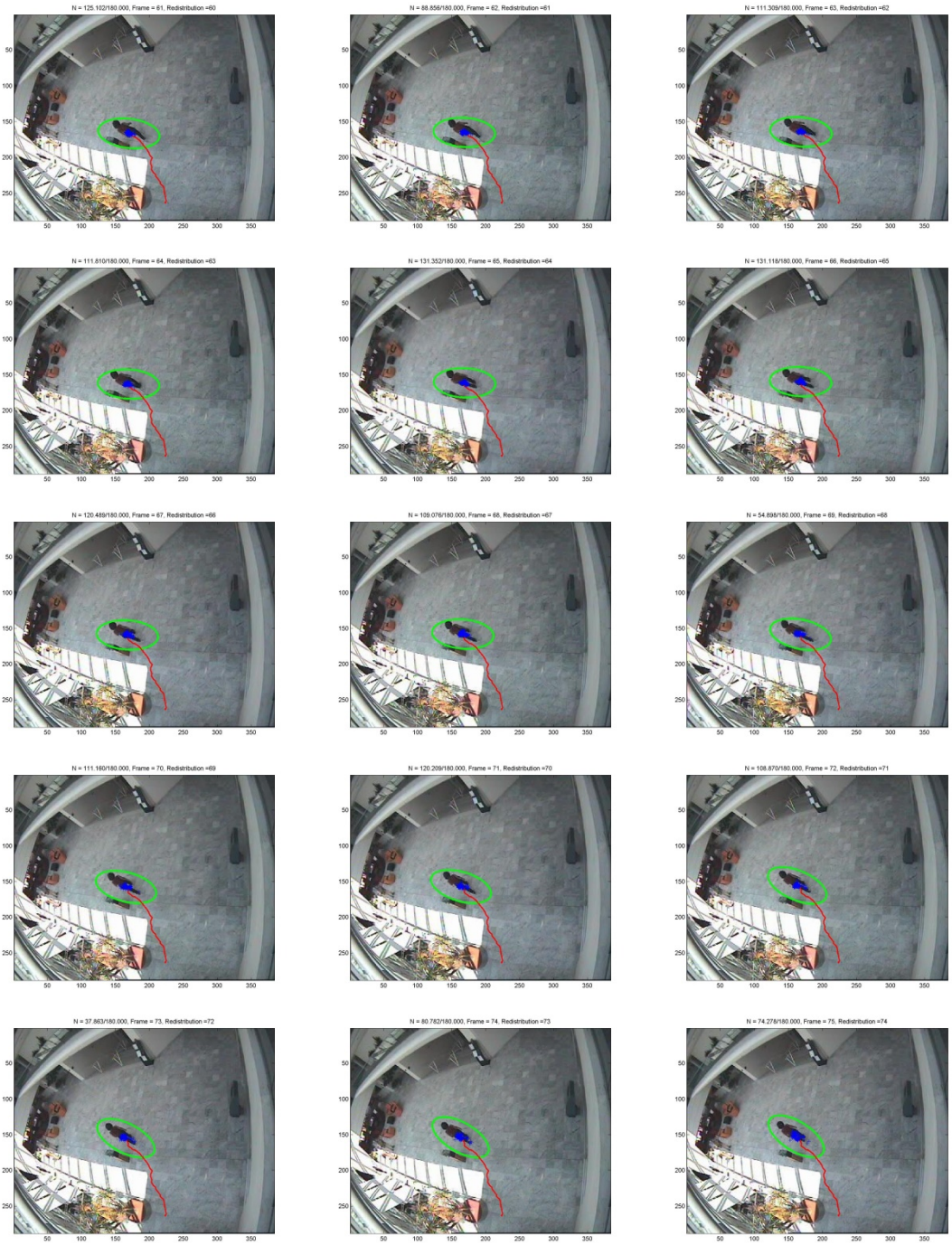


Figure B.31: Particle filter object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence CAVIAR.

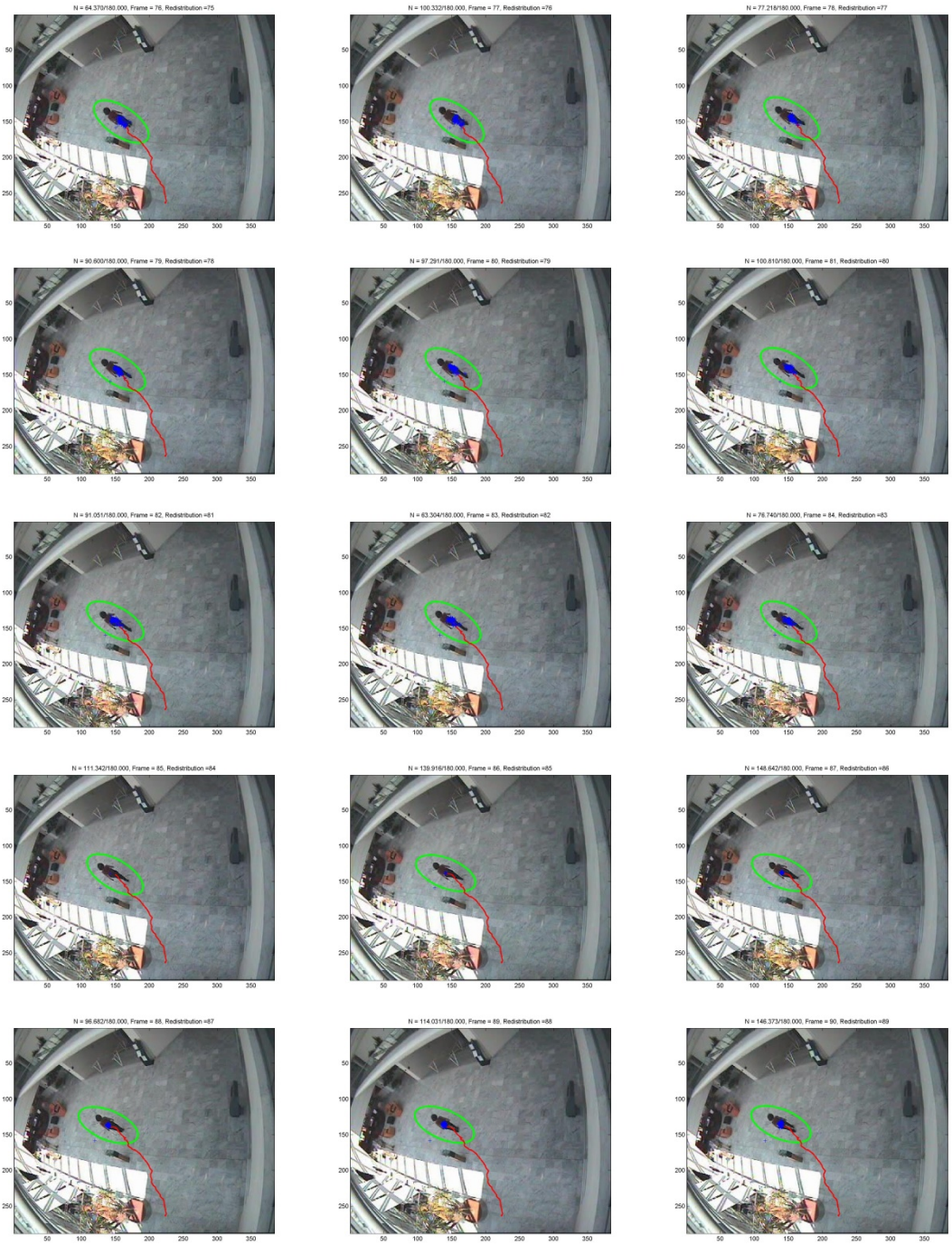


Figure B.32: Particle filter object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence CAVIAR.

Appendix C:
Mean Shift Object Tracking Result Frame by Frame

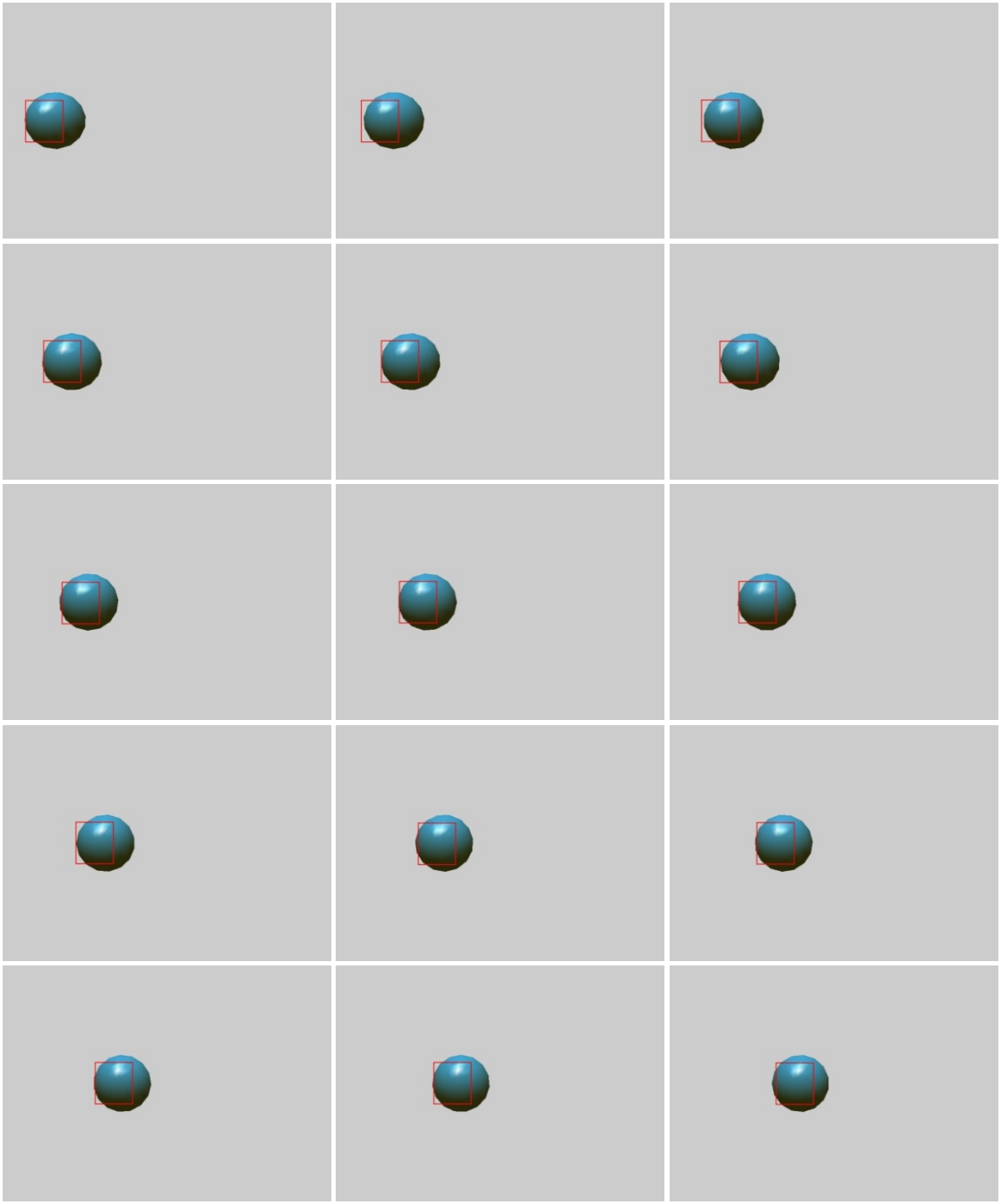


Figure C.1: Mean shift object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A1.

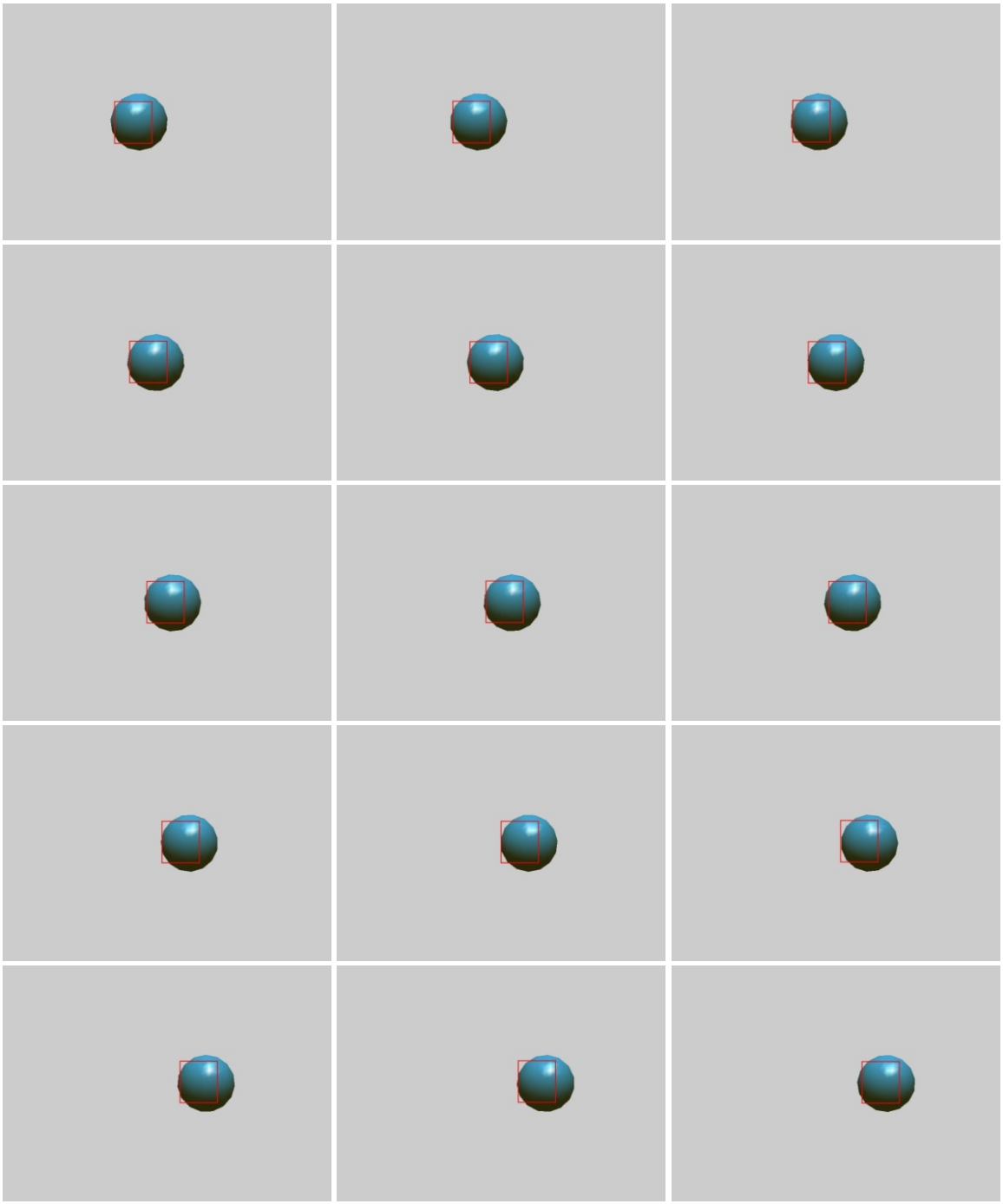


Figure C.2: Mean shift object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A1.

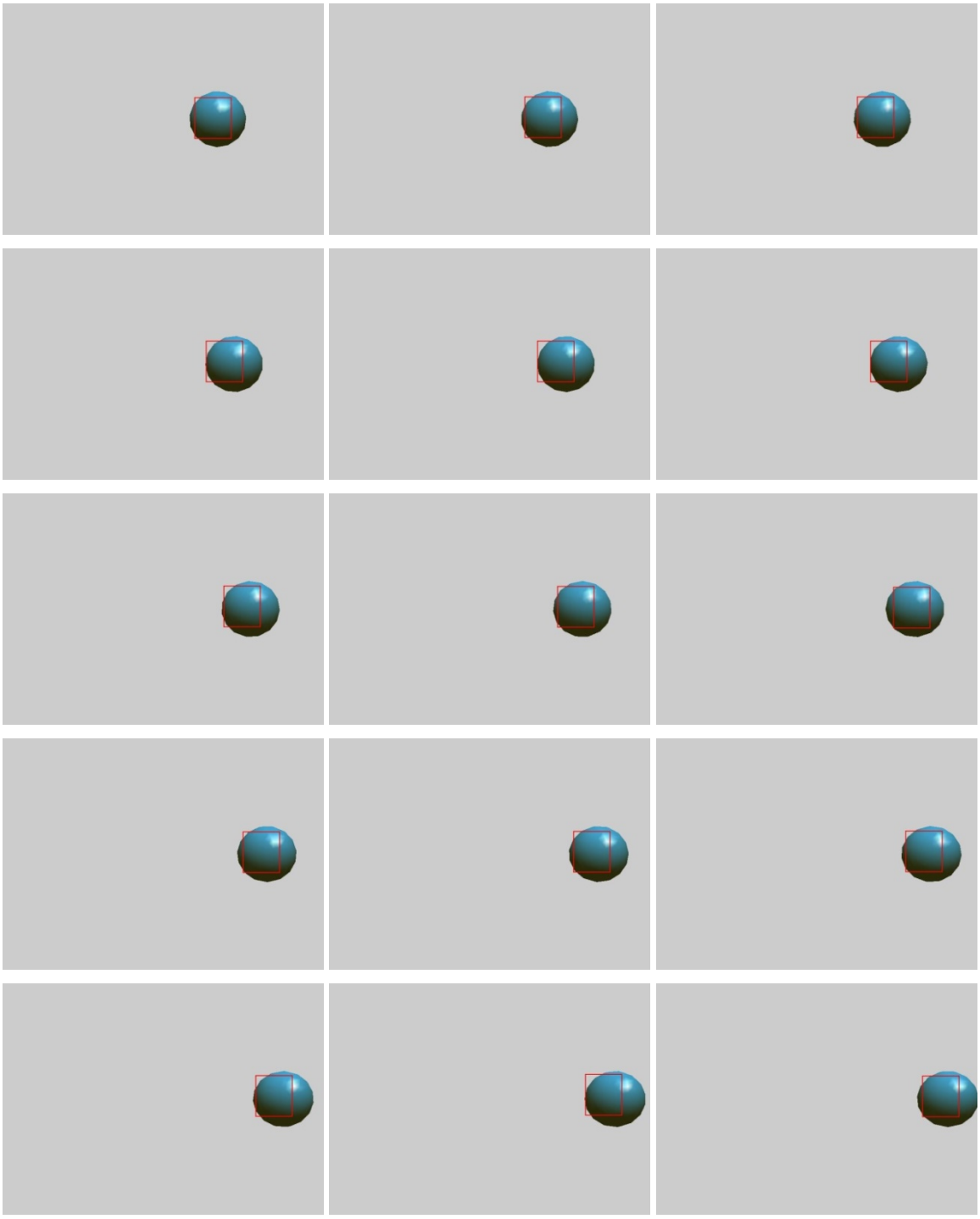


Figure C.3: Mean shift object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A1.

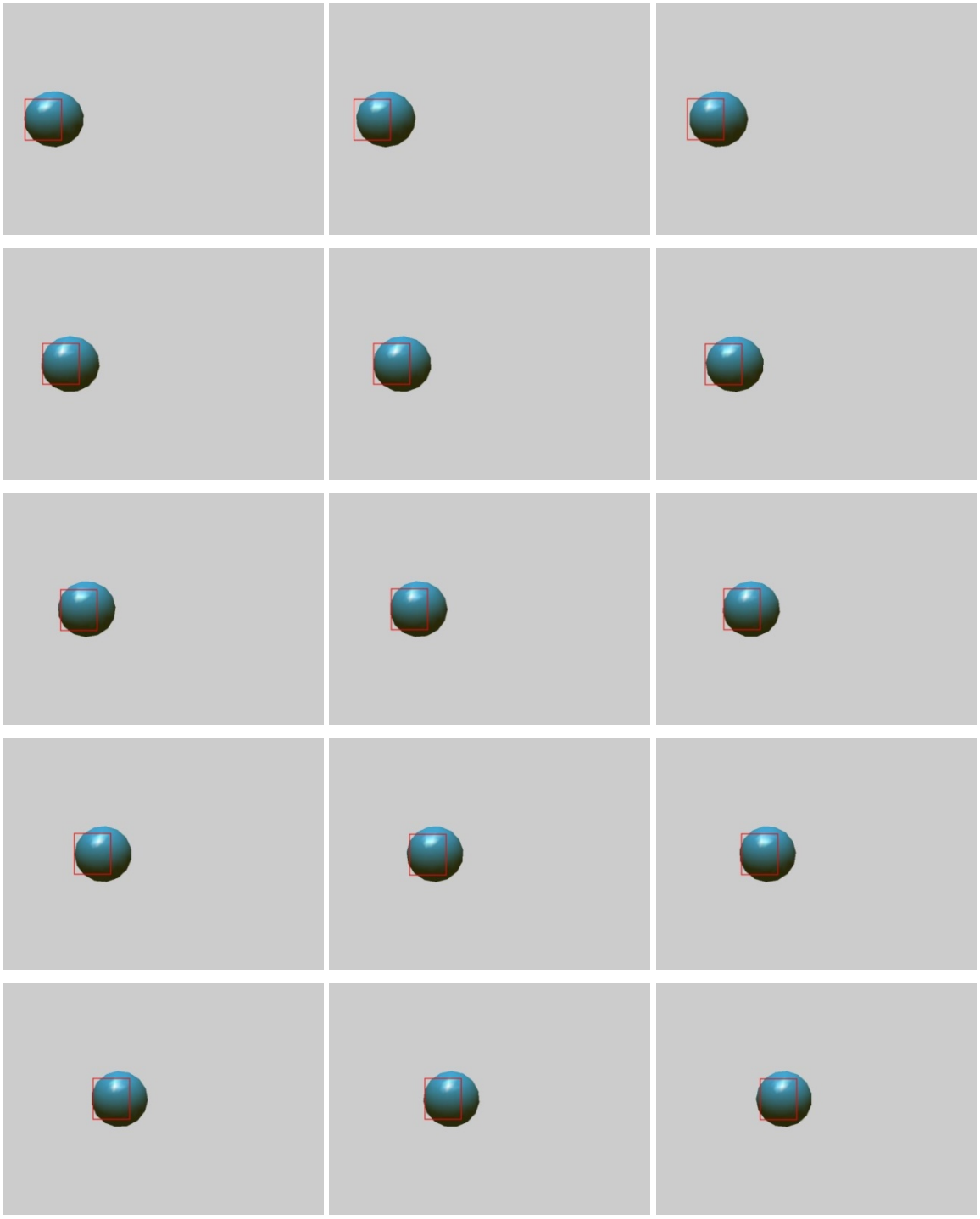


Figure C.4: Mean shift object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A2.

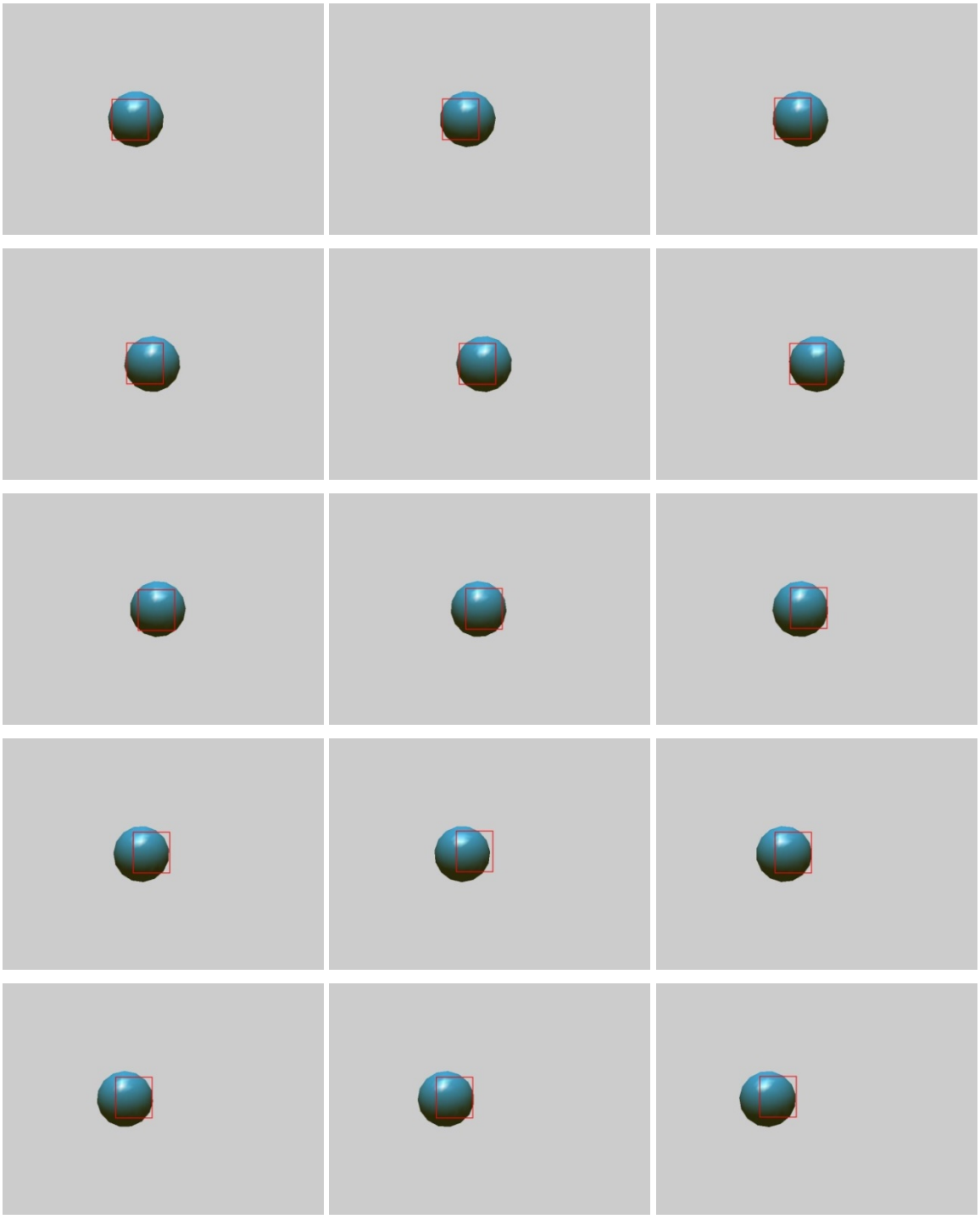


Figure C.5: Mean shift object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A2.

