



저작자표시-비영리-동일조건변경허락 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



동일조건변경허락. 귀하가 이 저작물을 개작, 변형 또는 가공했을 경우에는, 이 저작물과 동일한 이용허락조건하에서만 배포할 수 있습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

PH.D. DISSERTATION

**Bio-mimetic Models for Detection
and Tracking of Moving Objects**

움직이는 물체 검출 및 추적을 위한 생체 모방 모델

By

Kwang Moo Yi

January 2014

SCHOOL OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Bio-mimetic Models for Detection and Tracking of Moving Objects

움직이는 물체 검출 및 추적을 위한

생체 모방 모델

지도 교수 최진영

이 논문을 공학박사 학위논문으로 제출함

2014년 1월


서울대학교 대학원


전기컴퓨터공학부

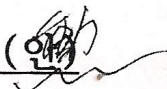
이광무


이광무의 공학박사 학위논문을 인준함


2014년 1월

위원장 장 병 탁 (인) 

부위원장 최 진 영 (인) 

위원 강 훈 (인) 

위원 곽 노 준 (인) 

위원 오 성 희 (인) 

Abstract

In this thesis, we propose bio-mimetic models for motion detection and visual tracking to overcome the limitations of existing methods in actual environments. The models are inspired from the theory that there are four different forms of visual memory for human visual perception when representing a scene; visible persistence, informational persistence, visual short-term memory (VSTM), and visual long-term memory (VLTM). We view our problem as a problem of modeling and representing an observed scene with temporary short-term models (TSTM) and conservative long-term models (CLTM). We study on building efficient and effective models for TSTM and CLTM, and utilizing them together to obtain robust detection and tracking results under occlusions, clumsy initializations, background clutters, drifting, and non-rigid deformations encountered in actual environments.

First, we propose an efficient representation of TSTM to be used for moving object detection on non-stationary cameras, which runs within 5.8 milliseconds (ms) on a PC, and real-time on mobile devices. To achieve real-time capability with robust performance, our method models the background through the proposed dual-mode kernel model (DMKM) and compensates the motion of the camera by mixing neighboring models. Modeling through DMKM prevents the background model from being contaminated by foreground pixels, while still allowing the model to be able to adapt to changes of the background. Mixing neighboring models reduces the errors arising from motion compensation and their influences are further reduced by keeping the age of the model. Also, to decrease computation load, the proposed method applies one DMKM to multiple pixels without performance degradation. Experimental results show the compu-

tational lightness and the real-time capability of our method on a smart phone with robust detection performances.

Second, by using the concept from both TSTM and CLTM, a new visual tracking method using the novel tri-model is proposed. The proposed method aims to solve the problems of occlusions, background clutters, and drifting simultaneously with the new tri-model. The proposed tri-model is composed of three models, where each model learns the target object, the background, and other non-target moving objects online. The proposed scheme performs tracking by finding the best explanation of the scene with the three learned models. By utilizing the information in the background and the foreground models as well as the target object model, our method obtains robust results under occlusions and background clutters. Also, the target object model is updated in a conservative way to prevent drifting. Furthermore, our method is not restricted to bounding-boxes when representing the target object, and is able to give pixel-wise tracking results.

Third, we go beyond pixel-wise modeling and propose a local feature based tracking model using both TSTM and CLTM to track objects in case of uncertain initializations and severe occlusions. To track objects accurately in such situations, the proposed scheme uses “motion saliency” and “descriptor saliency” of local features and performs tracking based on generalized Hough transform (GHT). The proposed motion saliency of a local feature utilizes instantaneous velocity of features to form TSTM and emphasizes features having distinctive motions, compared to the motions coming from local features which are not from the object. The descriptor saliency models local features as CLTM and emphasizes features which are likely to be of the object in terms of its feature descriptors. Through these saliencies, the proposed method tries to “learn and find” the target object rather than looking for what was given at initialization, becoming robust

to initialization problems. Also, our tracking result is obtained by combining the results of each local features of the target and the surroundings, thus being robust against severe occlusions as well. The proposed method is compared against eight other methods, with nine image sequences, and hundred random initializations. The experimental results show that our method outperforms all other compared methods.