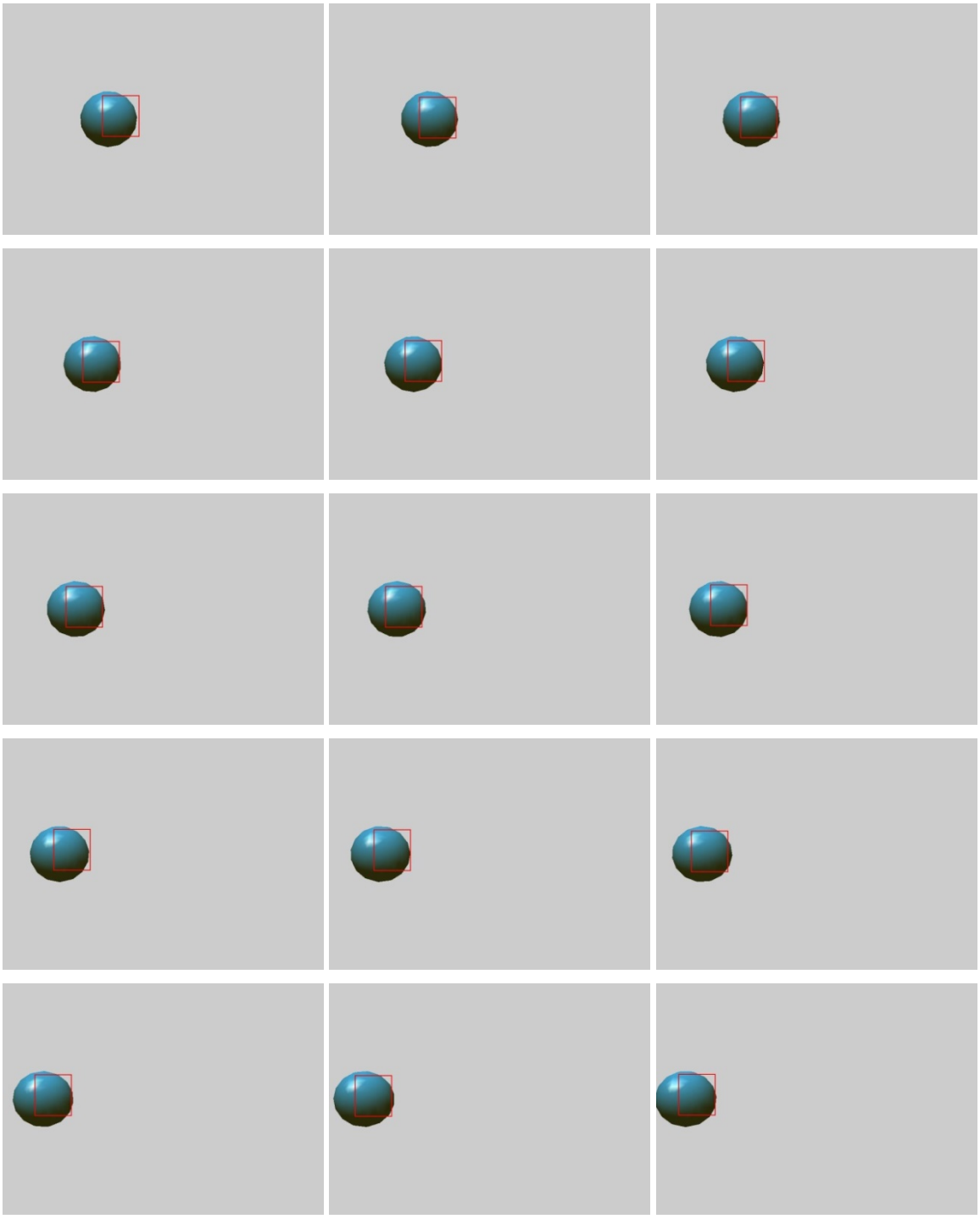


Figure C.6: Mean shift object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A2.

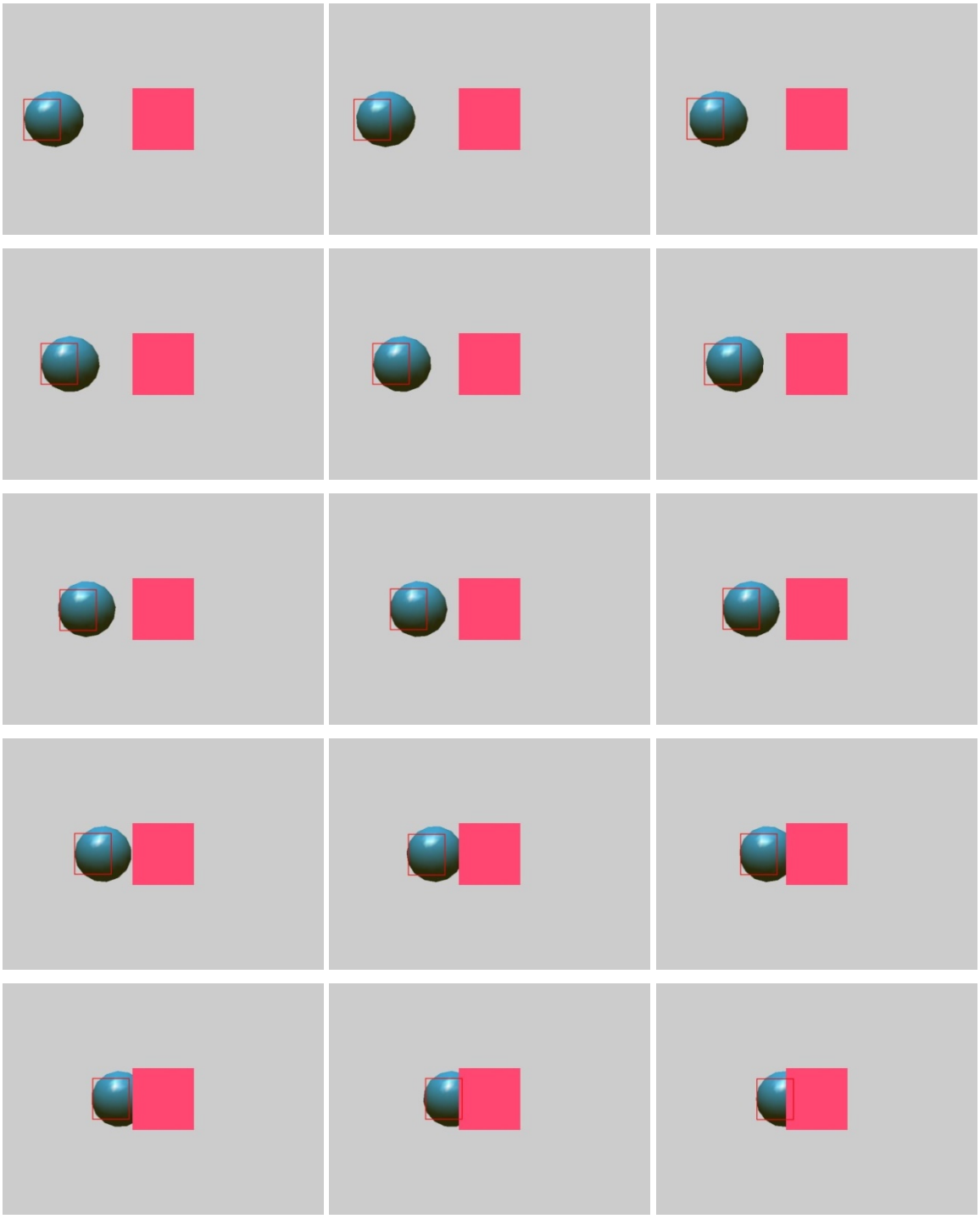


Figure C.7: Mean shift object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A3.

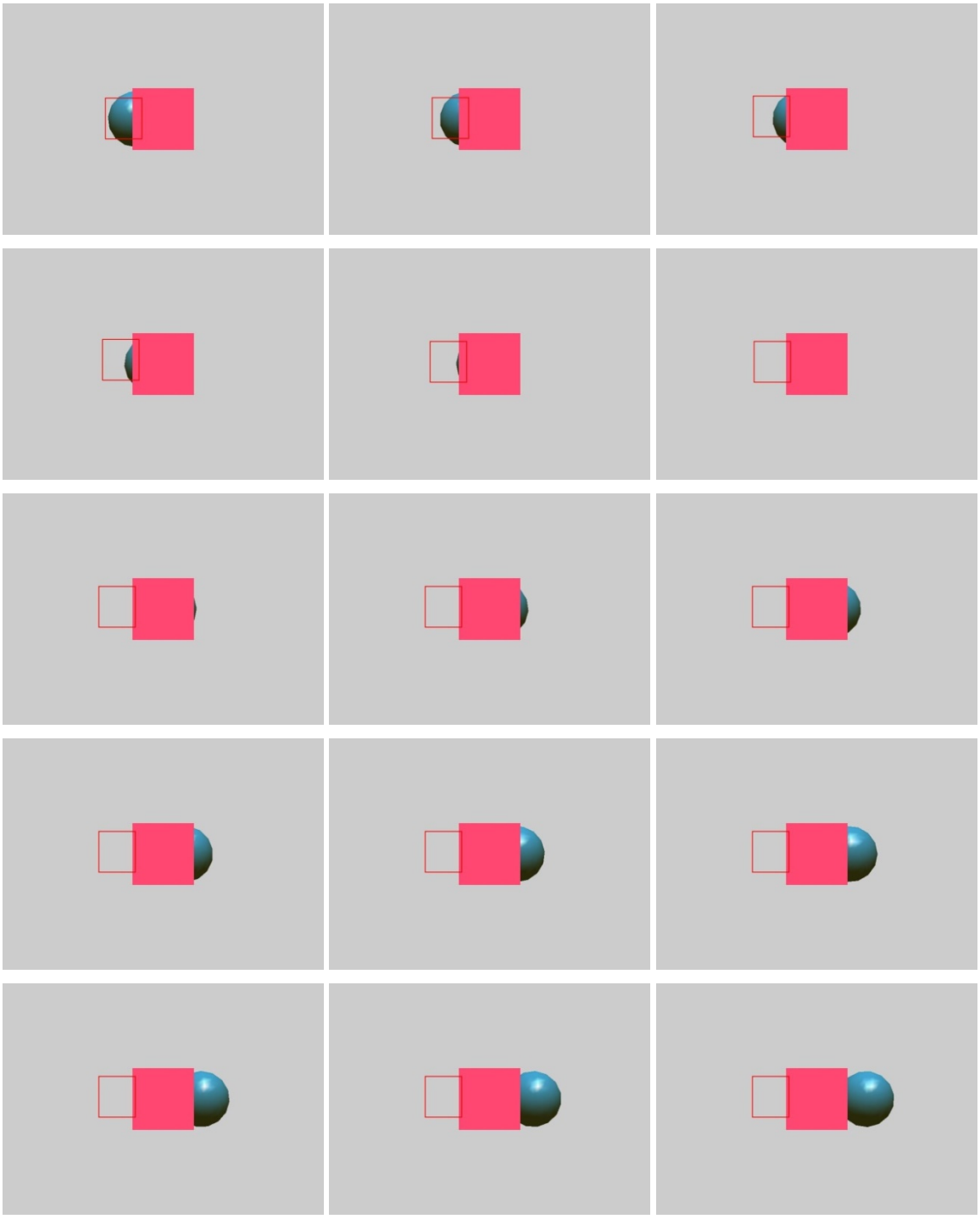


Figure C.8: Mean shift object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A3.

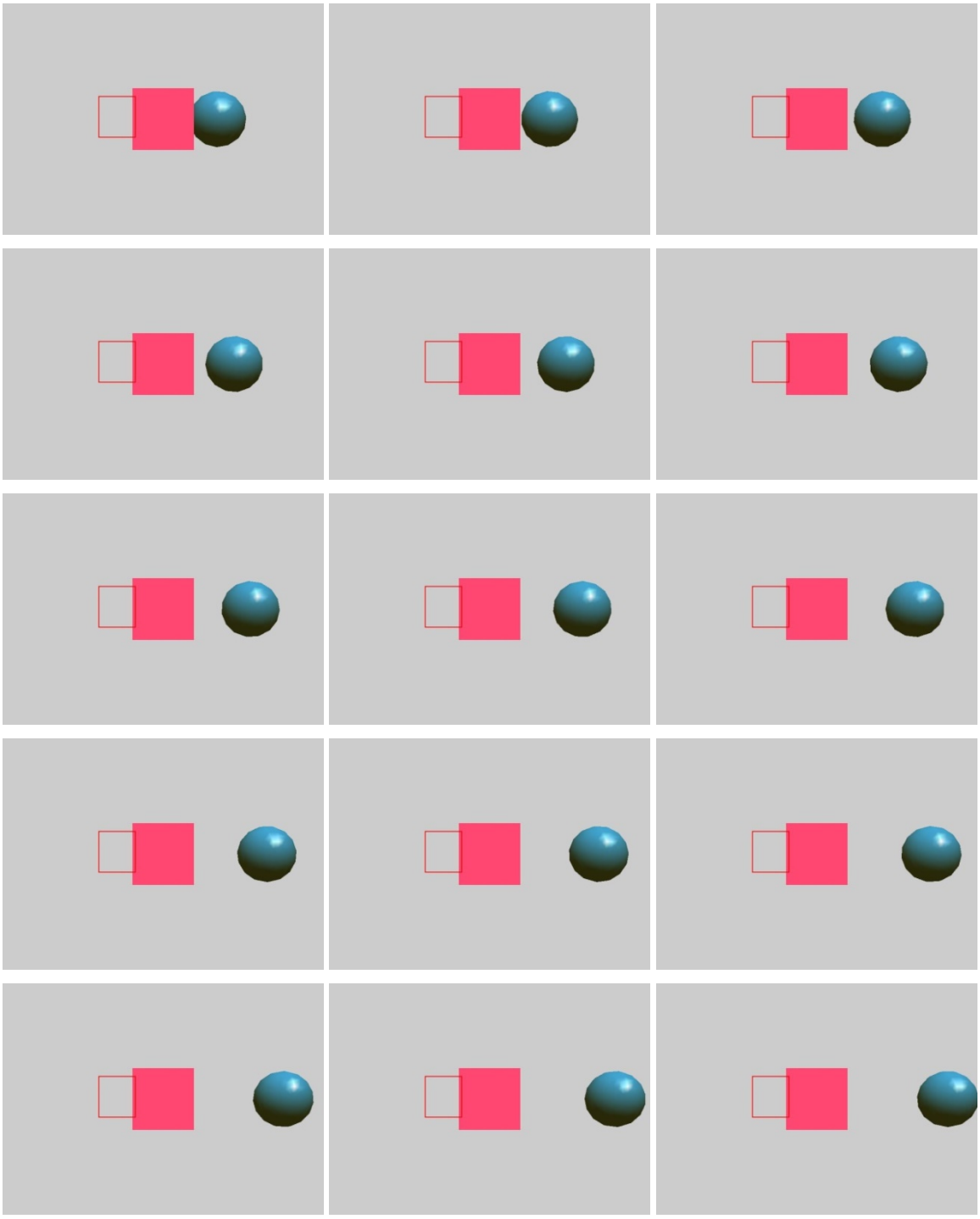


Figure C.9: Mean shift object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A3.

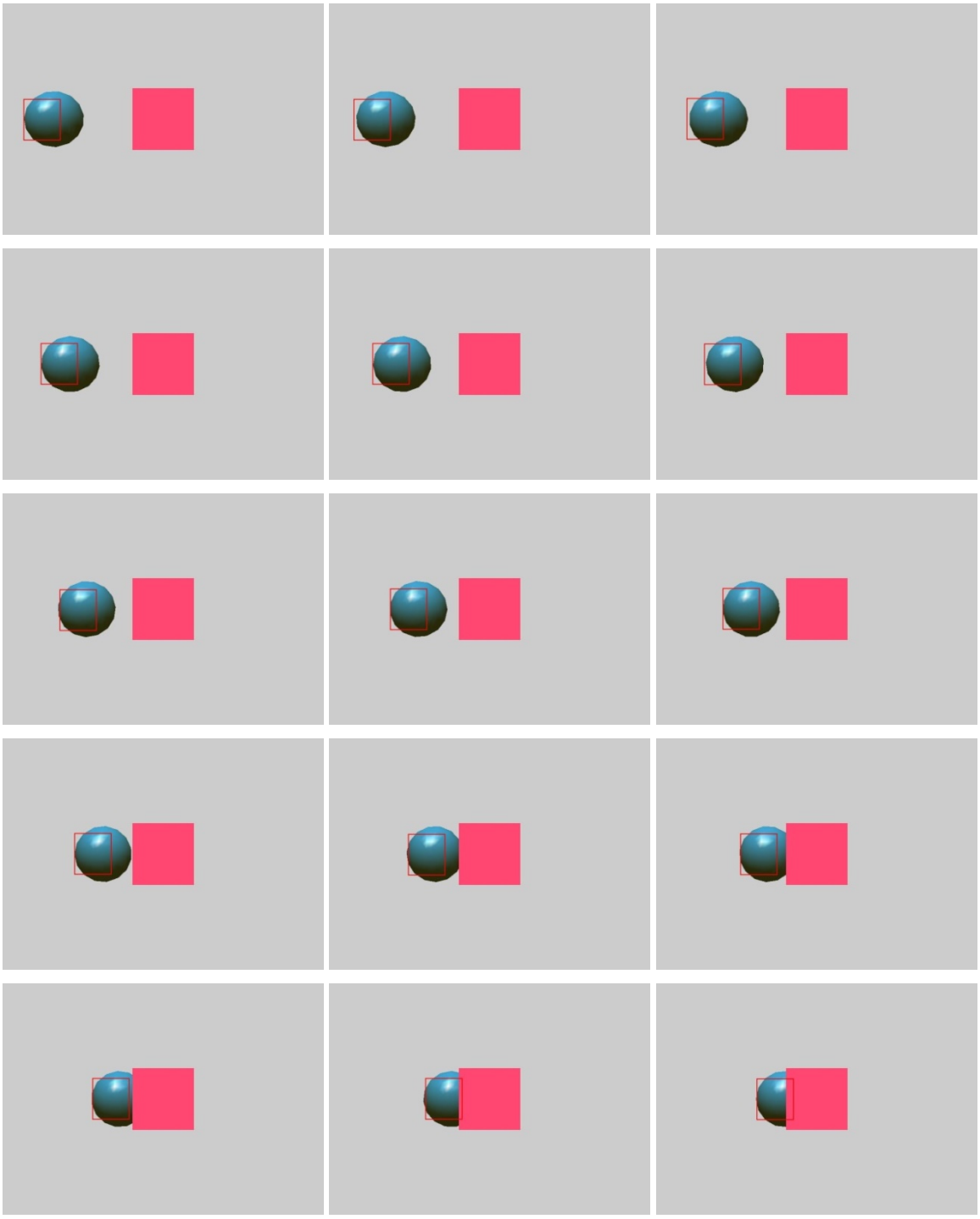


Figure C.10: Mean shift object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A4.

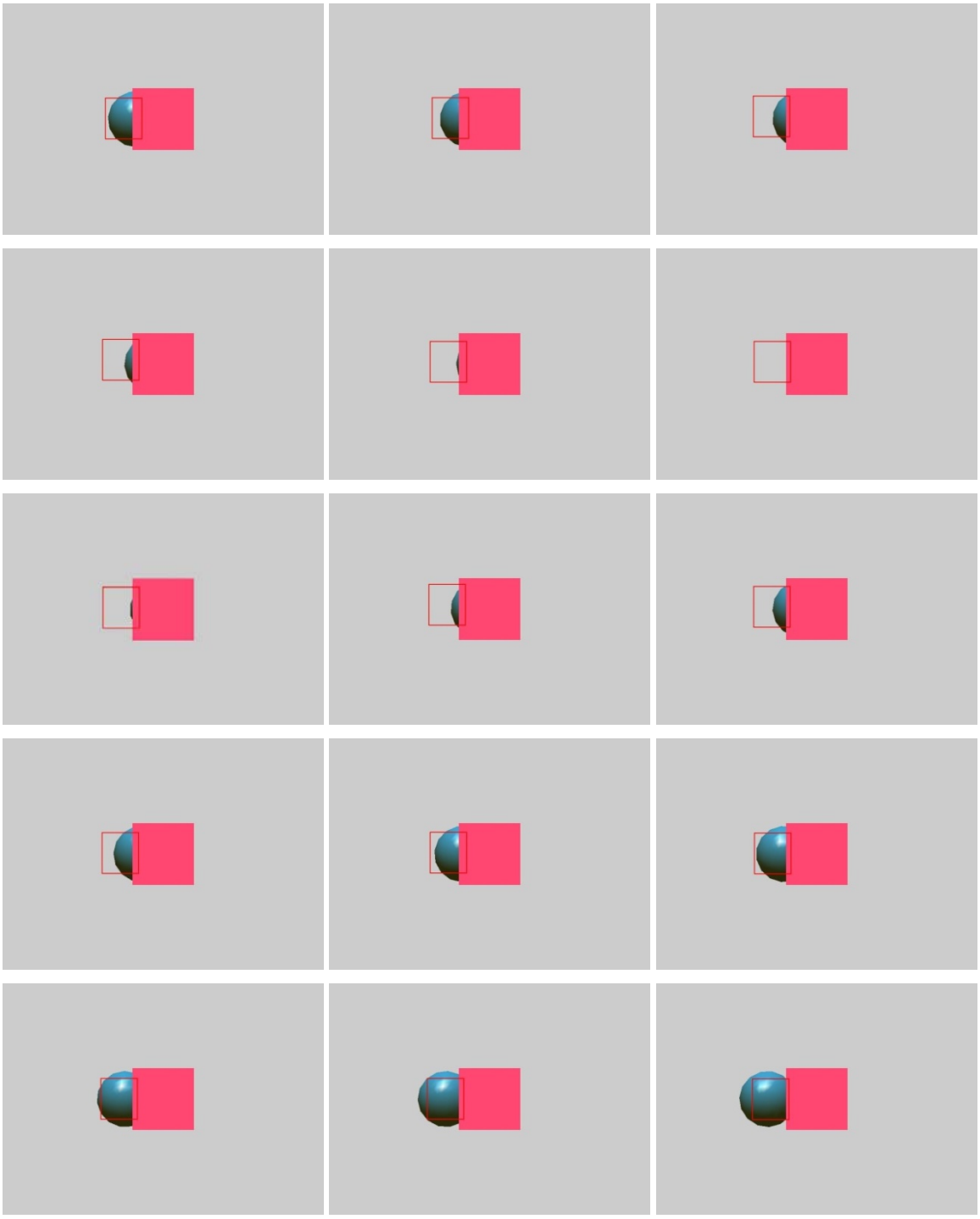


Figure C.11: Mean shift object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A4.

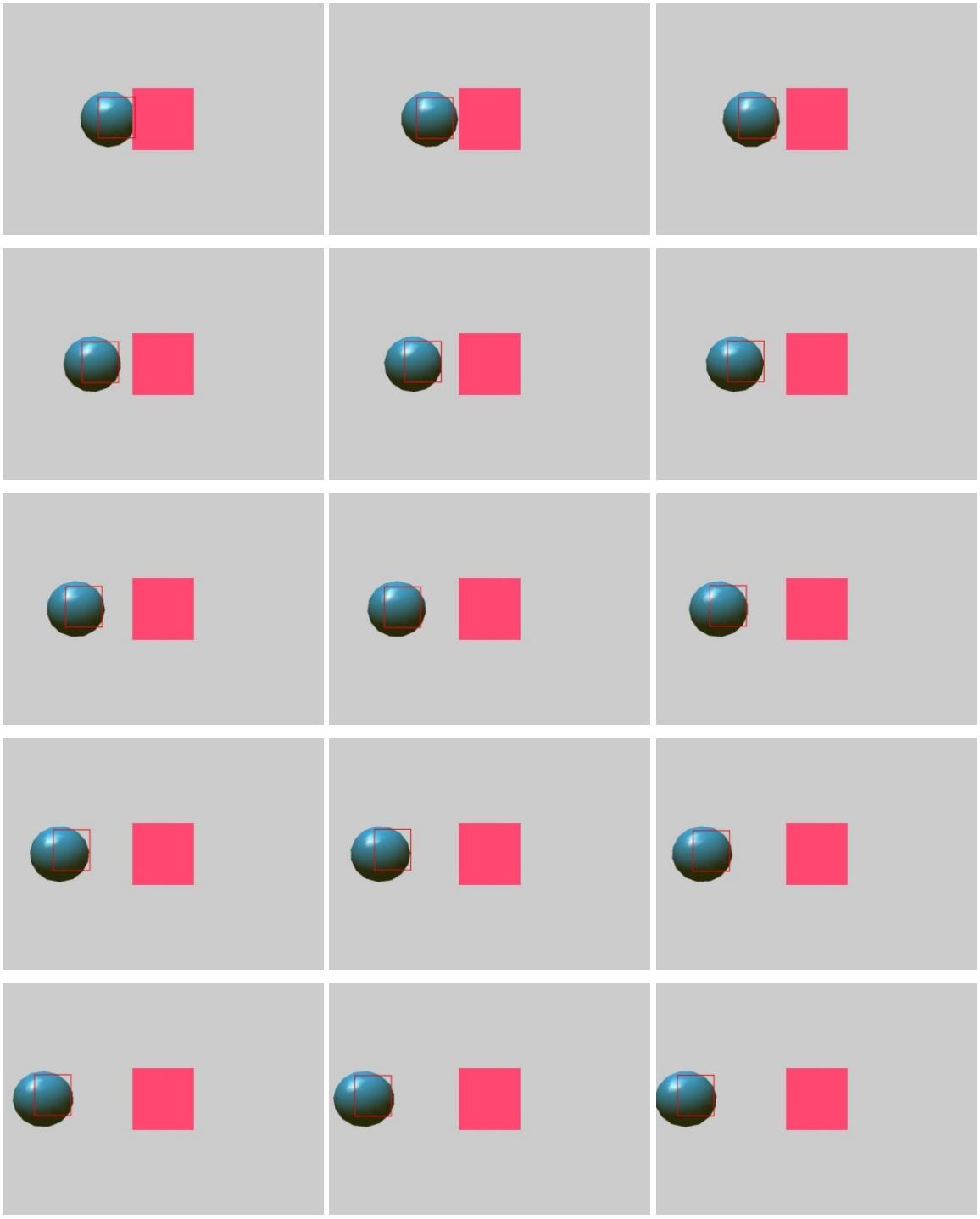


Figure C.12: Mean shift object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A4.

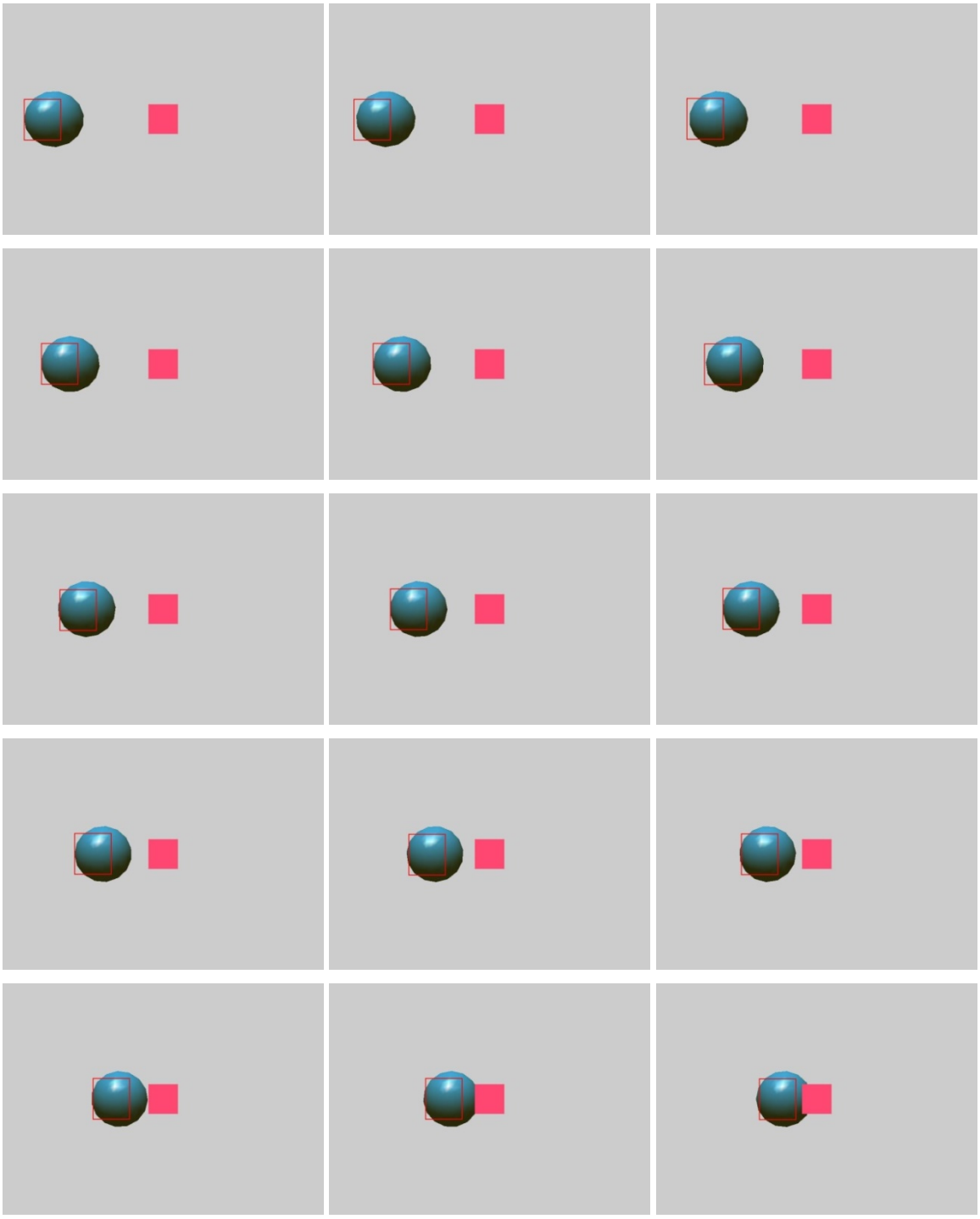


Figure C.13: Mean shift object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A5.

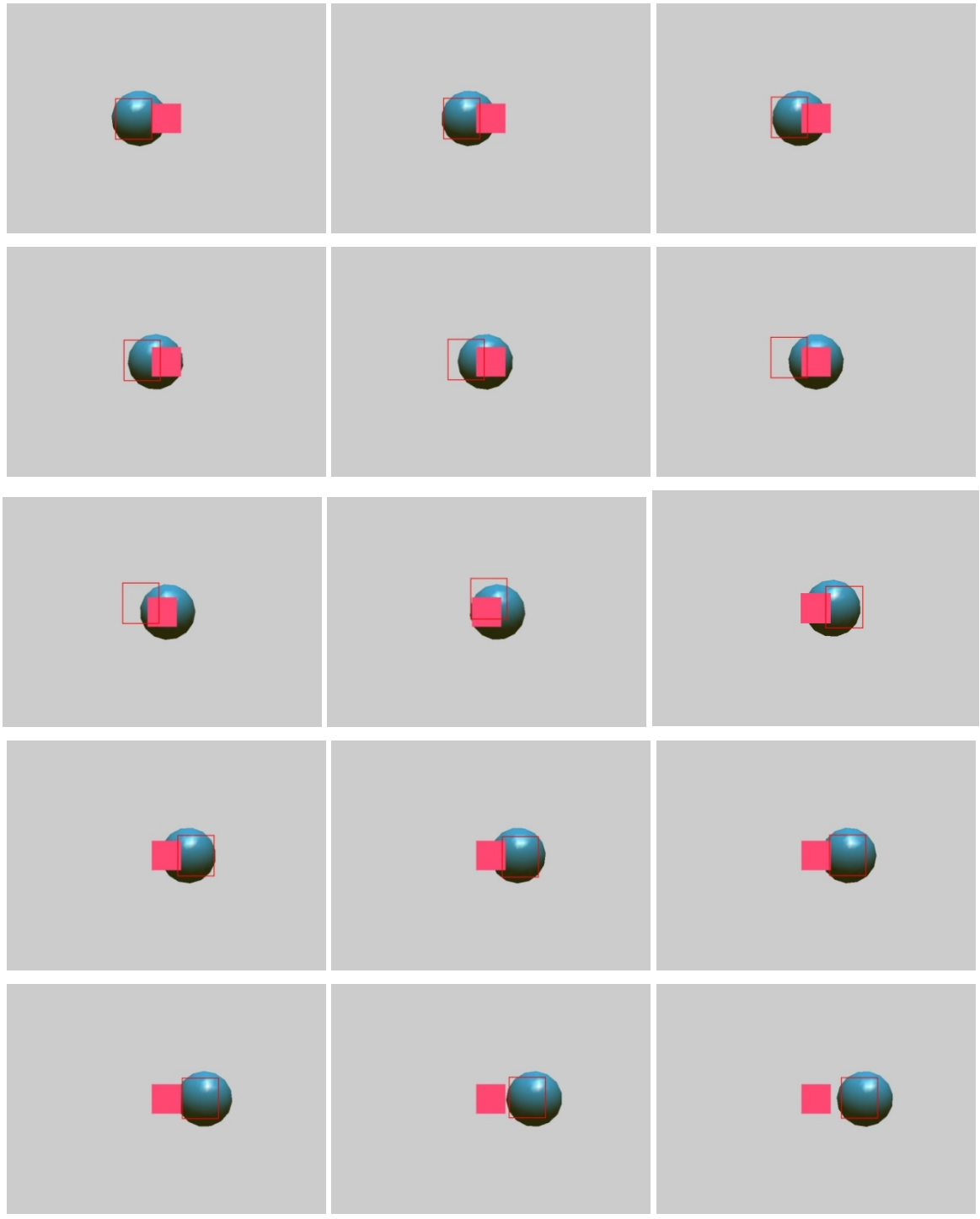


Figure C.14: Mean shift object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A5.

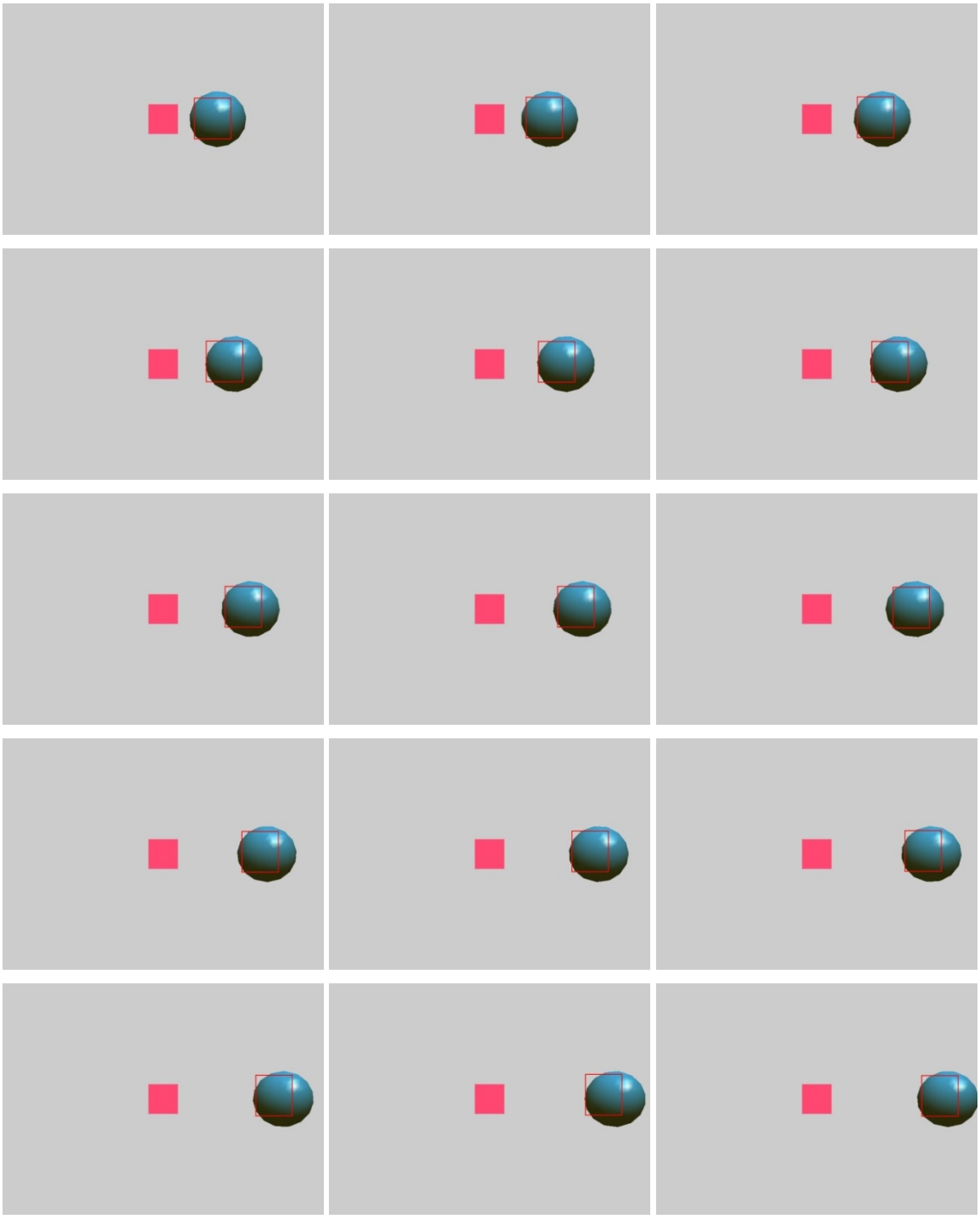


Figure C.15: Mean shift object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A5.

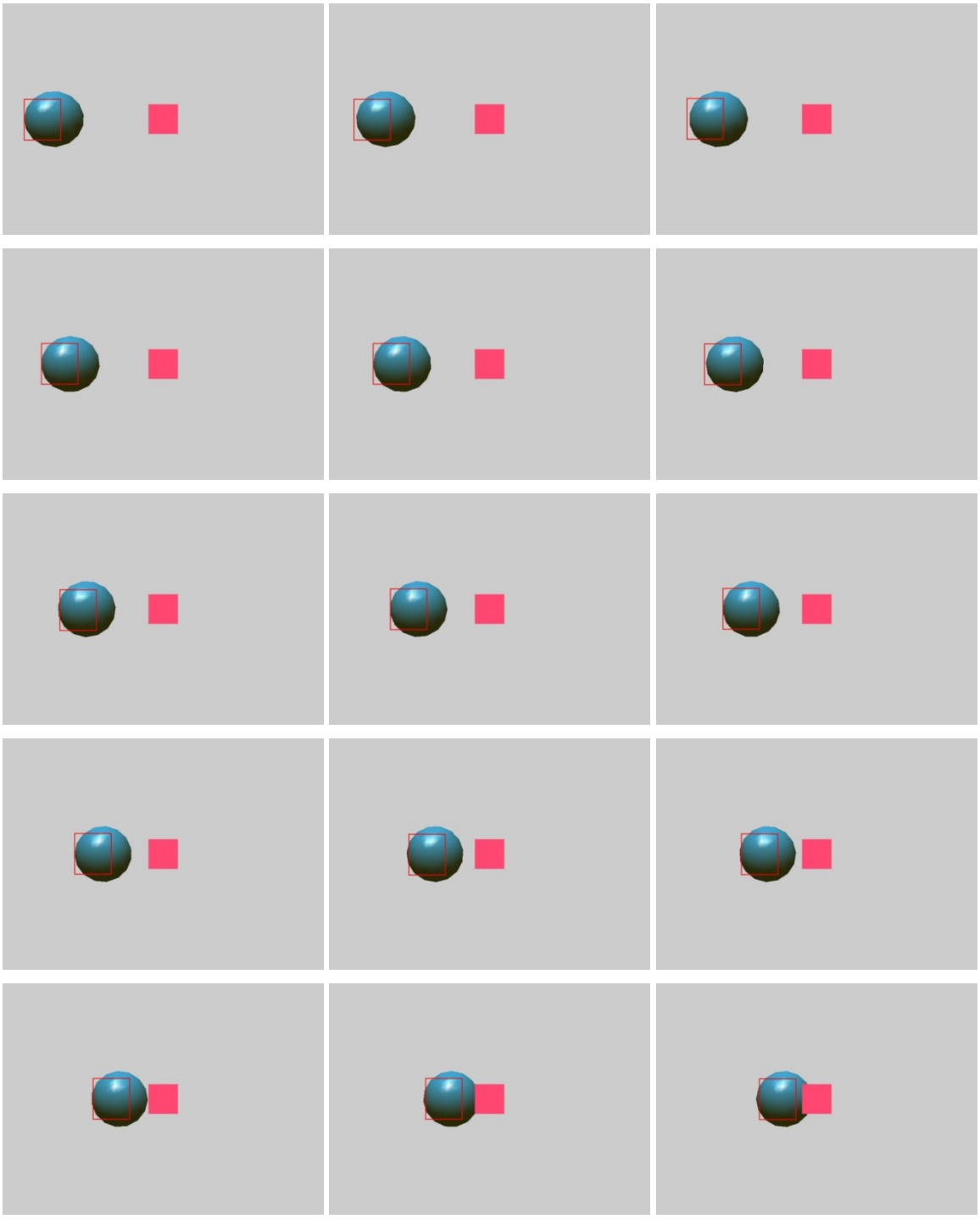


Figure C.16: Mean shift object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A6.

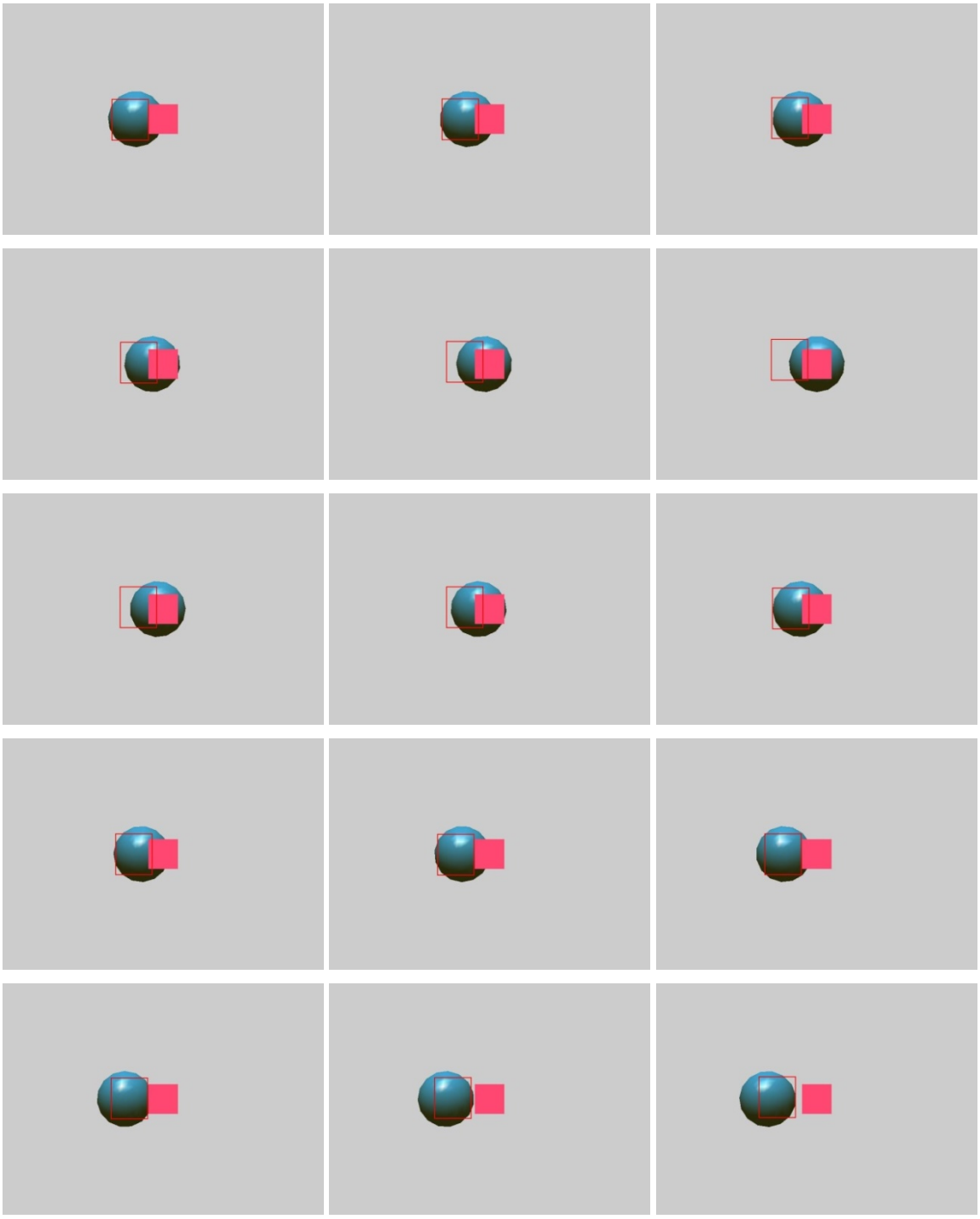


Figure C.17: Mean shift object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A6

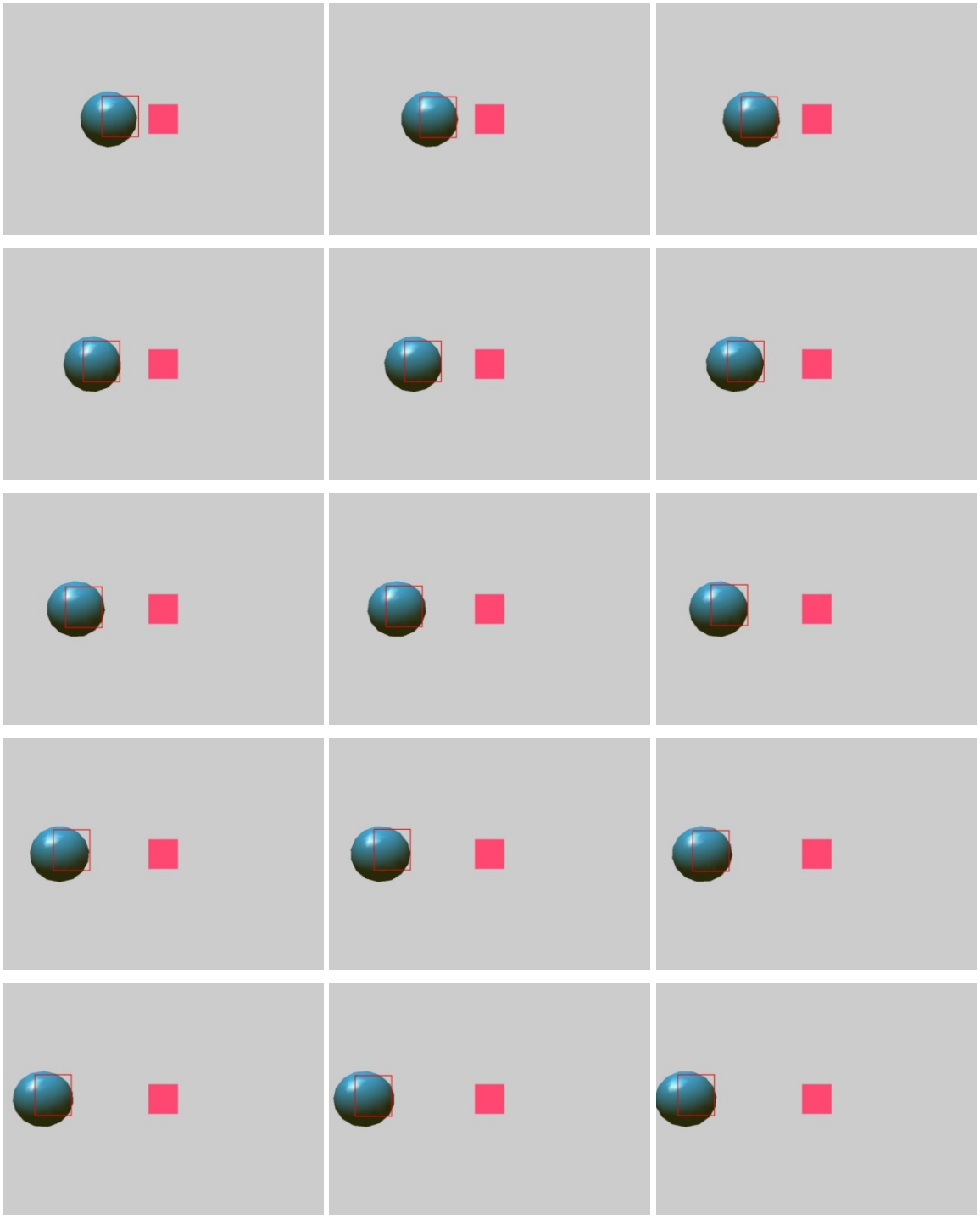


Figure C.18: Mean shift object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A6.

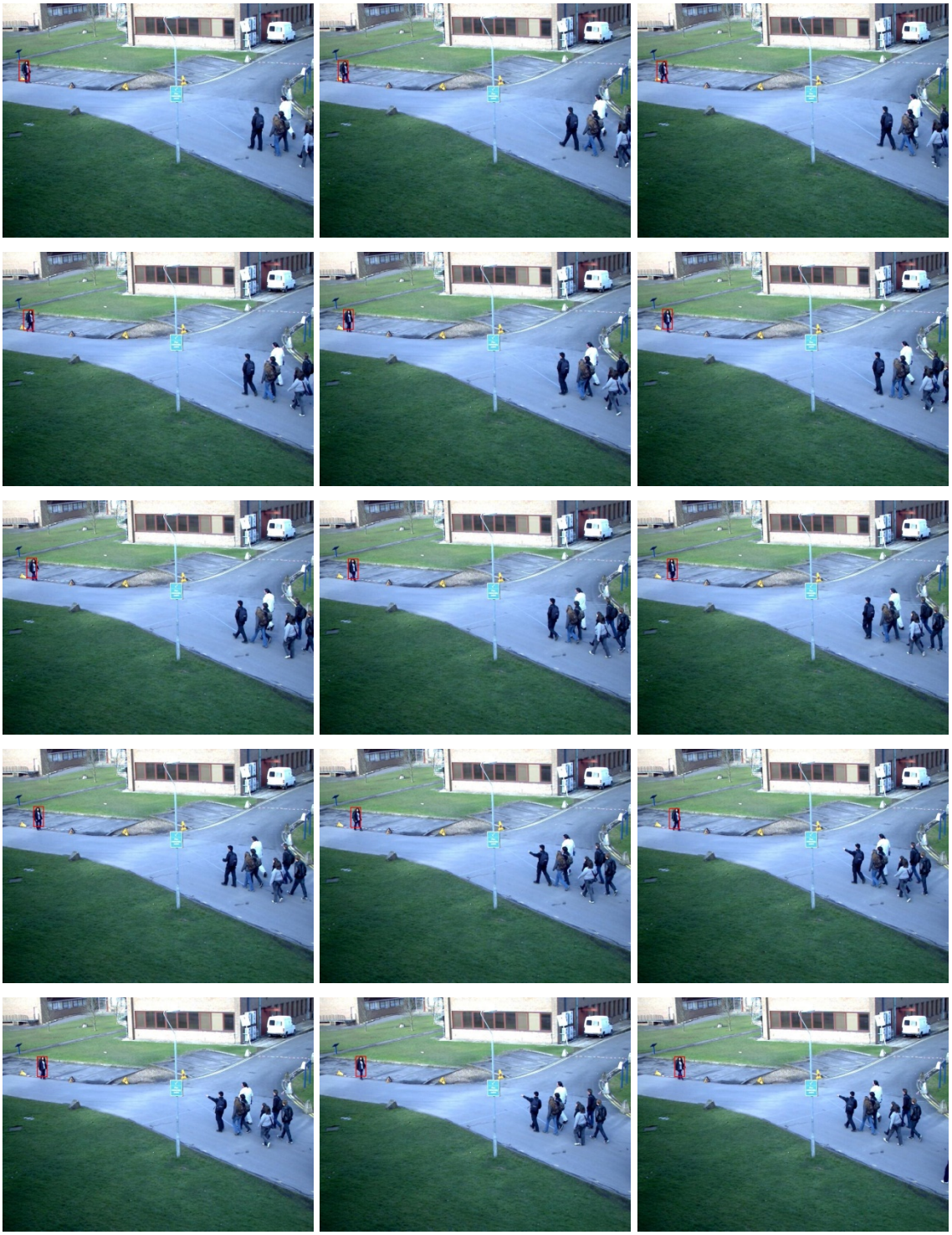


Figure C.19: Mean shift object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence PETS 2009.

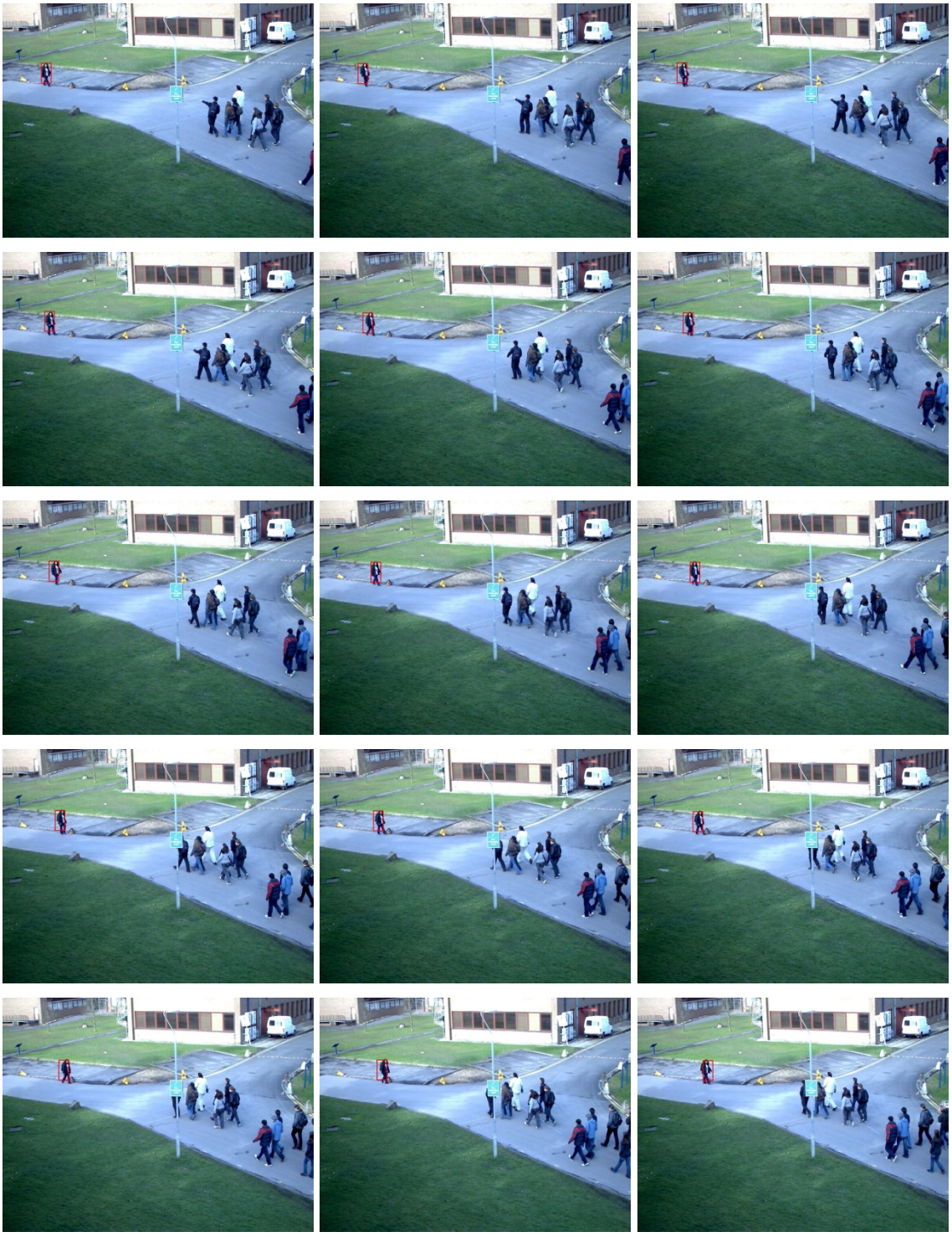


Figure C.20: Mean shift object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure C.21: Mean shift object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence PETS 2009.

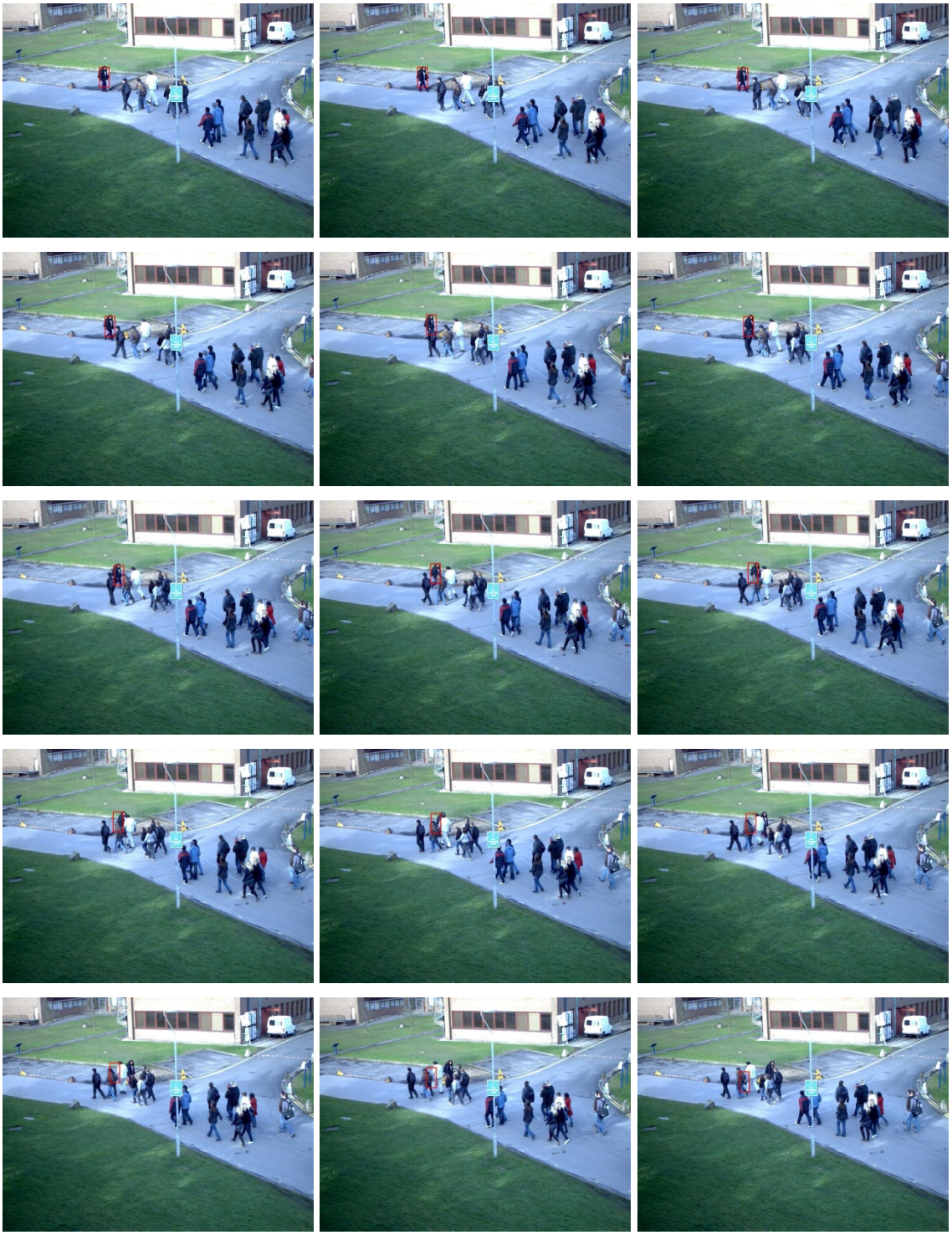


Figure C.22: Mean shift object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence PETS 2009.

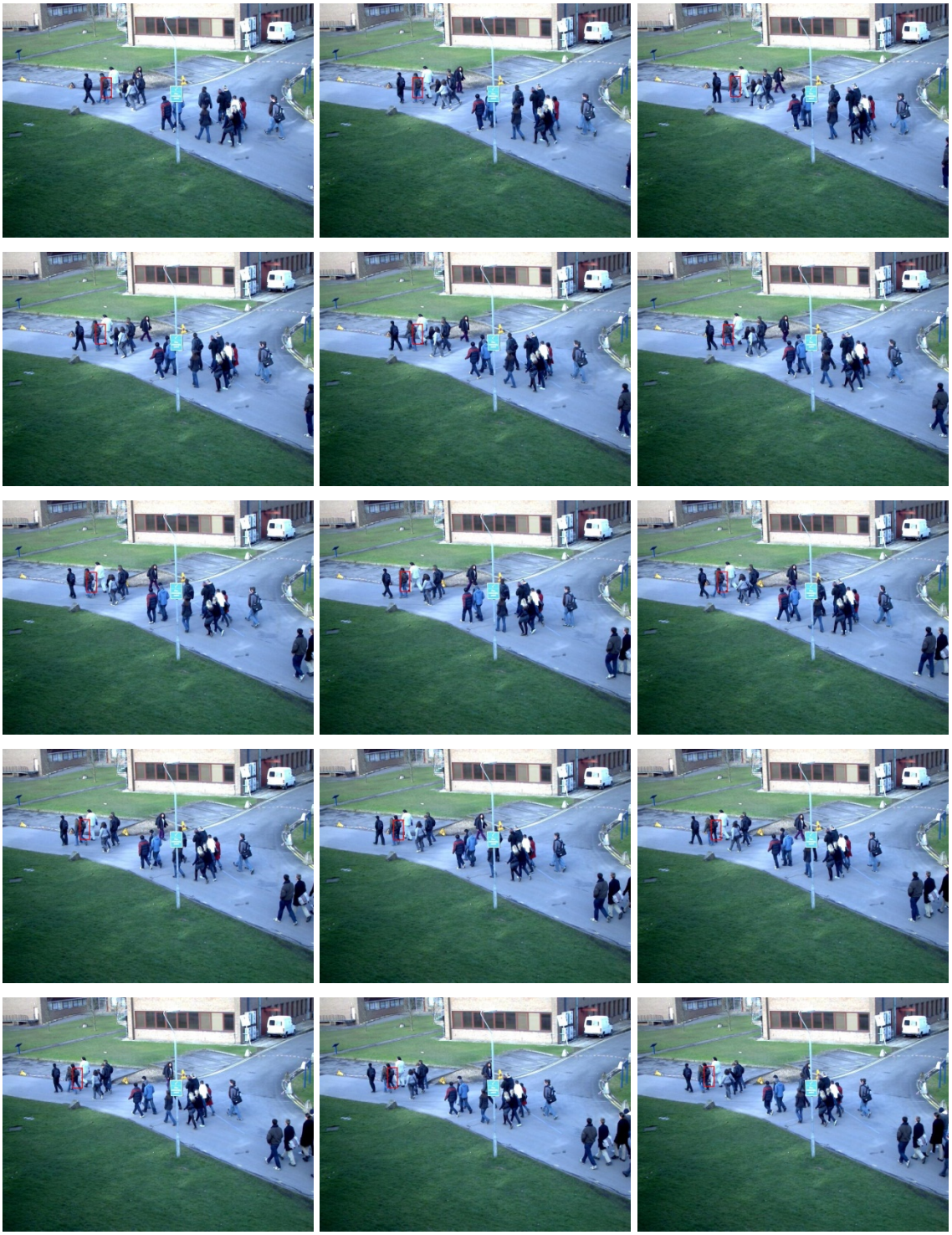


Figure C.23: Mean shift object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence PETS 2009.

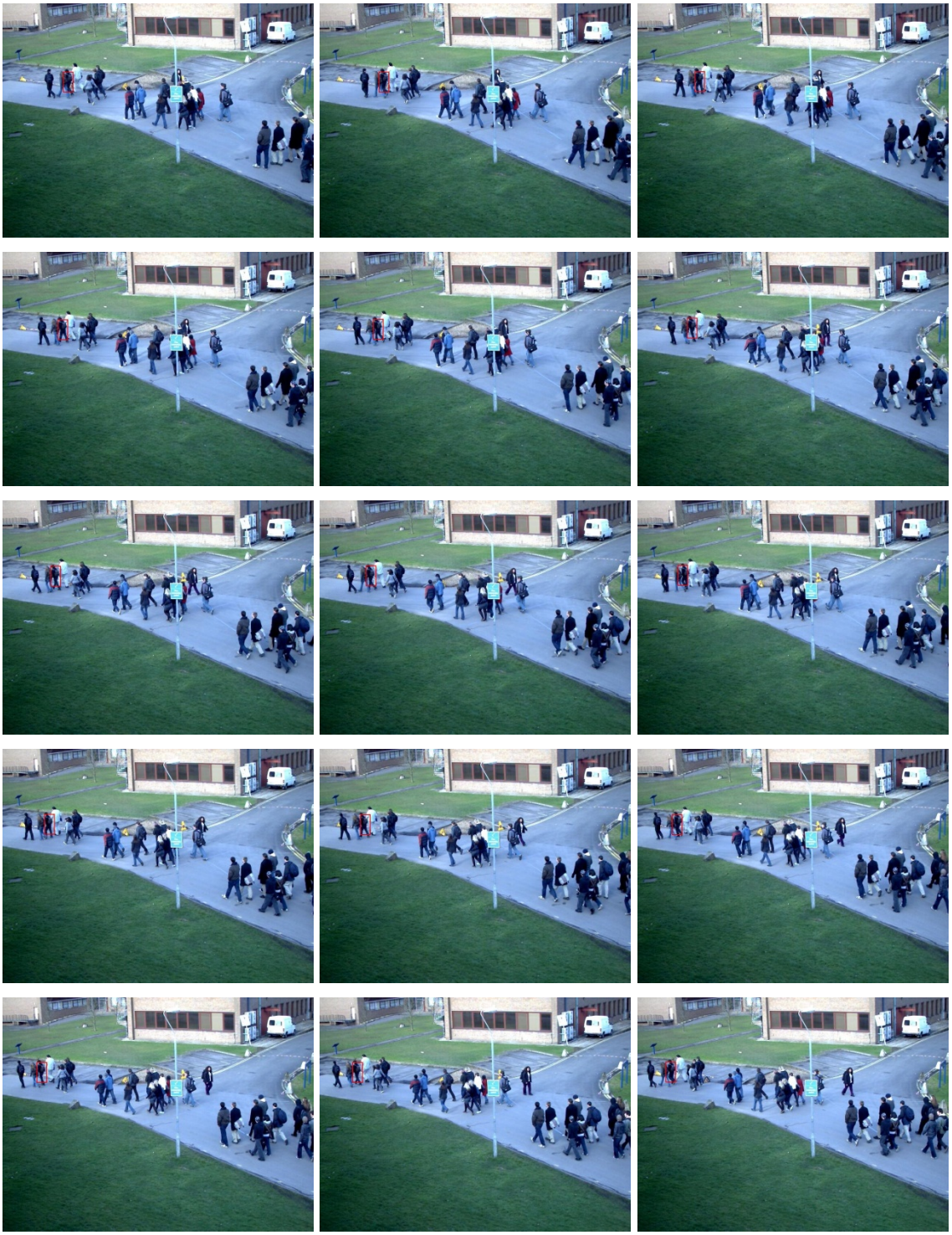


Figure C.24: Mean shift object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure C.25: Mean shift object tracking result when tracking a pedestrian in Frame 91 to Frame 101 (From top left to top right and then to bottom) in Video sequence PETS 2009.

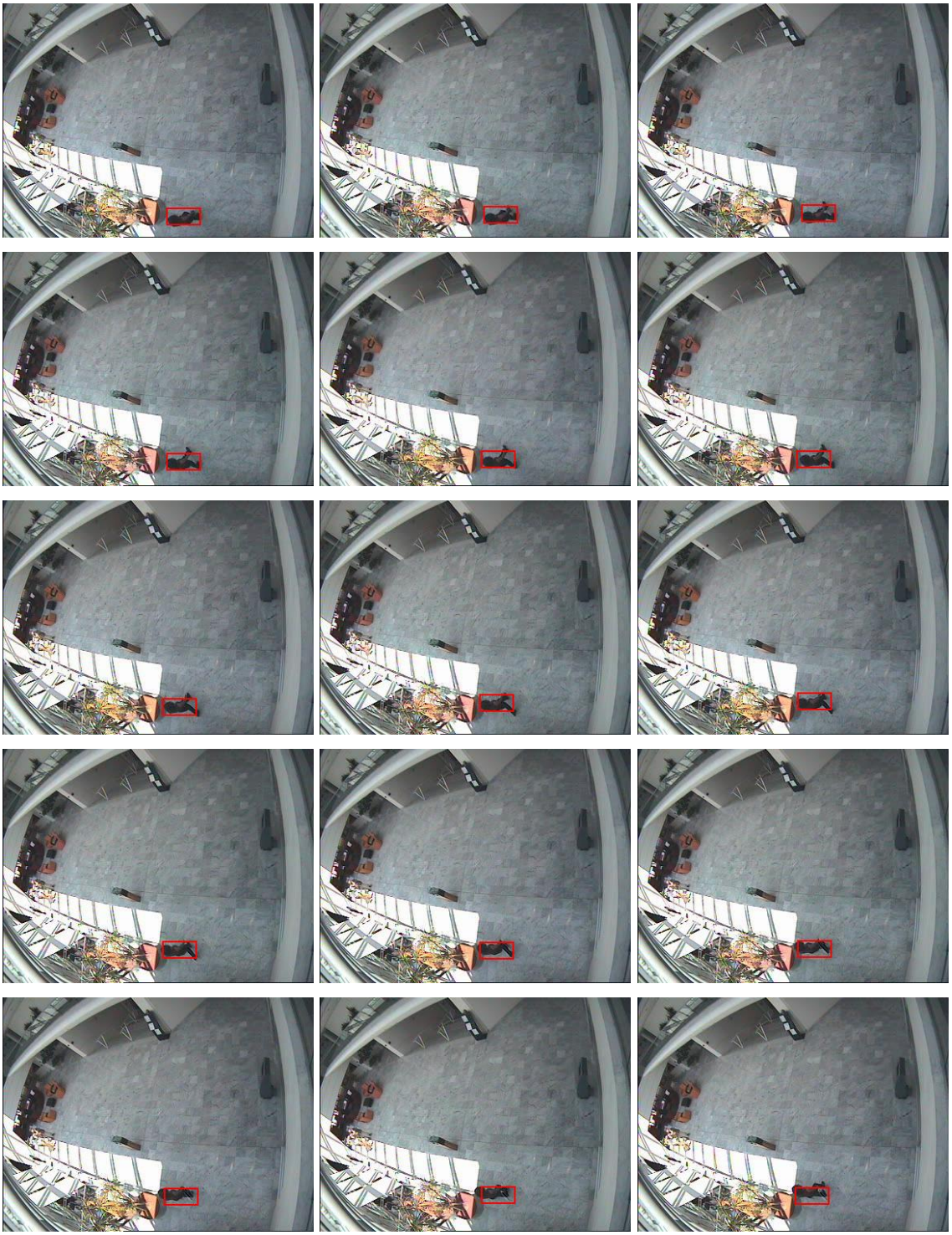


Figure C.26: Mean shift object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence CAVIAR.

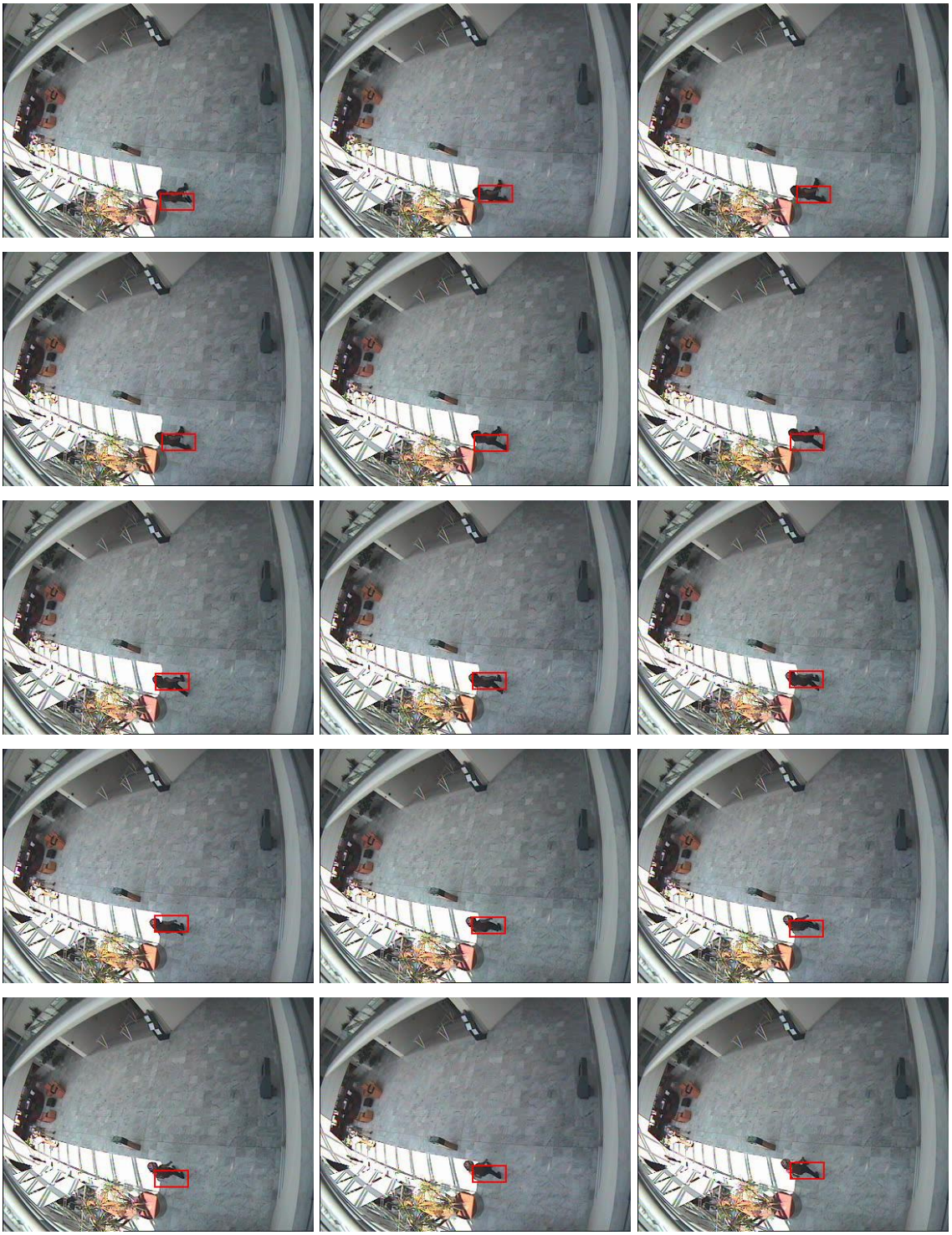


Figure C.27: Mean shift object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence CAVIAR.

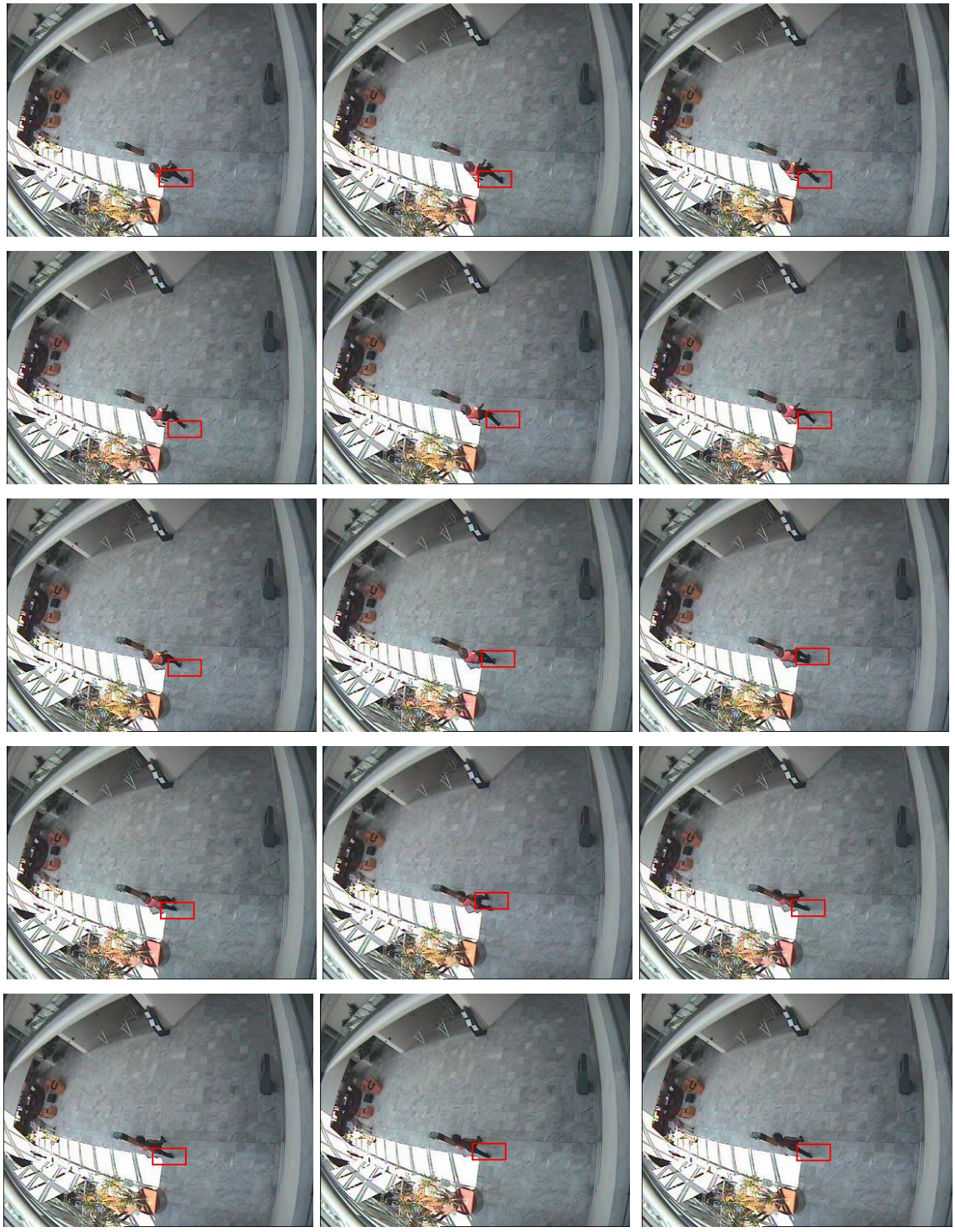


Figure C.28: Mean shift object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence CAVIAR.

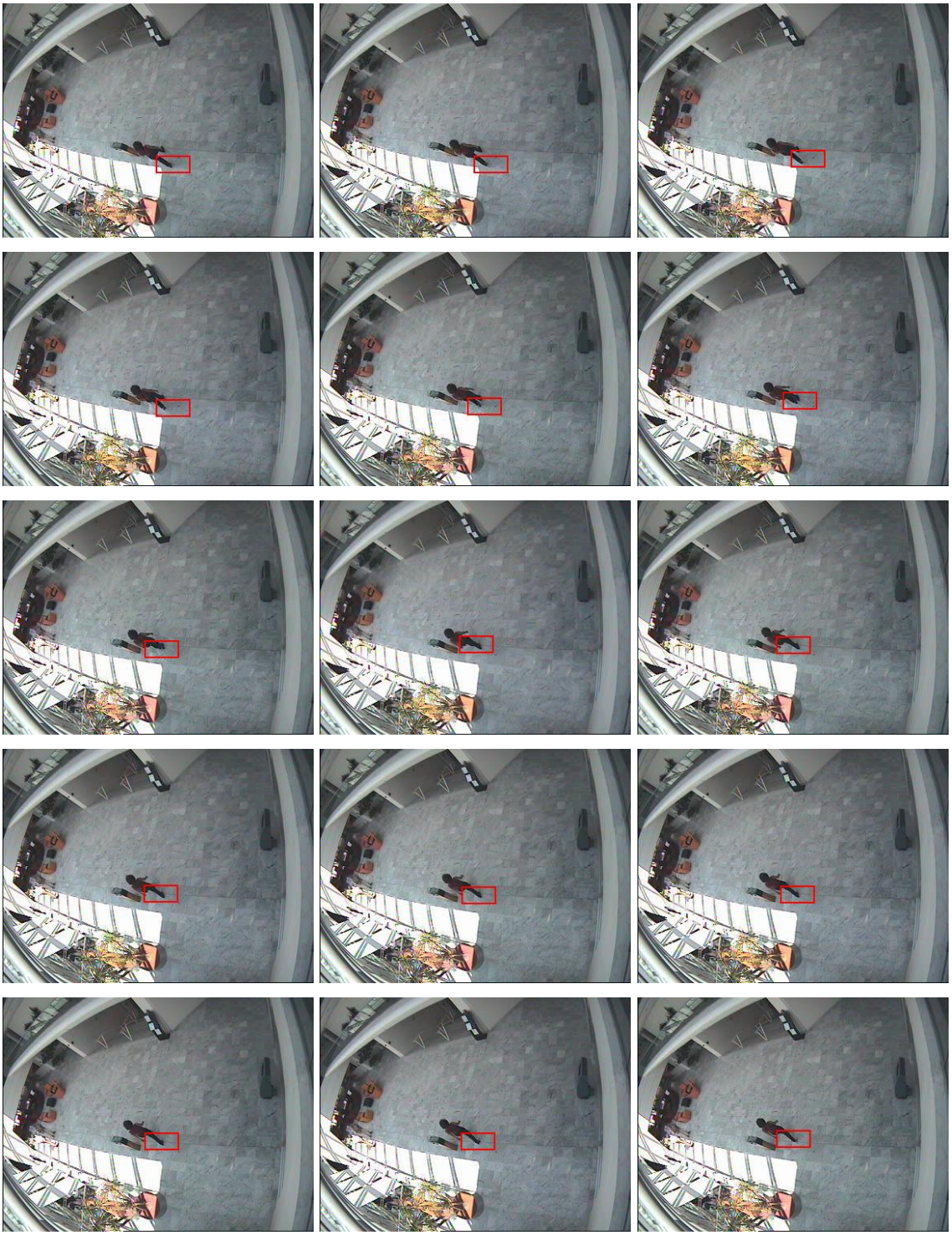


Figure C.29: Mean shift object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence CAVIAR.

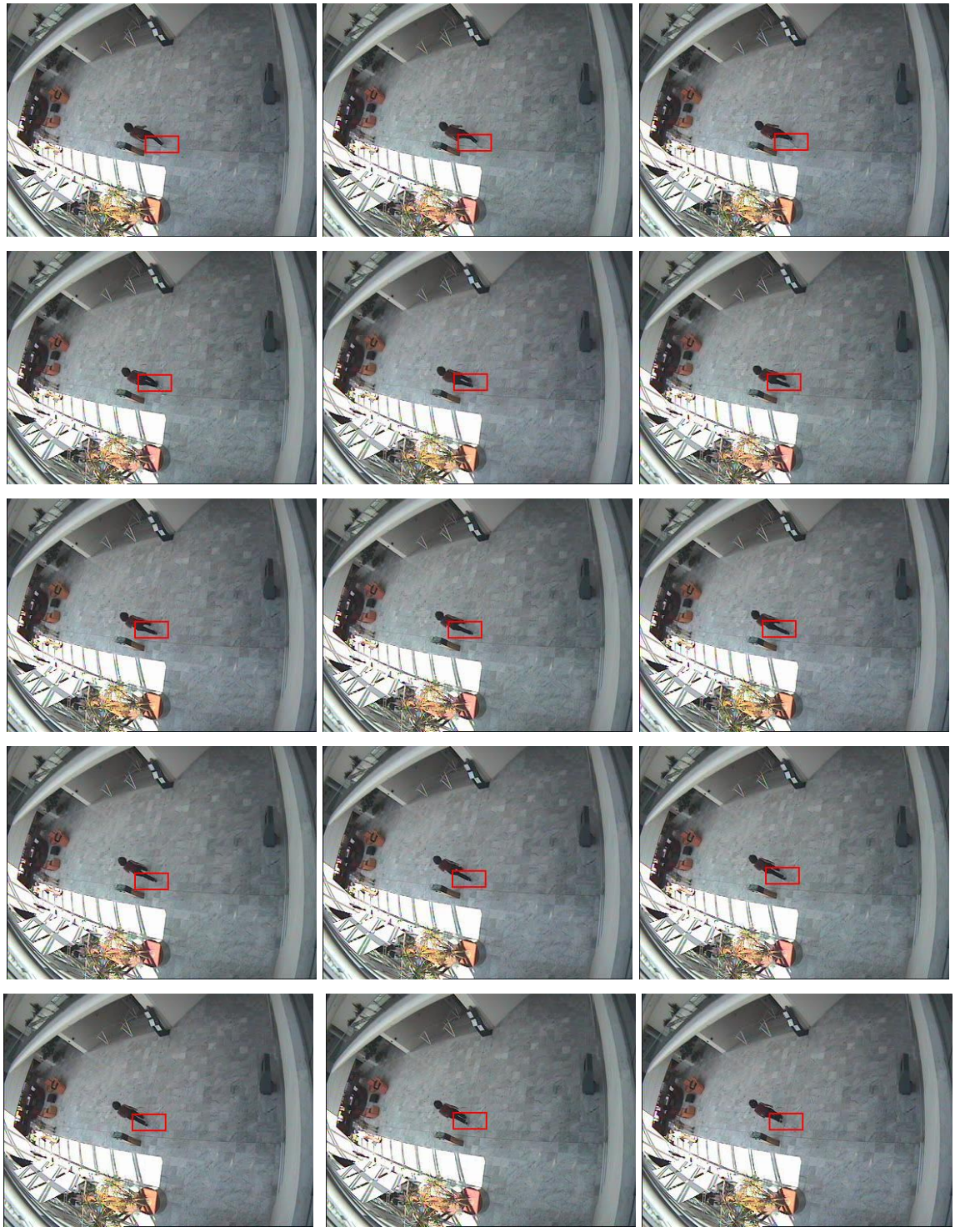


Figure C.30: Mean shift object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence CAVIAR.

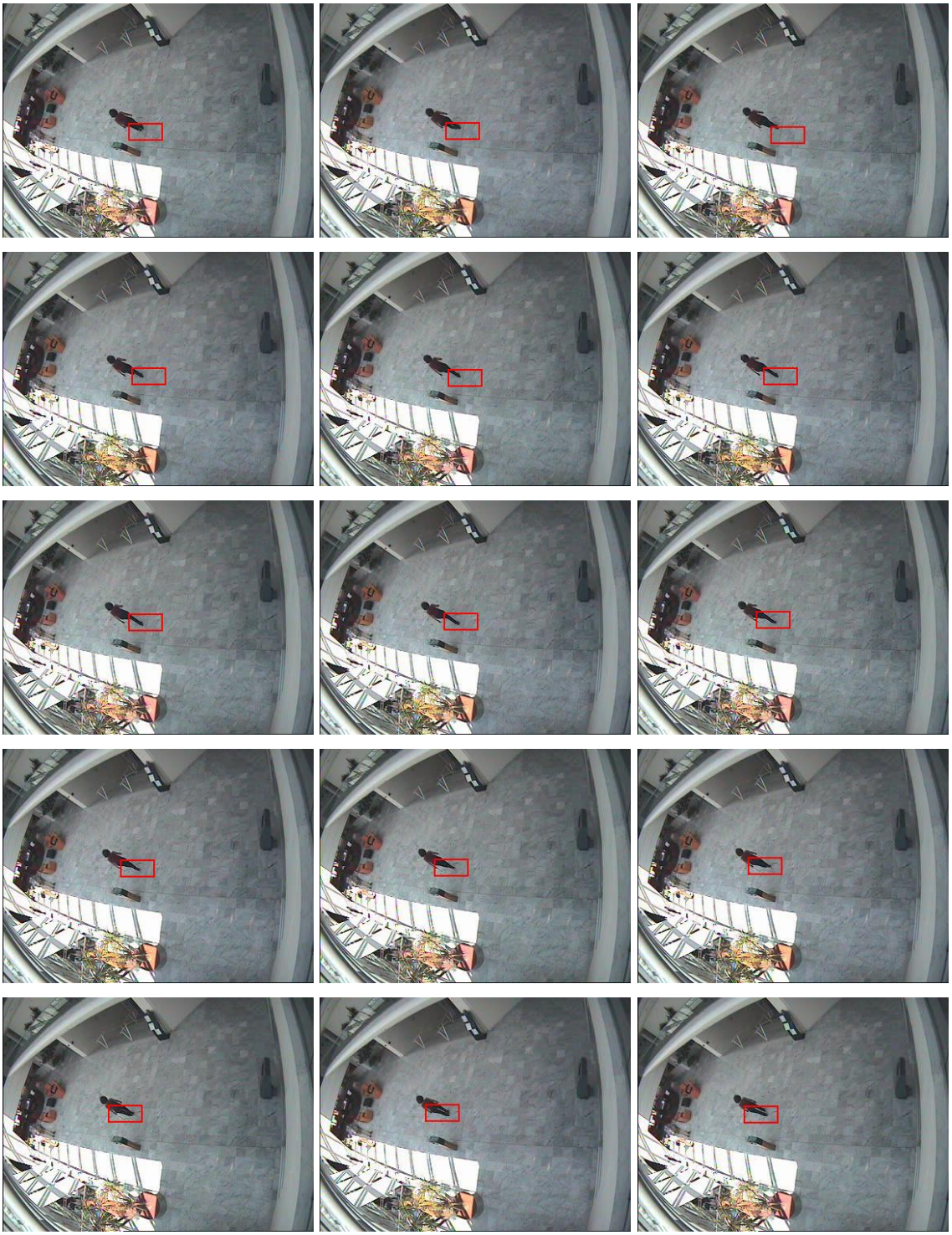


Figure C.31: Mean shift object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence CAVIAR.

Appendix D:
MATLAB CAMShift Object Tracking Result Frame by Frame

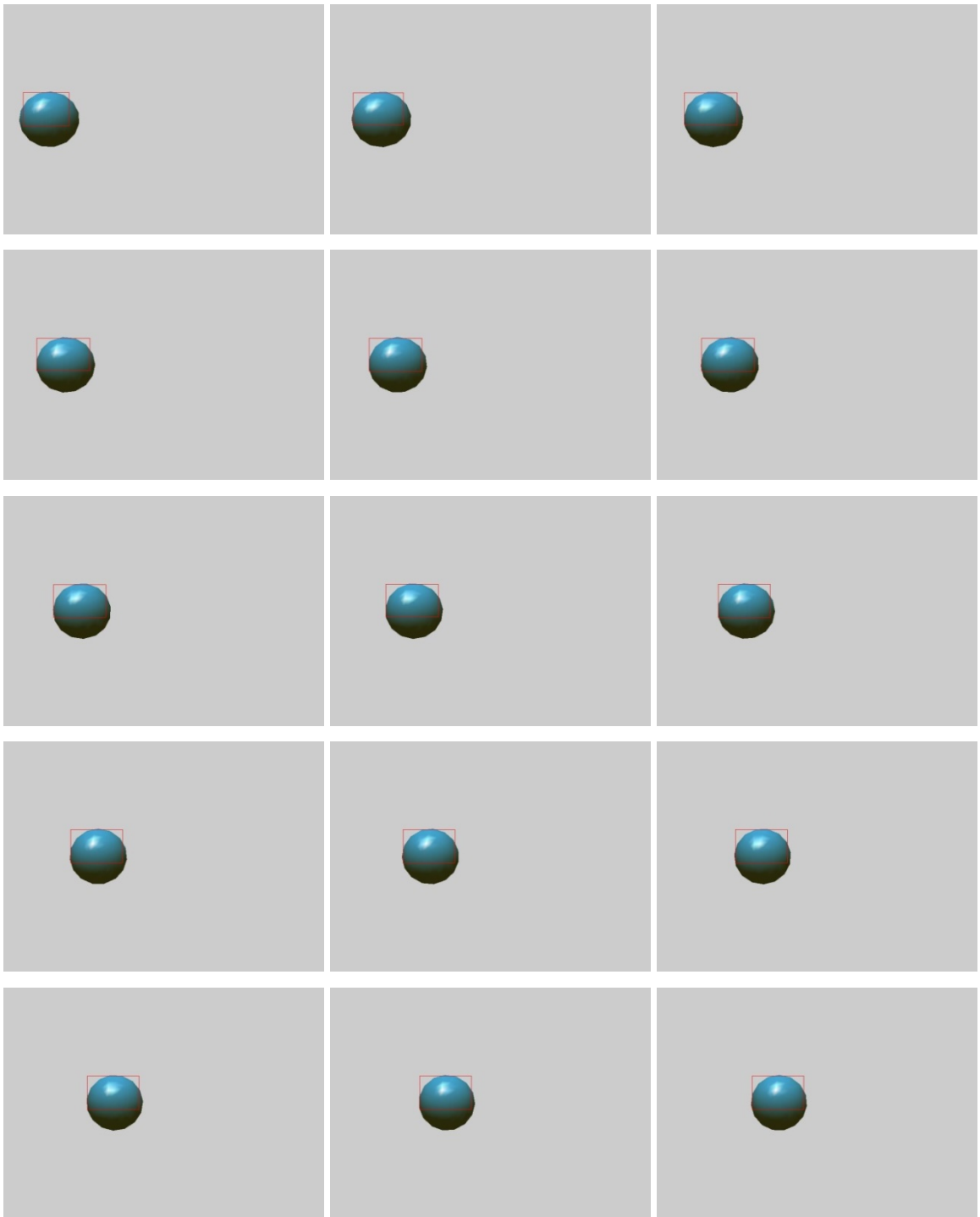


Figure D.1: CAMShift object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A1.

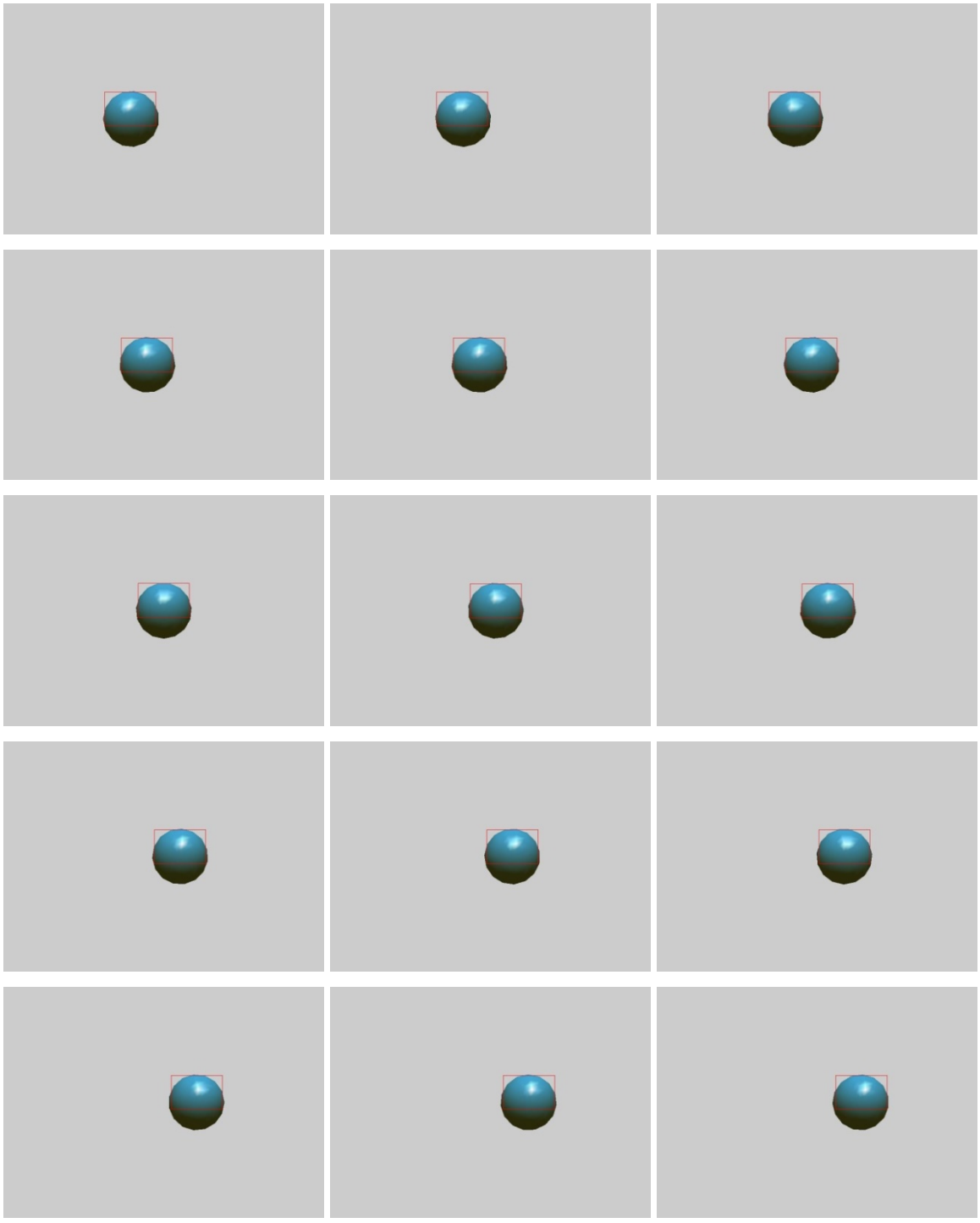


Figure D.2: CAMShift object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A1.

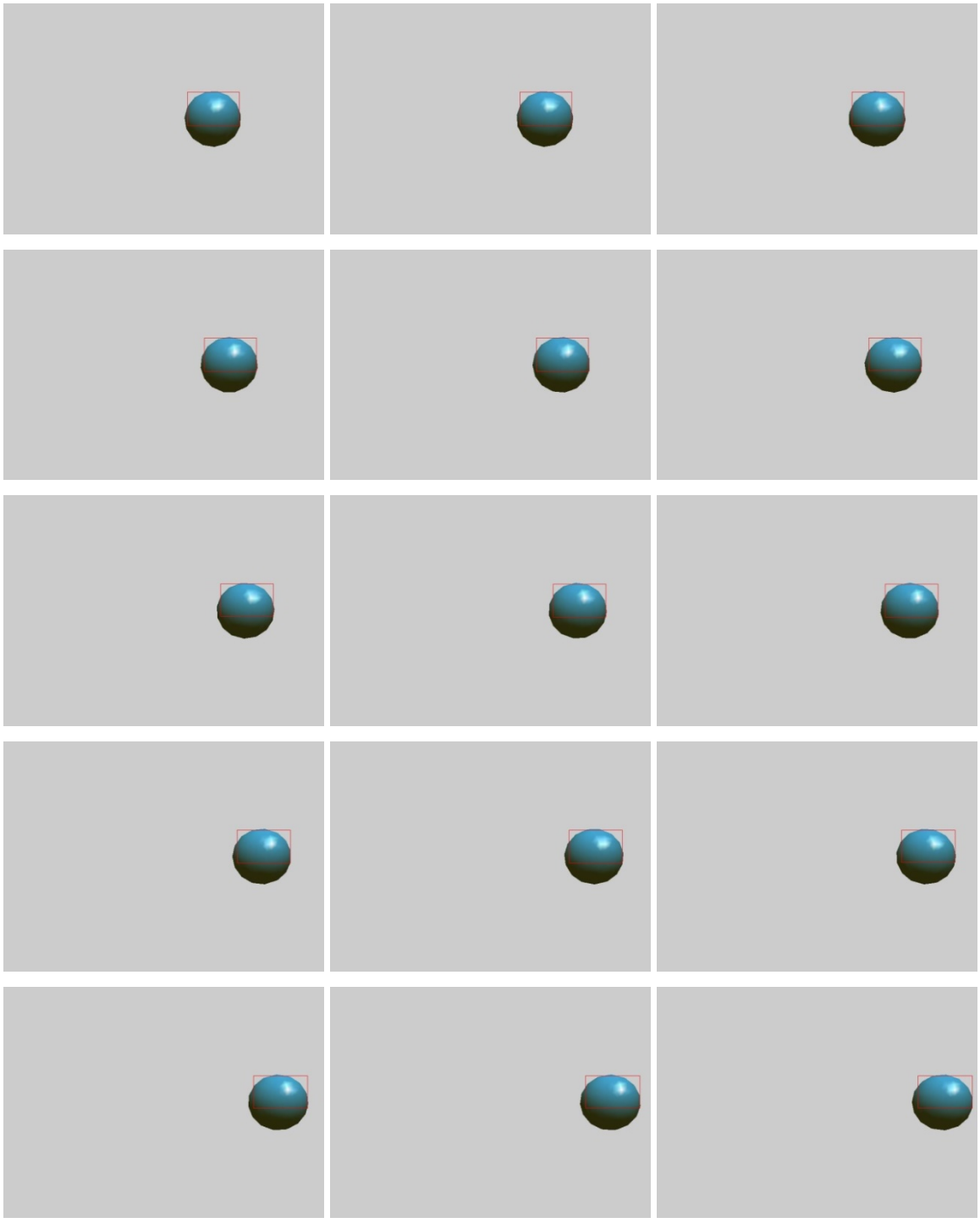


Figure D.3: CAMShift object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A1.

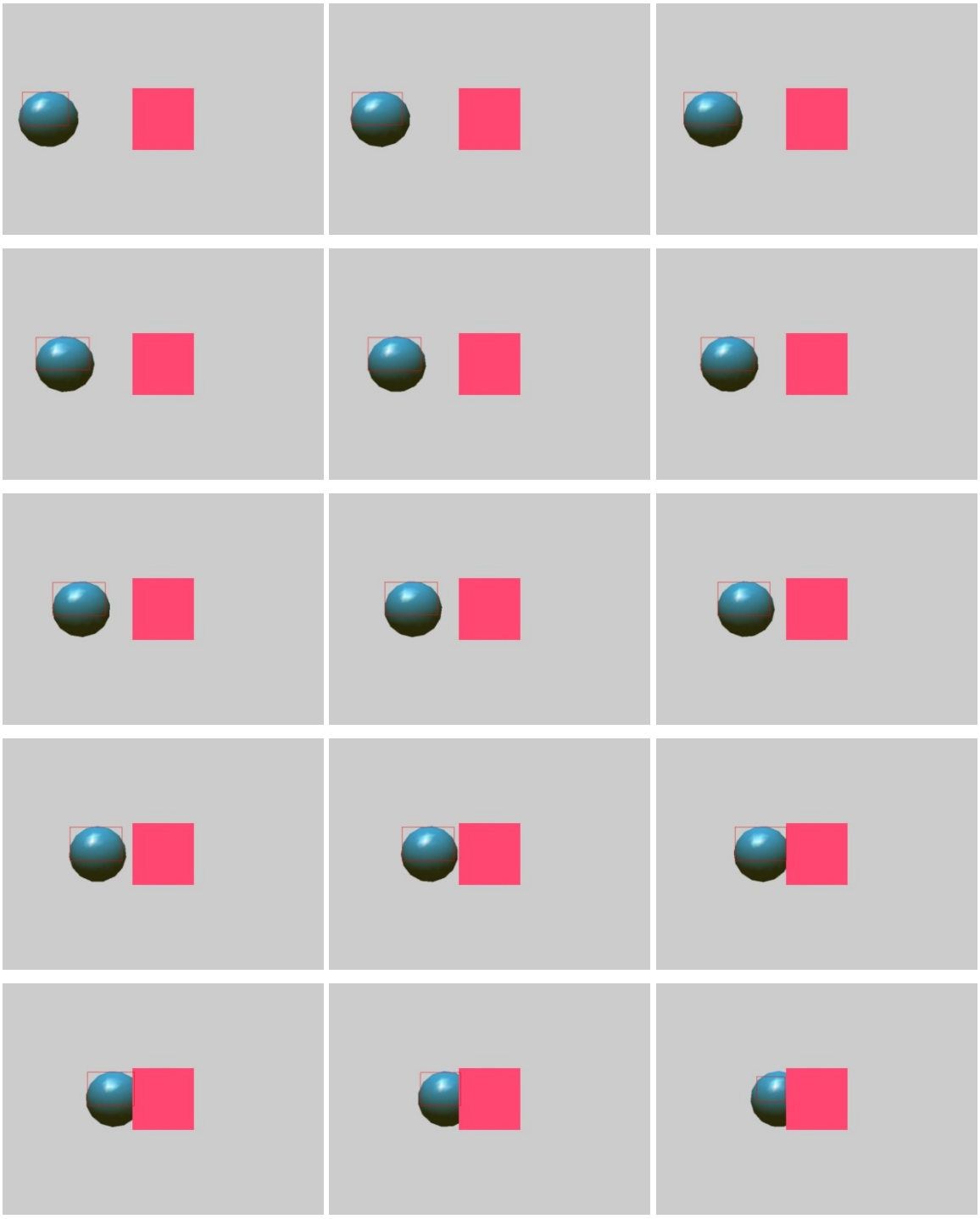


Figure D.4: CAMShift object tracking result when tracking a moving ball in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence A3.

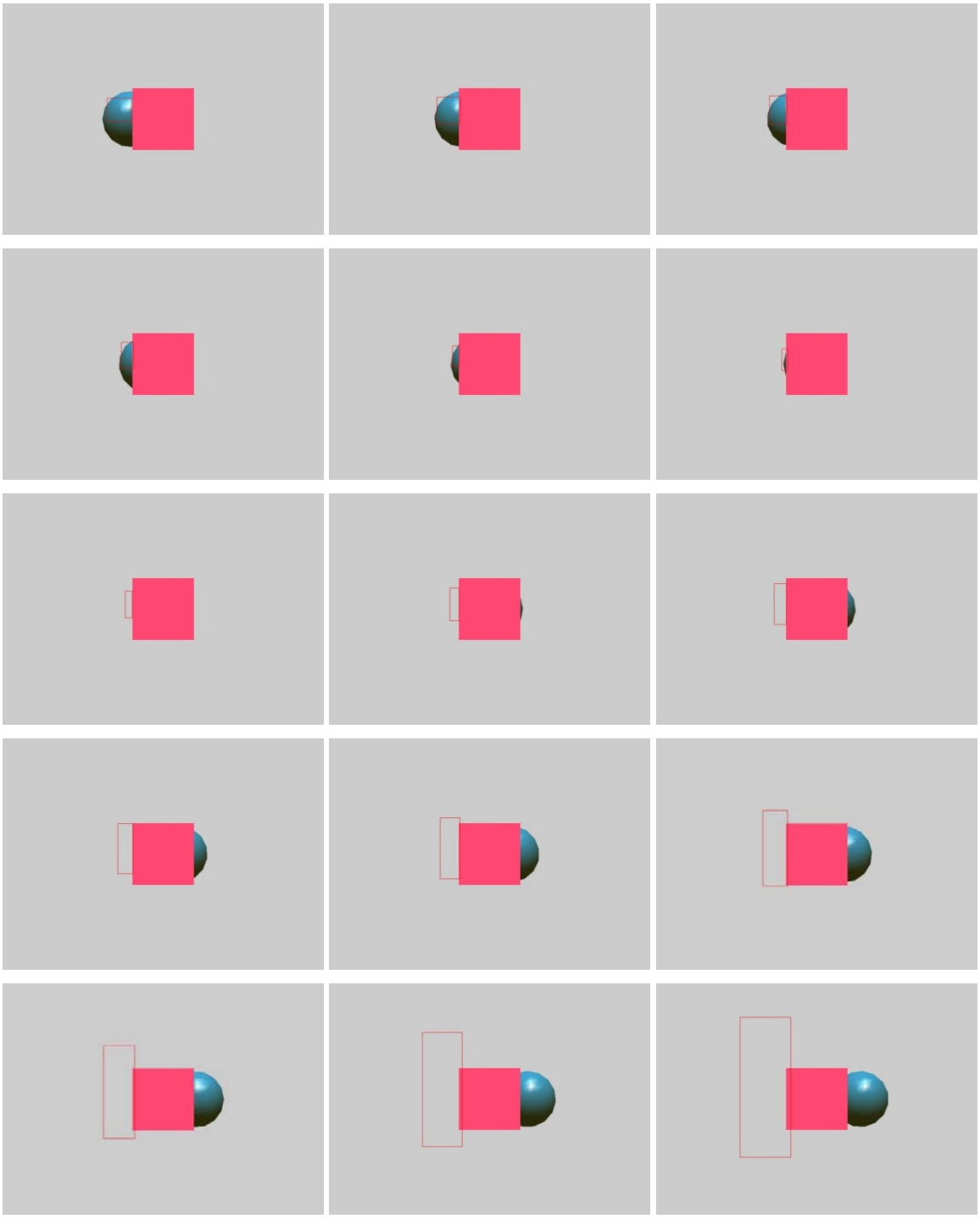


Figure D.5: CAMShift object tracking result when tracking a moving ball in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence A3.

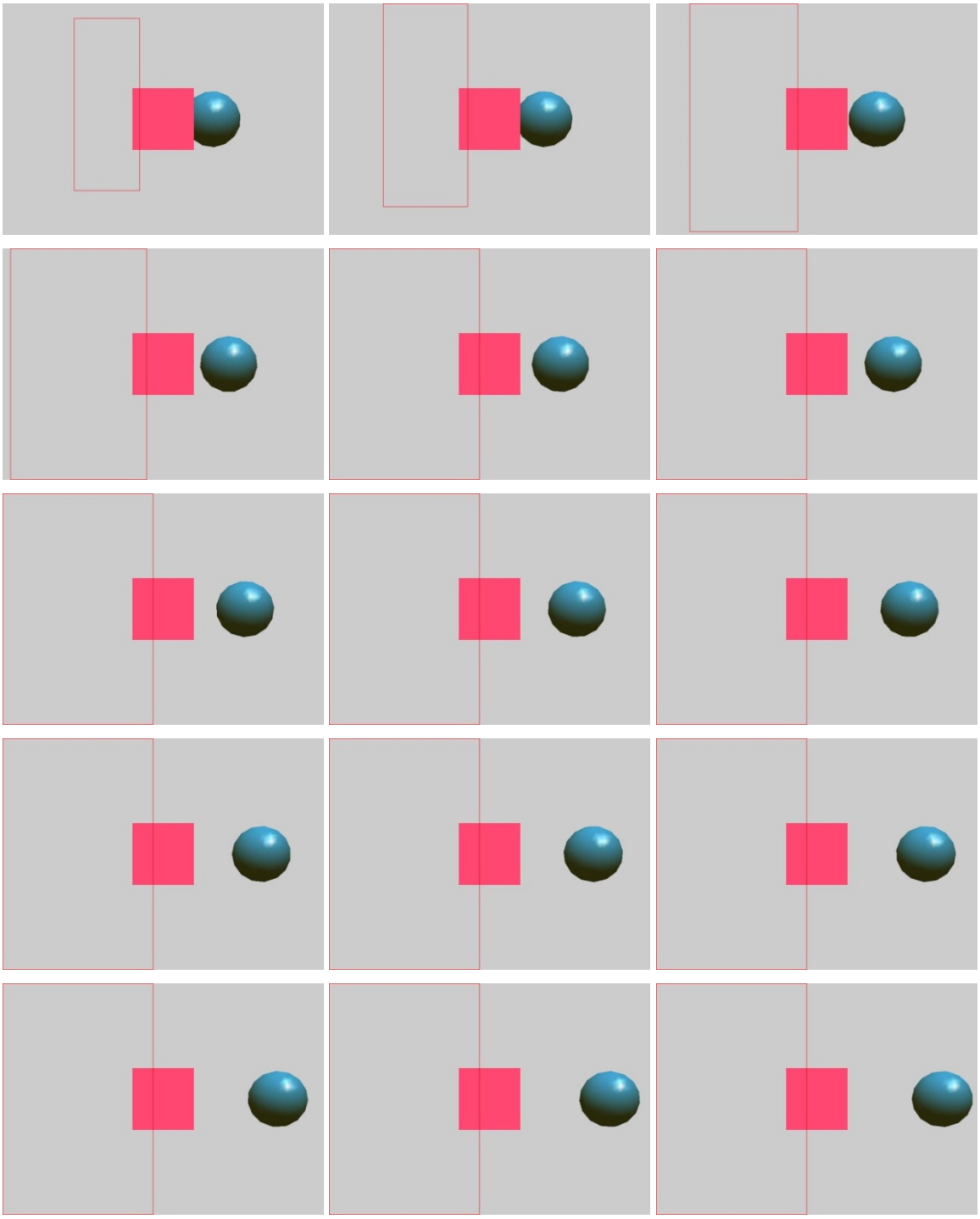


Figure D.6: CAMShift object tracking result when tracking a moving ball in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence A3.

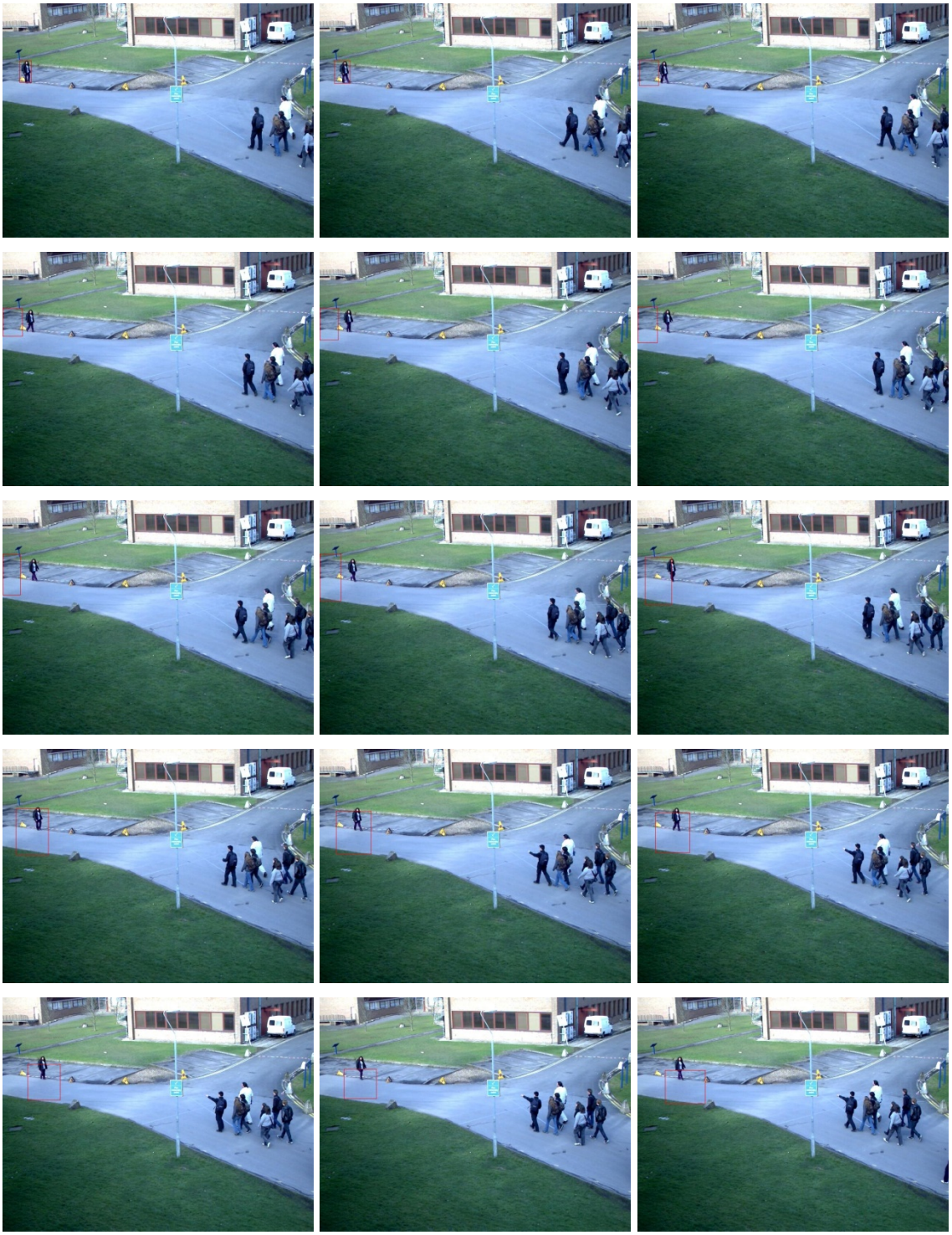


Figure D.7: CAMShift object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence PETS 2009.

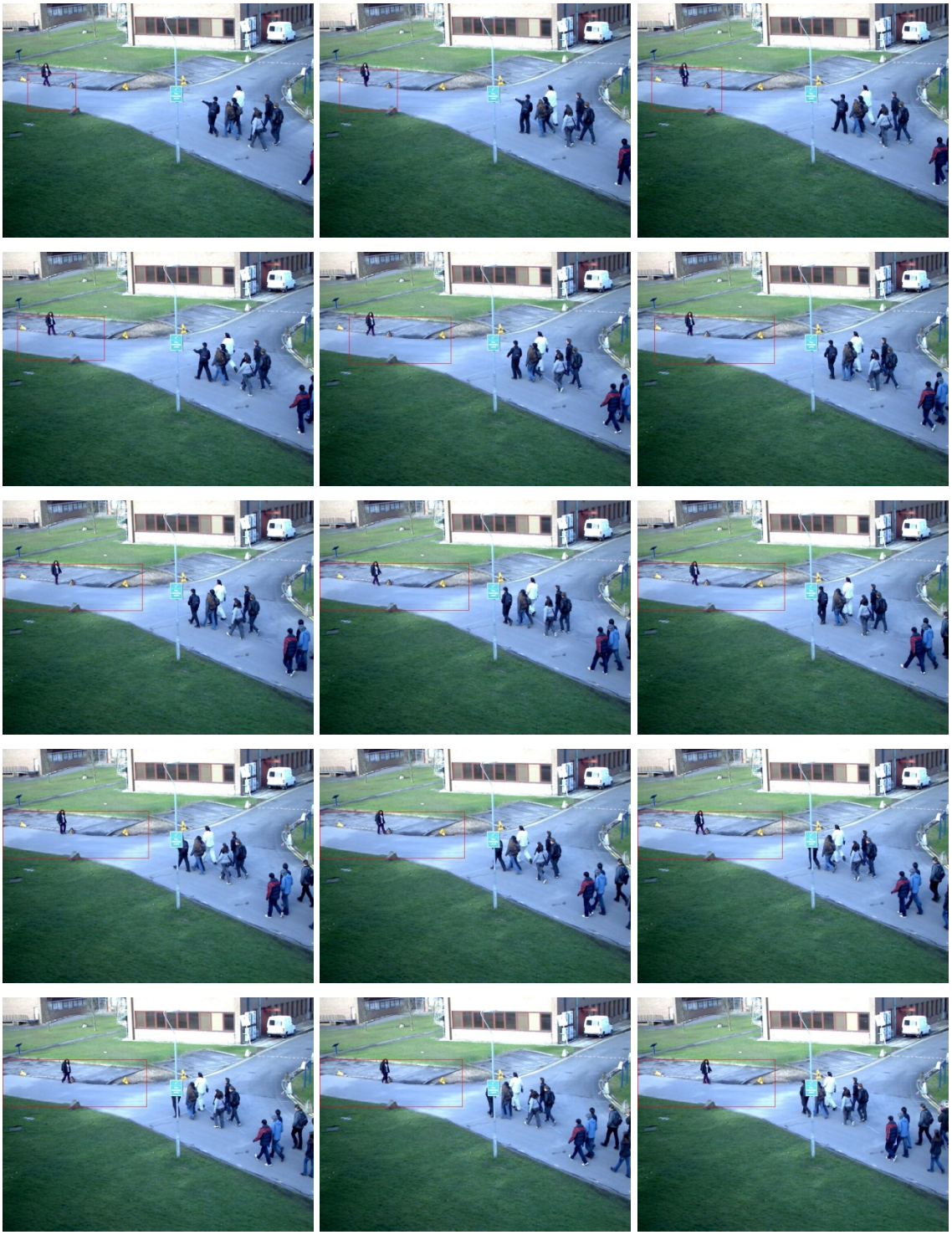


Figure D.8: CAMShift object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure D.9: CAMShift object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence PETS 2009.

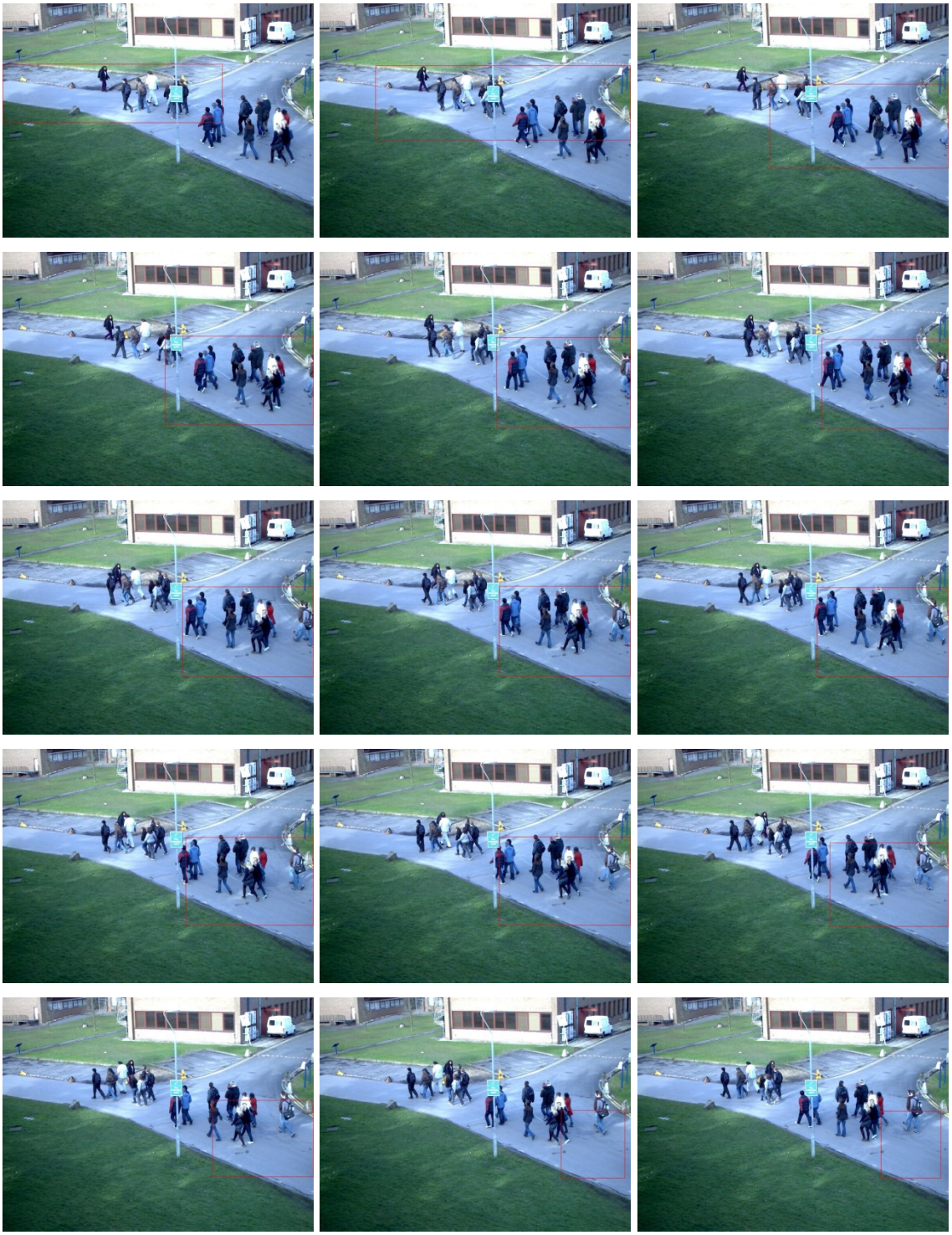


Figure D.10: CAMShift object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure D.11: CAMShift object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure D.12: CAMShift object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence PETS 2009.

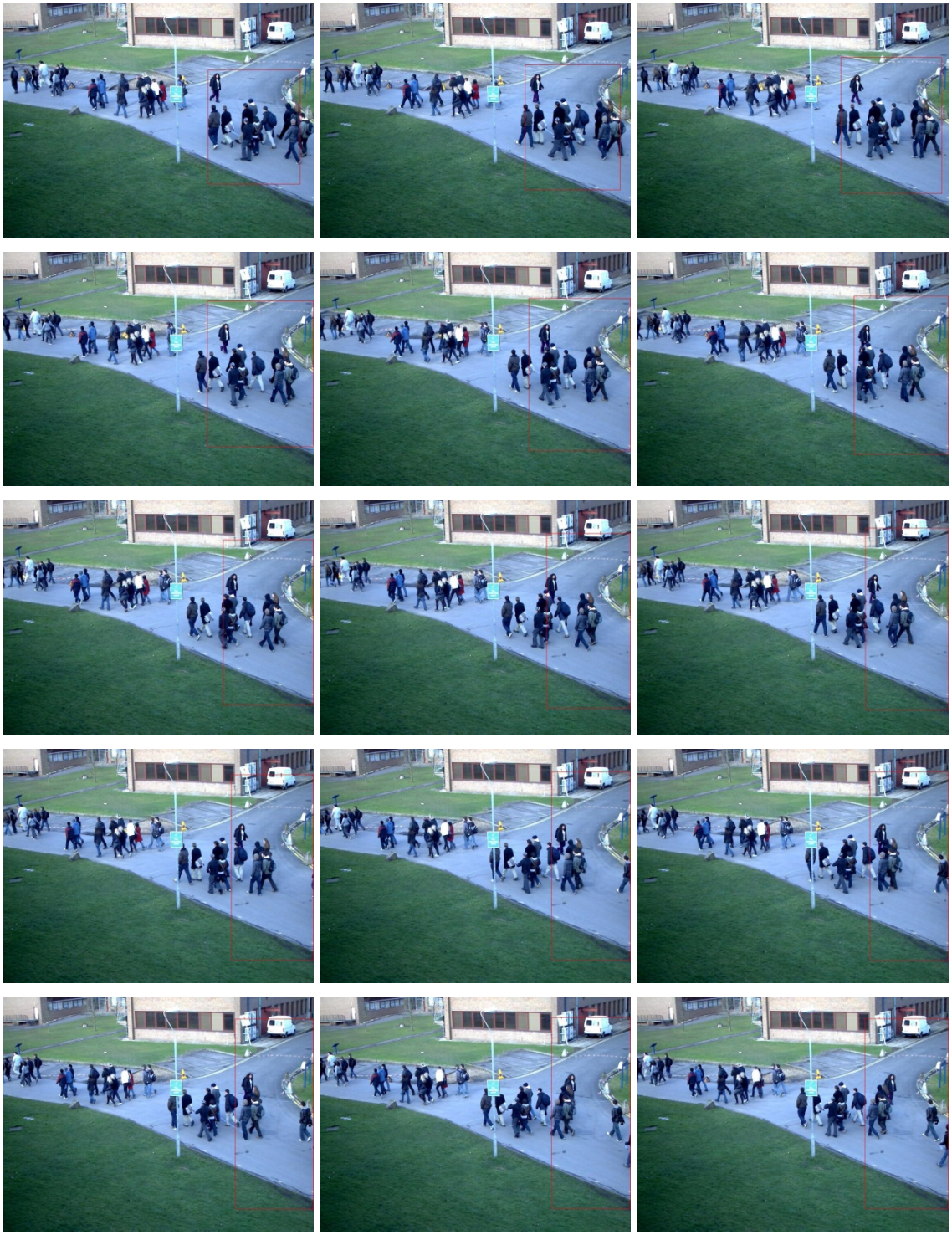


Figure D.13: CAMShift object tracking result when tracking a pedestrian in Frame 91 to Frame 105 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure D.14: CAMShift object tracking result when tracking a pedestrian in Frame 106 to Frame 120 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure D.15: CAMShift object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence CAVIAR.

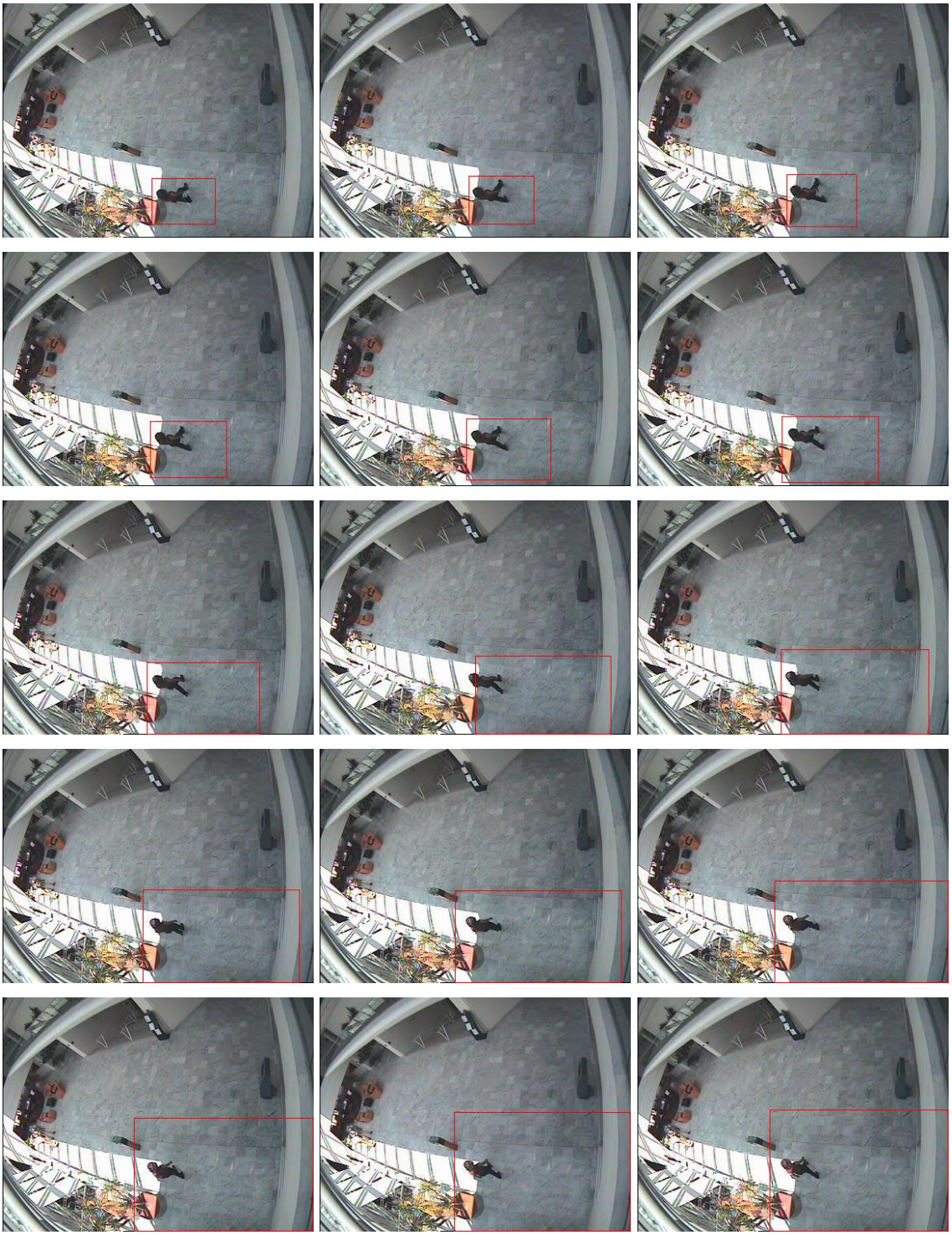


Figure D.16: CAMShift object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence CAVIAR.

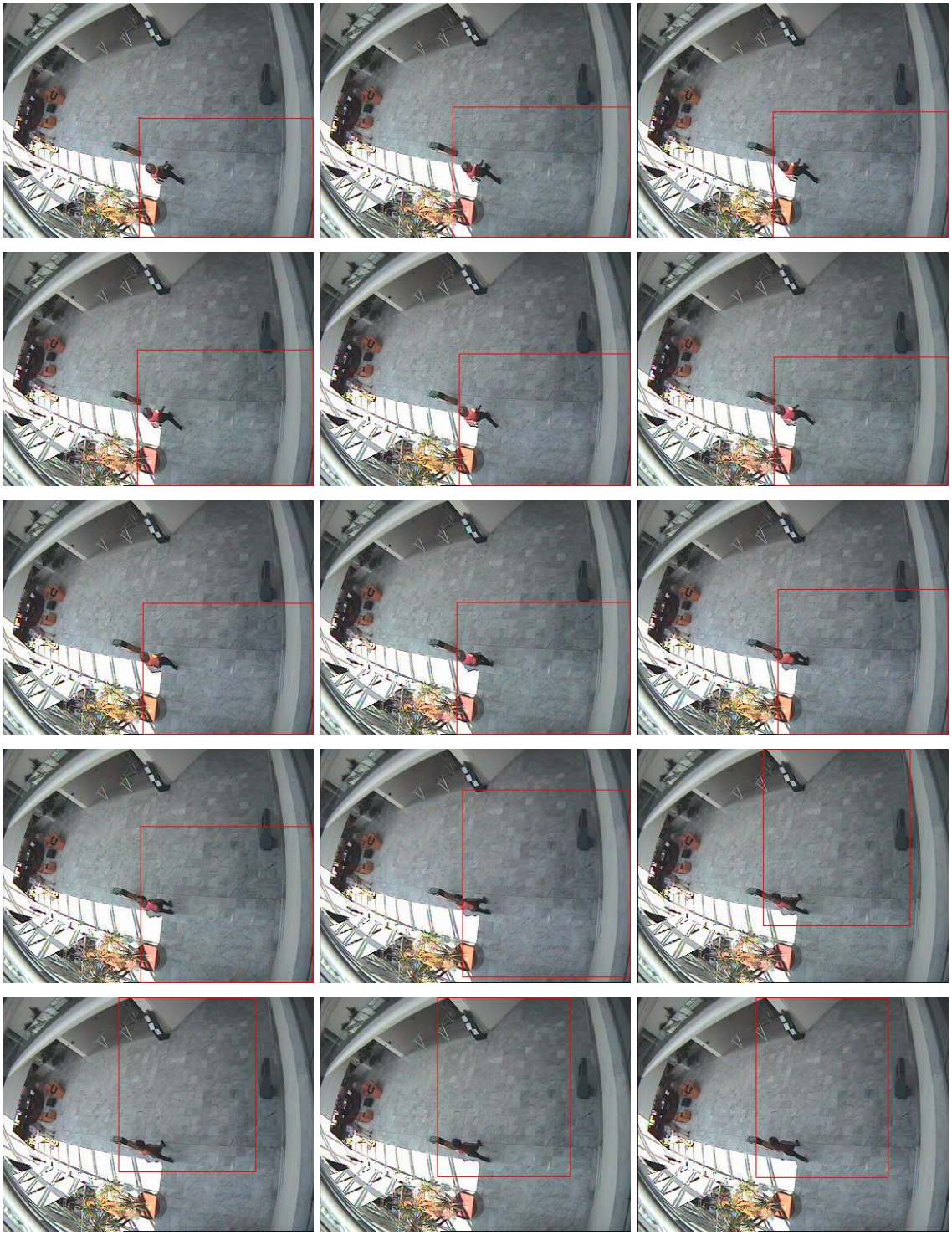


Figure D.17: CAMShift object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence CAVIAR.

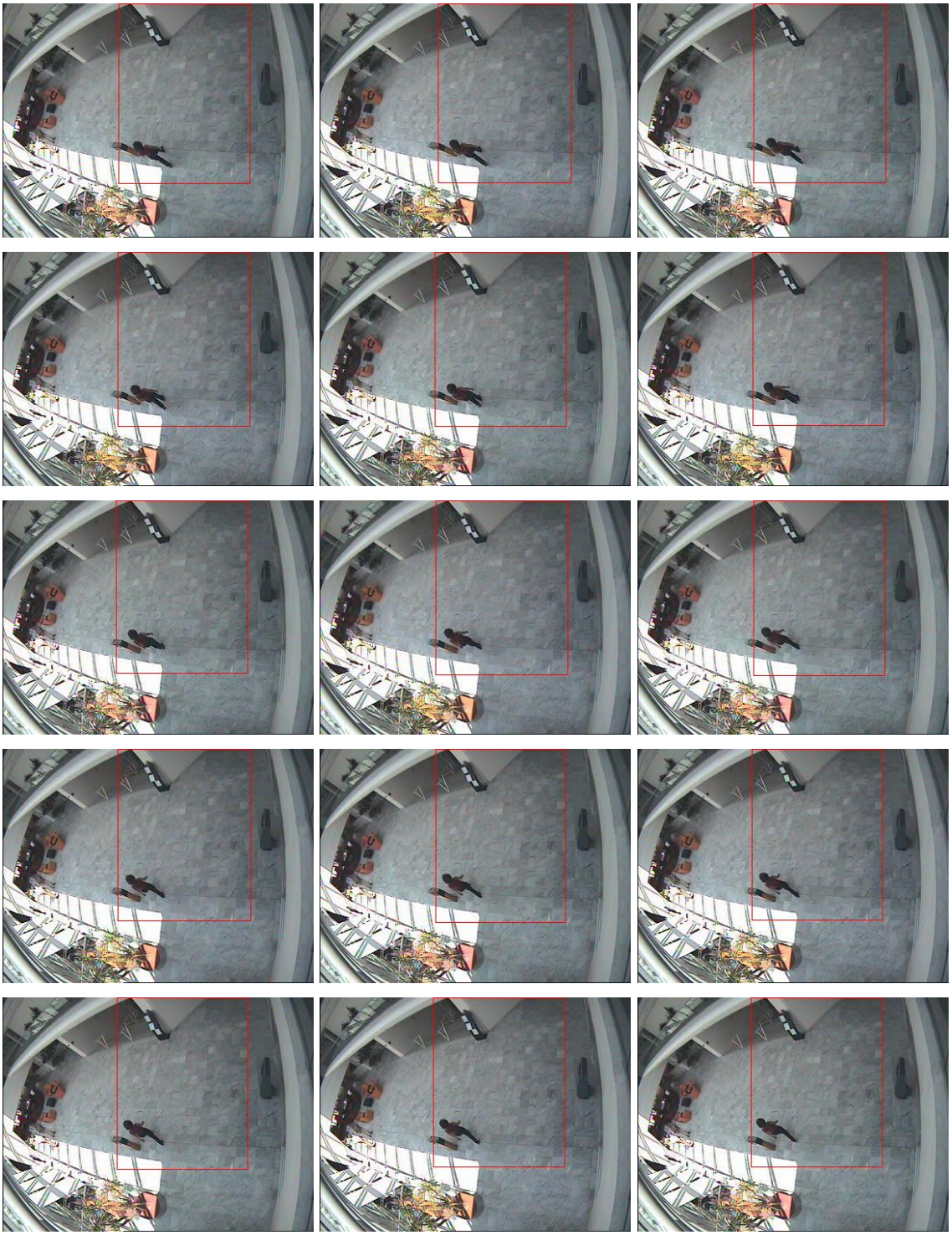


Figure D.18: CAMShift object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence CAVIAR.

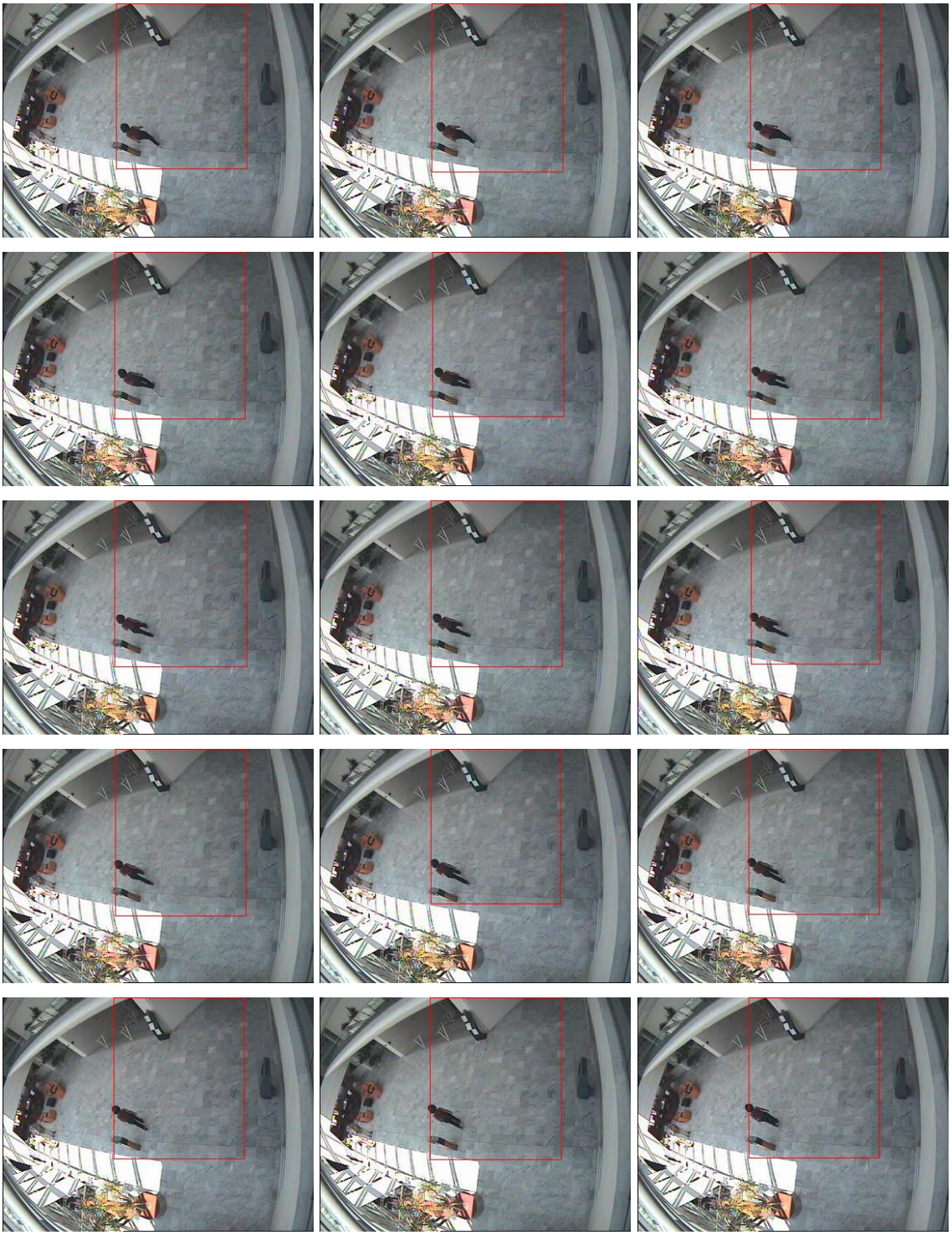


Figure D.19: CAMShift object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence CAVIAR.

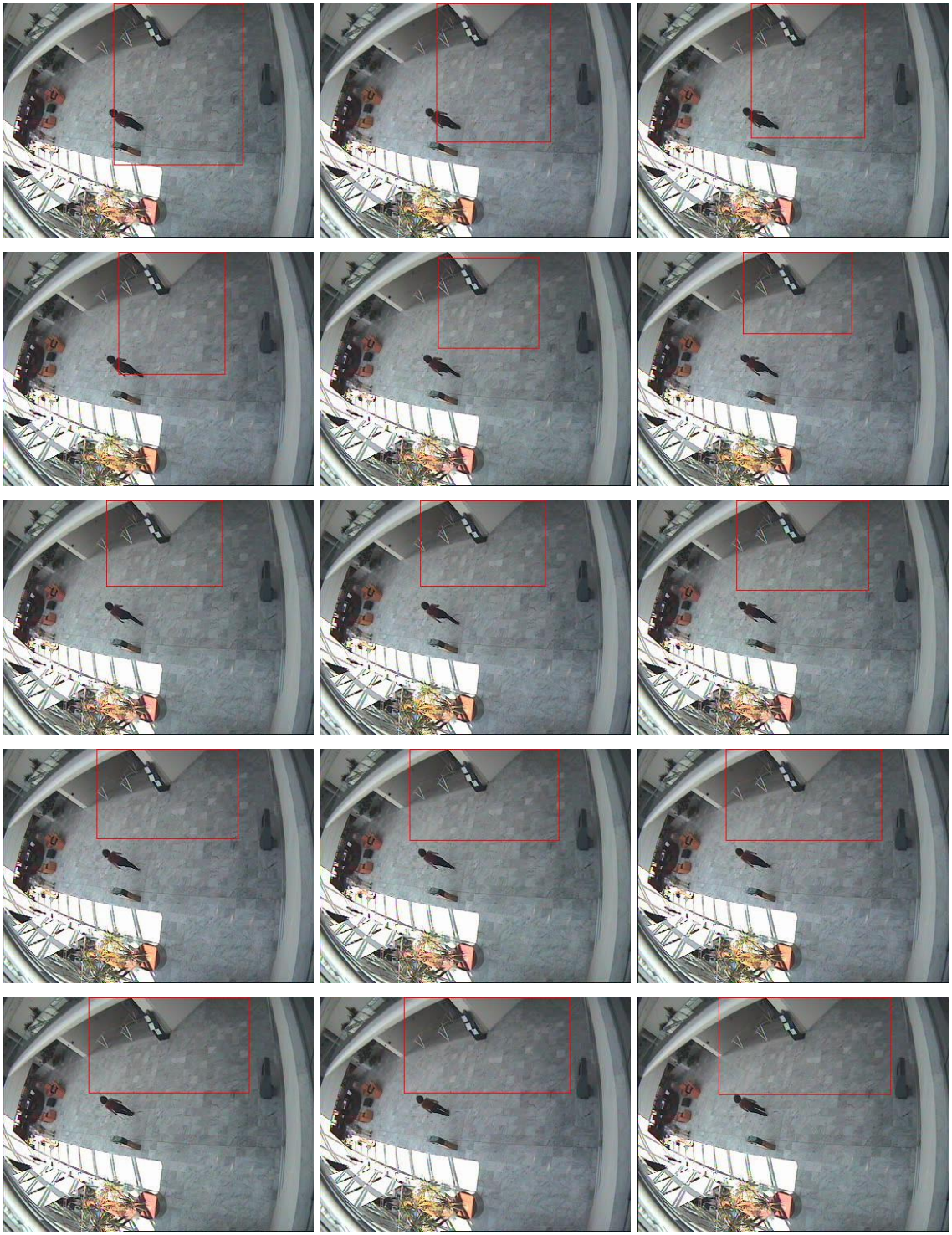


Figure D.20: CAMShift object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence CAVIAR.

Appendix E:

Multi CAMShift Agent Object Tracking Result Frame by Frame

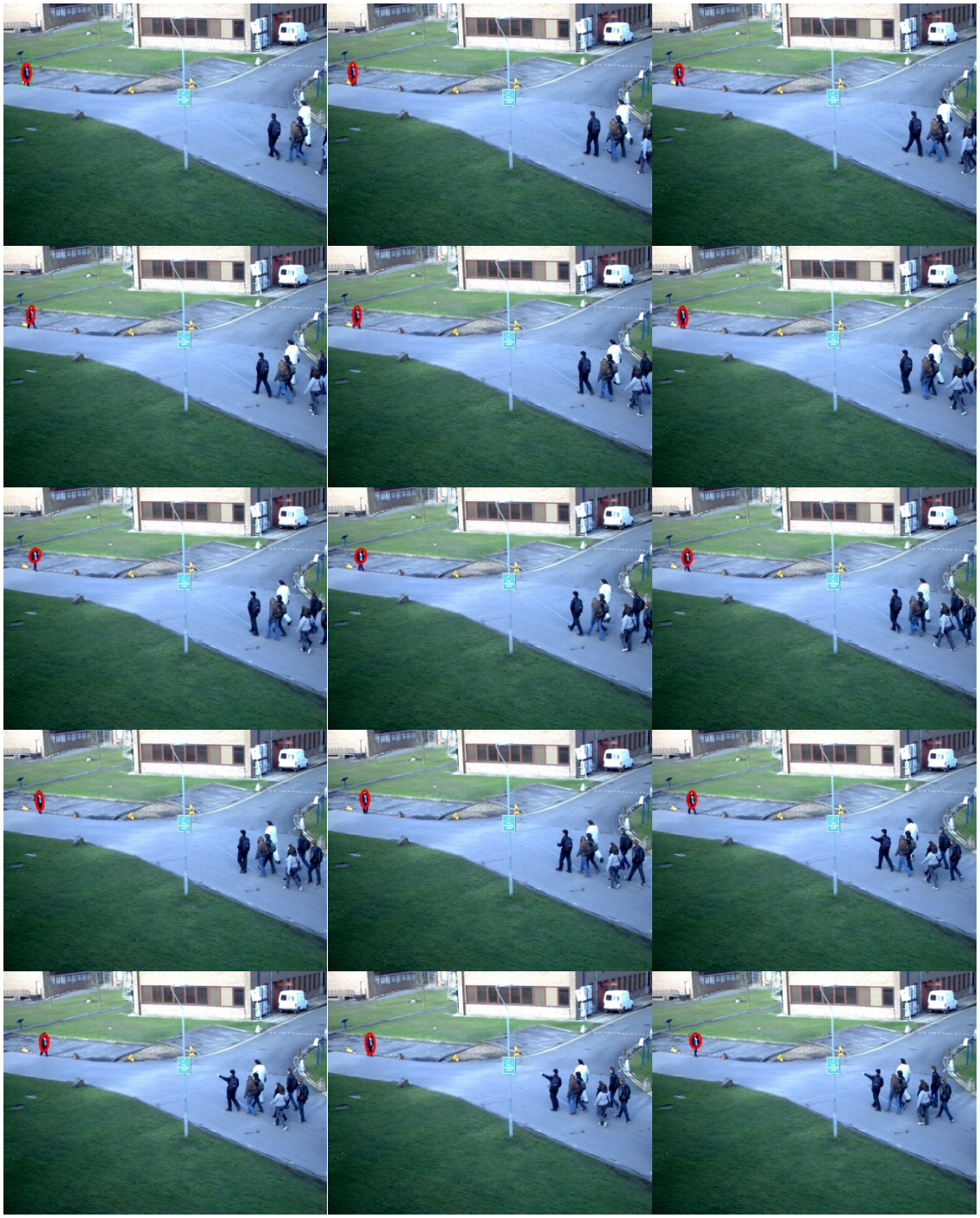


Figure E.1: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 01 to Frame 15 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure E.2: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure E.3: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence PETS 2009.

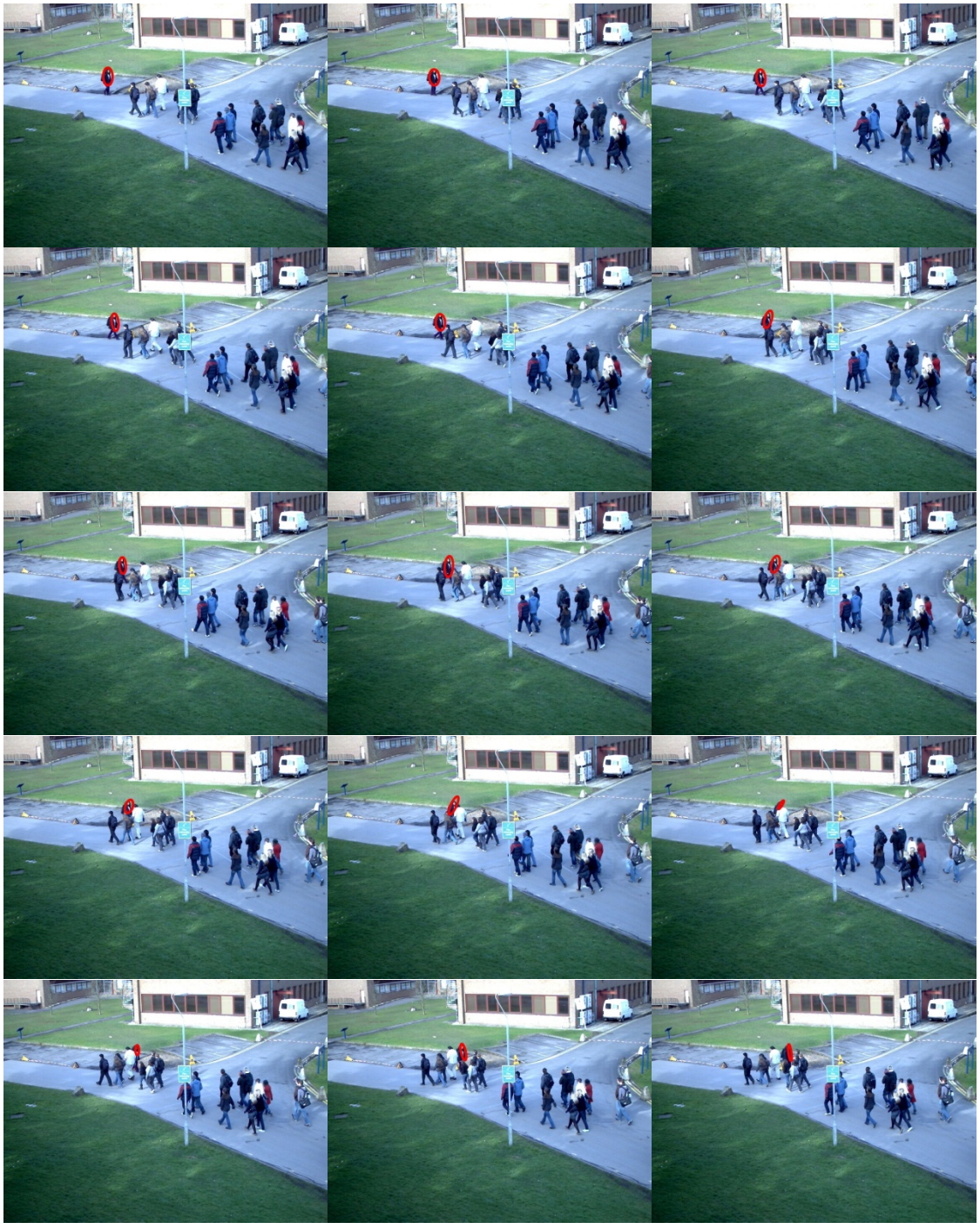


Figure E.4: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure E.5: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure E.6: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure E.7: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 90 to Frame 105 (From top left to top right and then to bottom) in Video sequence PETS 2009.



Figure E.8: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 106 to Frame 120 (From top left to top right and then to bottom) in Video sequence PETS 2009.

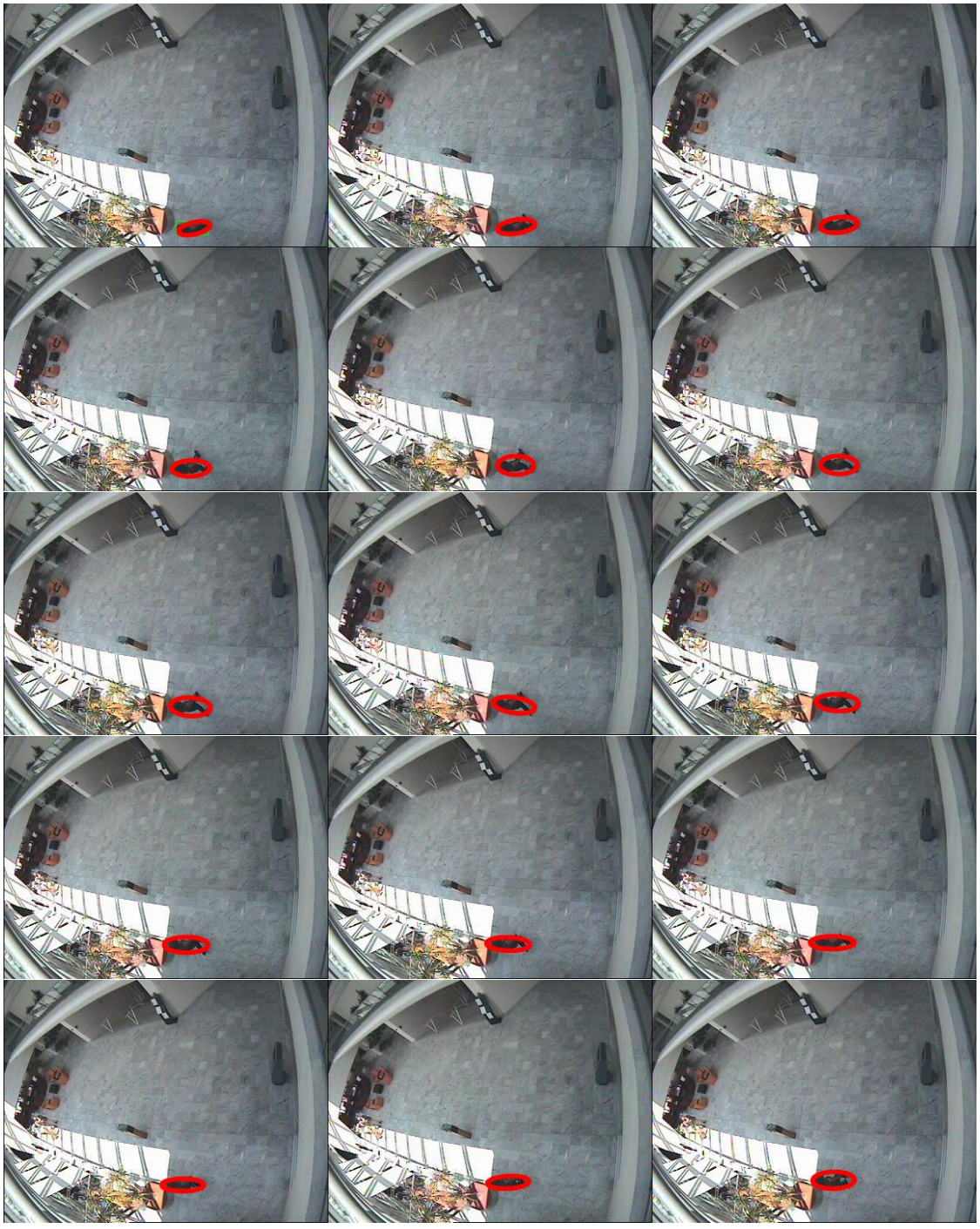


Figure E.9: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 1 to Frame 15 (From top left to top right and then to bottom) in Video sequence CAVIAR.

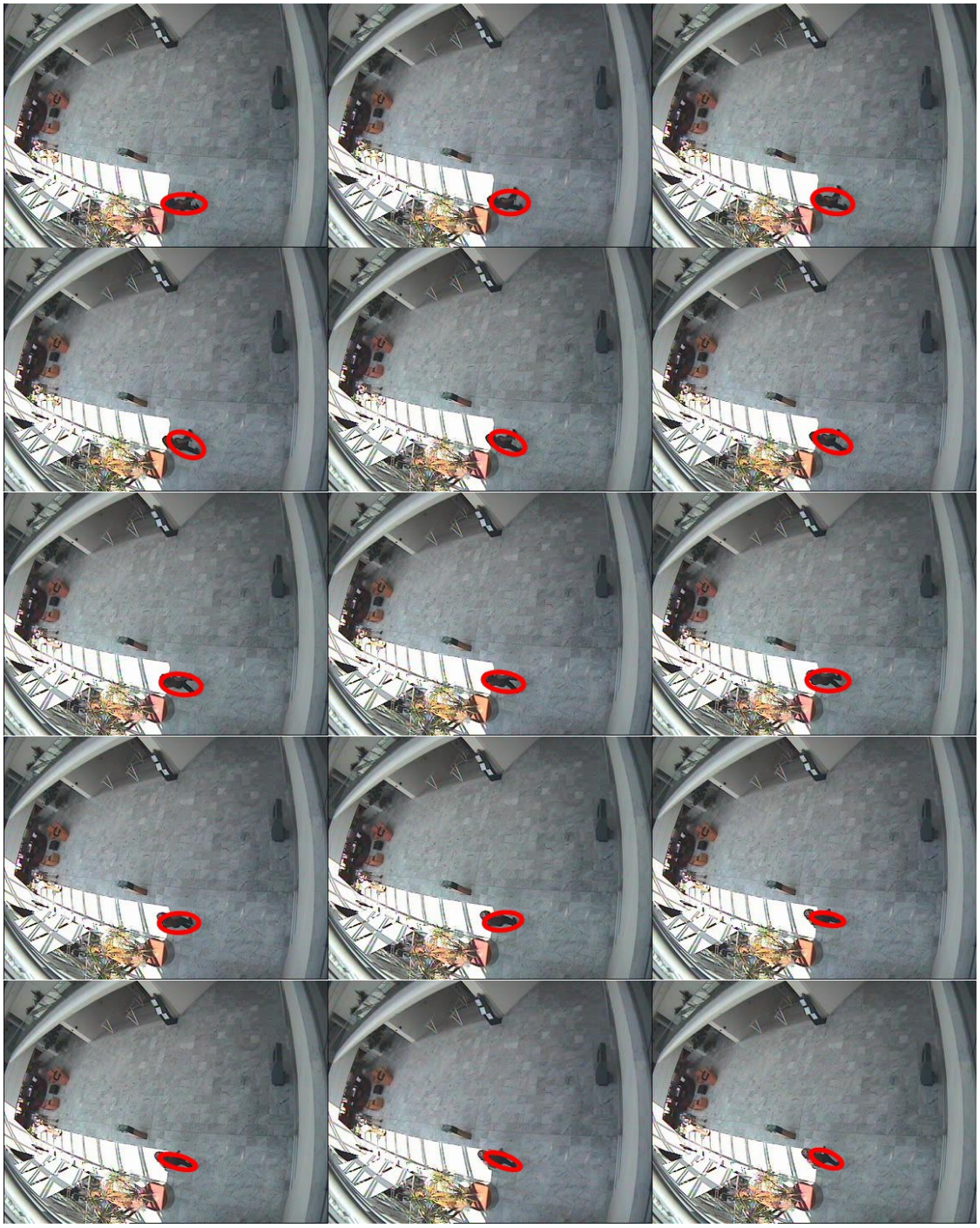


Figure E.10: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 16 to Frame 30 (From top left to top right and then to bottom) in Video sequence CAVIAR.

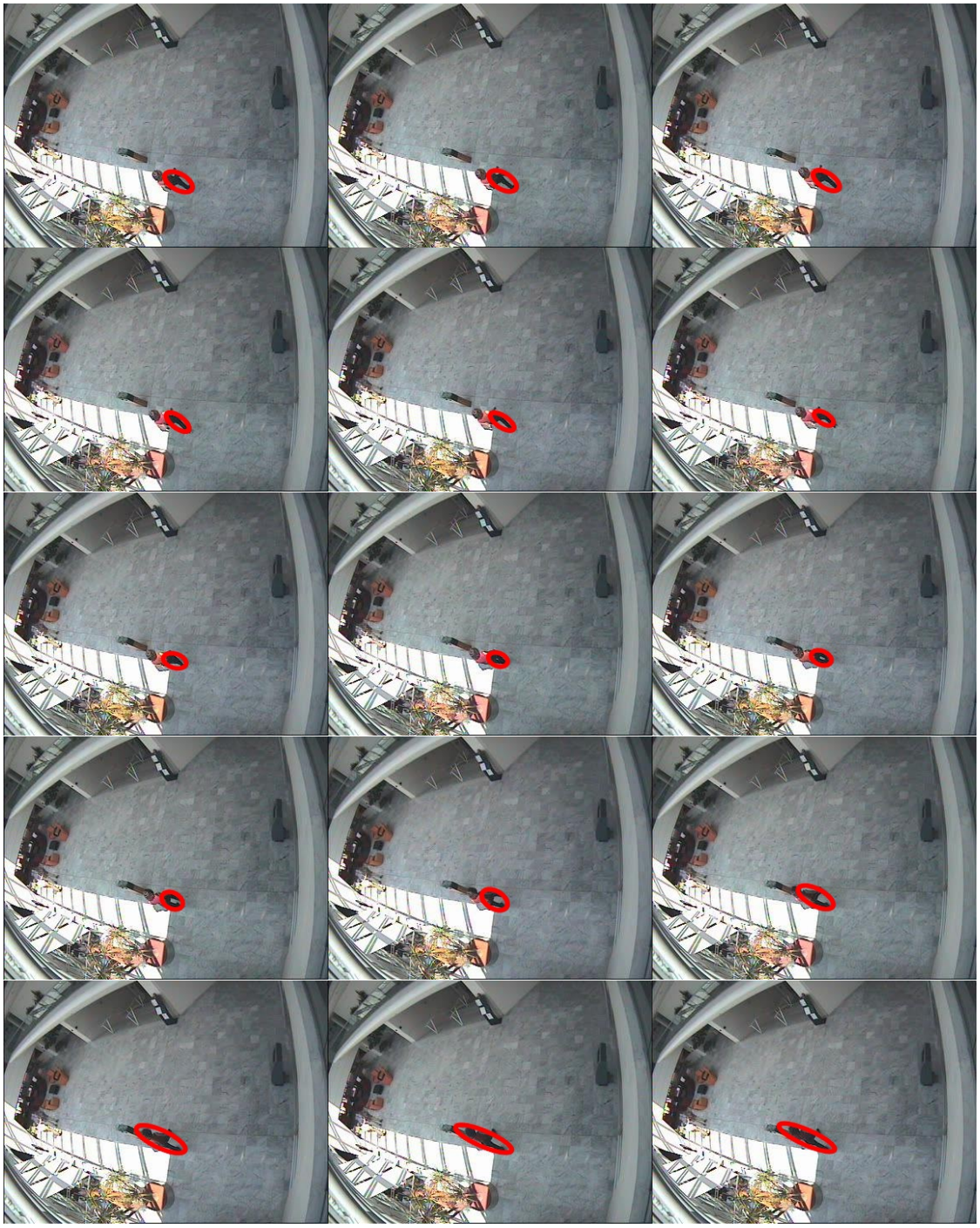


Figure E.11: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 31 to Frame 45 (From top left to top right and then to bottom) in Video sequence CAVIAR.

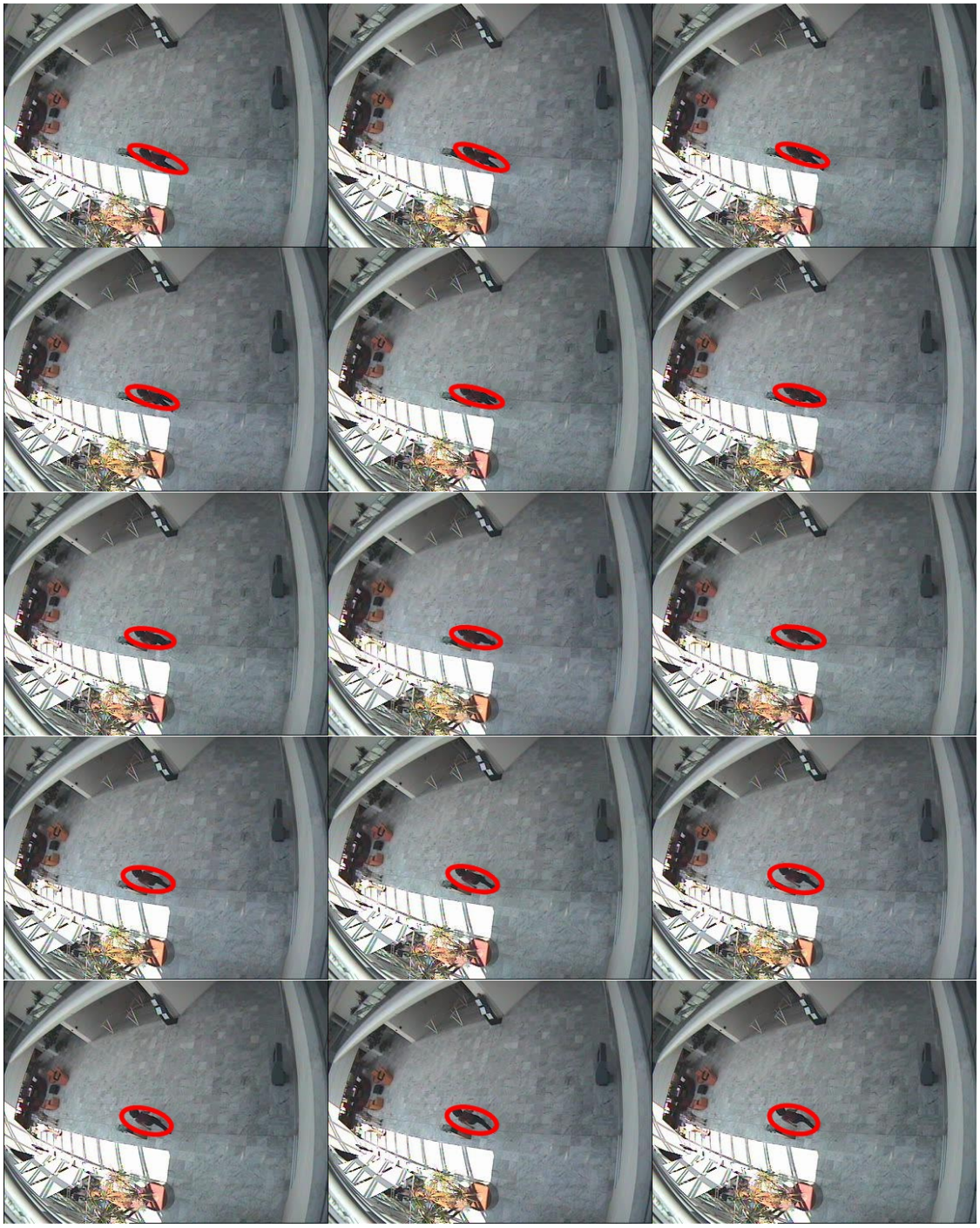


Figure E.12: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 46 to Frame 60 (From top left to top right and then to bottom) in Video sequence CAVIAR.



Figure E.13: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 61 to Frame 75 (From top left to top right and then to bottom) in Video sequence CAVIAR.



Figure E.14: Multi CAMShift Agents object tracking result when tracking a pedestrian in Frame 76 to Frame 90 (From top left to top right and then to bottom) in Video sequence CAVIAR.

Appendix F:
Multi CAMShift Agent Tracked Position Changing Curves

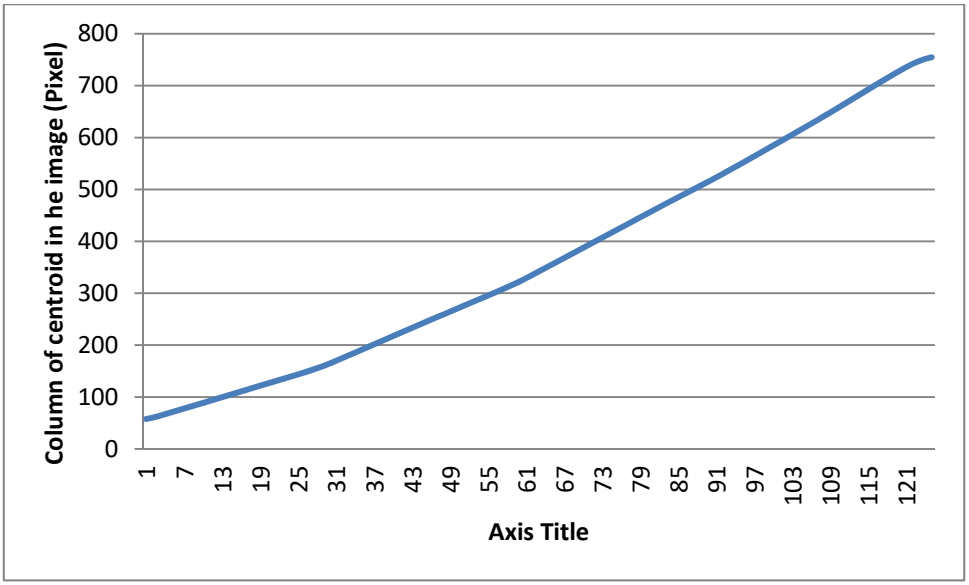


Figure F.1: Ground truth position changing curves of the pedestrian centroid in a horizontal direction in video (PETS 2009)

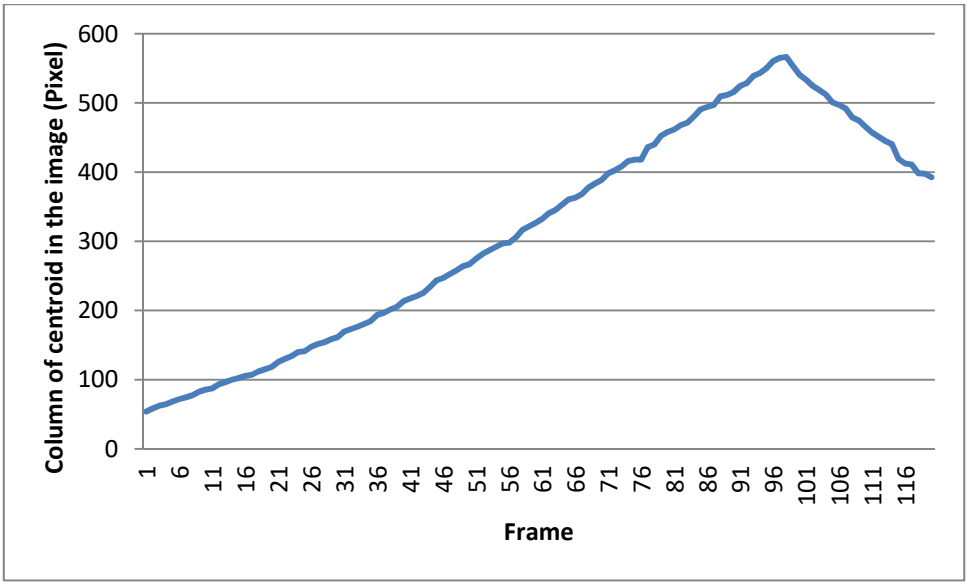


Figure F.2: Multi CAMShift Agent tracked position changing curves of the pedestrian centroid in a horizontal direction in video (PETS 2009)

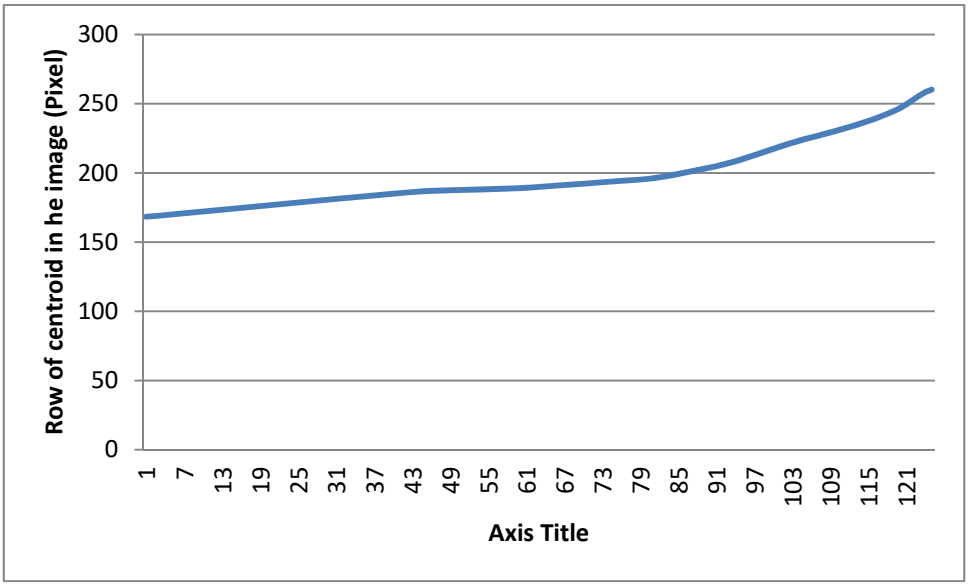


Figure F.3: Ground truth position changing curves of the pedestrian centroid in a vertical direction in video (PETS 2009)

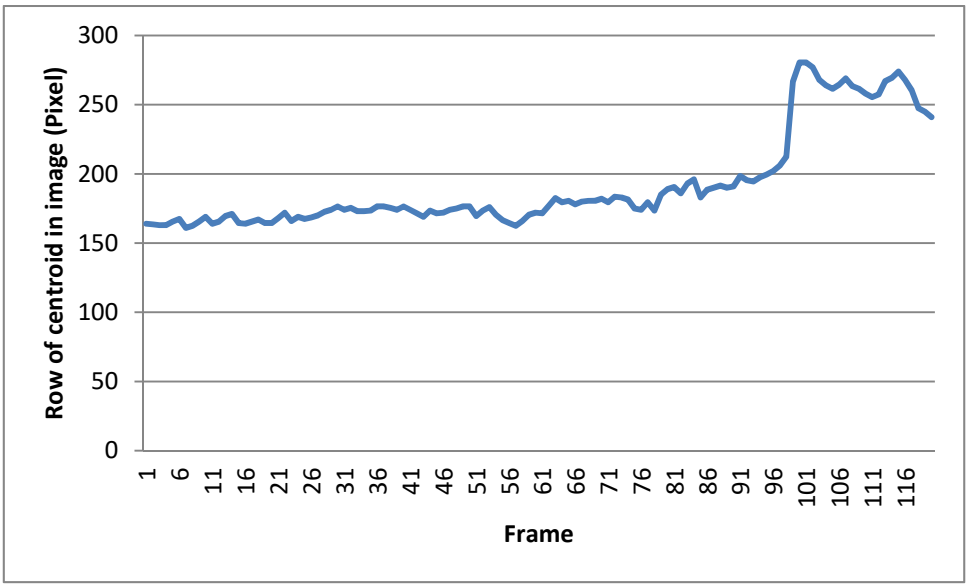


Figure F.4: Multi CAMShift Agent tracked position changing curves of the pedestrian centroid in a horizontal direction in video (PETS 2009)

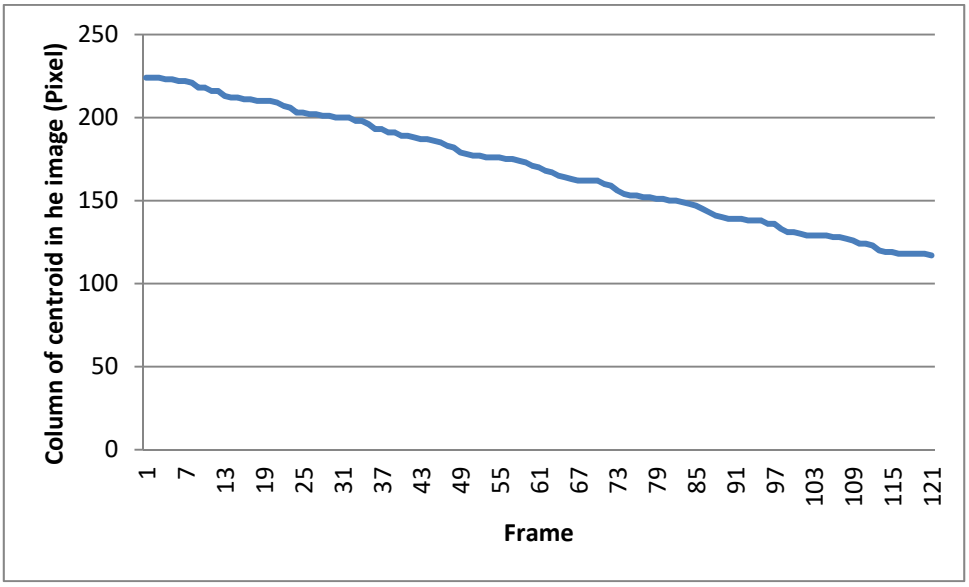


Figure F.5: Ground truth position changing curves of the pedestrian centroid in a horizontal direction in video (CAVIAR 2009)

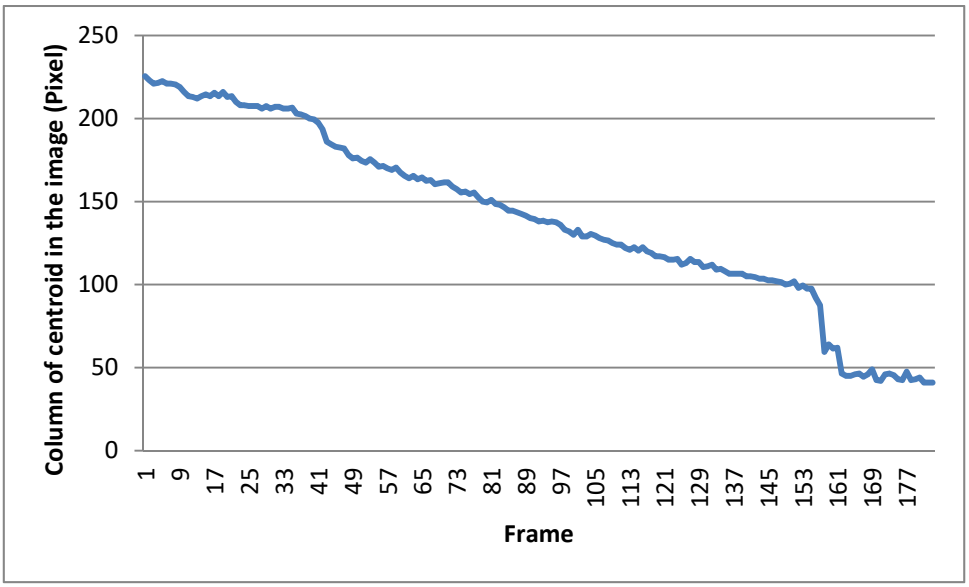


Figure F.6: Multi CAMShift Agent tracked position changing curves of the pedestrian centroid in a horizontal direction in video (CAVIAR 2009)

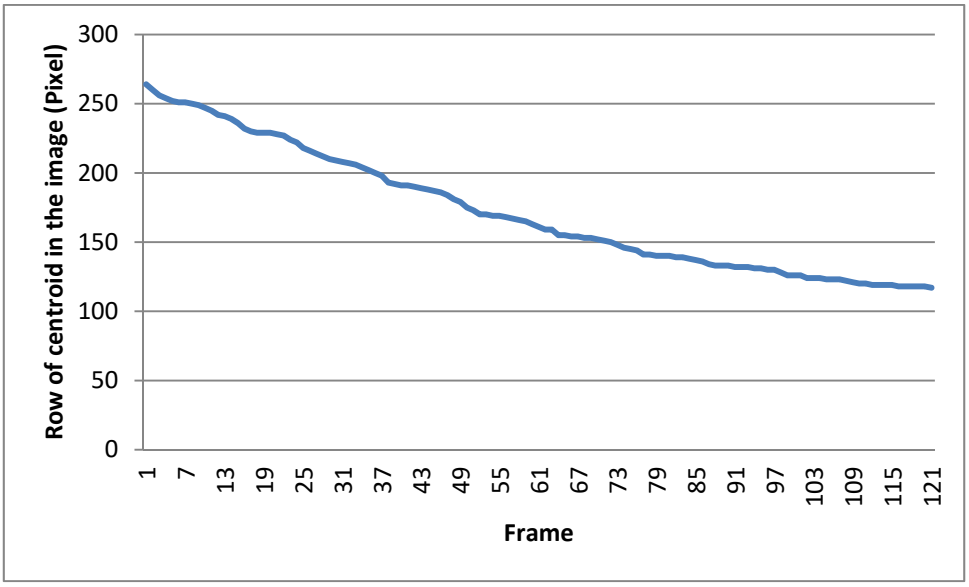


Figure F.7: Ground truth position changing curves of the pedestrian centroid in a vertical direction in video (CAVIAR)

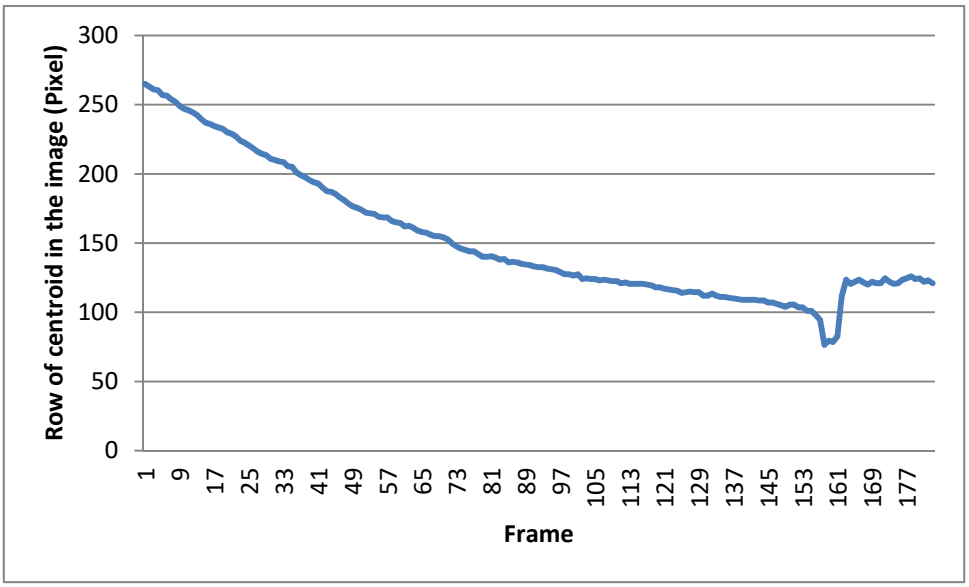


Figure F.8: Multi CAMShift Agent tracked position changing curves of the pedestrian centroid in a vertical direction in video (CAVIAR)