Fourth and last, we focus on building robust CLTM with local patches and their neighboring structures. The proposed method is based on sequential Bayesian inference and focuses on solving both the problem of tracking under partial occlusions and the problem of non-rigid object tracking in real-time on desktop personal computers (PC). The proposed scheme is mainly composed of two parts: (1) modeling the target object using elastic structure of local patches for robust performance; and (2) efficient hierarchical diffusion method to perform the tracking process in real-time. The elastic structure of local patches allows the proposed scheme to handle partial occlusions and non-rigid deformations through the relationship among neighboring patches. The proposed hierarchical diffusion generates samples from the region where the posterior is concentrated to reduce computation time. The method is extensively tested on a number of challenging image sequences with occlusion and non-rigid deformation. The experimental results show the real-time capability and the robustness of the proposed scheme under various situations.

Keywords: Bio-mimetic model, Visual tracking, Motion detection, Temporary short-term model, Conservative long-term model

Student ID Number: 2007-23039

Contents

1	Introduction	1
1.1	Background and Research Issues	2
1.1.1	Issues in Motion Detection	2
1.1.2	Issues in Object Tracking	4
1.2	The Human Visual Memory	11
1.2.1	Sensory Memory	12
1.2.2	Visual Short-Term Memory	13
1.2.3	Visual Long-Term Memory	16
1.3	Bio-mimetic Framework for Detection and Tracking	17
1.4	Contents of the Research	18
2	Detection by Pixel-wise Dual-Mode Kernel Model	22
2.1	Proposed Method	23
2.1.1	Approximated Gaussian Kernel Model	24
2.1.2	Dual-Mode Kernel Model (DMKM)	26
2.1.3	Motion Compensation by Mixing Models	29
2.1.4	Detection of Foreground Pixels	31

2.2	Experimental Results	32
2.2.1	Runtime Comparison	33
2.2.2	Qualitative Comparison	35
2.2.3	Quantitative Comparison	36
2.2.4	Effects of Dual-Mode Kernel Model	37
2.2.5	Effects of Motion Compensation	40
2.2.6	Mobile Results	40
2.3	Remarks and Discussion	41
3	Tracking by Pixel-wise Tri-Model Representation	42
3.1	Tri-Model Framework	43
3.1.1	Overall Scheme	43
3.1.2	Advantages	47
3.1.3	Practical Approximation	48
3.2	Tracking with the Tri-Model	50
3.2.1	Likelihood of the Tri-Model	50
3.2.2	Likelihood Maximization	51
3.2.3	Estimating Pixel-Wise Labels	54
3.3	Learning the Tri-Model	55
3.3.1	Target Model	55
3.3.2	Background Model	56
3.3.3	Foreground Model	57
3.4	Experimental Results	59
3.4.1	Experimental Settings	59
3.4.2	Tracking Accuracy: Bounding Box	60

3.4.3	Tracking Accuracy: Pixel-Wise	63
3.5	Remarks and Discussion	64
4	Tracking by Feature-point-wise Saliency Model	65
4.1	Proposed Method	66
4.1.1	Tracking based on GHT	67
4.1.2	Descriptor Saliency and Feature DB Update	70
4.1.3	Motion Saliency	73
4.2	Experimental Results	75
4.2.1	Tracking with Inaccurate Initializations	77
4.2.2	Tracking Under Occlusions	81
4.3	Remarks and Discussion	81
5	Tracking by Patch-wise Elastic Structure Model	84
5.1	Tracking with Elastic Structure of Local Patches	85
5.1.1	Sequential Bayesian Inference Framework	85
5.1.2	Elastic Structure of Local Patches	87
5.1.3	Modeling a Single Patch	89
5.1.4	Modeling the Relationship between Patches	91
5.1.5	Model Update	93
5.1.6	Hierarchical Diffusion	94
5.1.7	Summary of the Proposed Method	96
5.2	Experiments	96
5.2.1	Parameter Effects	96
5.2.2	Performance Evaluation	99

5.2.3	Discussion on Translation, Rotation, Illumination Changes	103
5.2.4	Discussion on Partial Occlusions	104
5.2.5	Discussion on Non-Rigid Deformations	106
5.2.6	Discussion on Additional Cases	106
5.2.7	Summary of Tracking Results	108
5.2.8	Effectiveness of Hierarchical Diffusion	110
5.2.9	Limitations	112
5.3	Remarks and Discussion	116
6	Concluding Remarks and Future Works	117
	Bibliography	120
	Abstract in Korean	130

List of Figures

1.1	Example of tracking in case of occlusions and with clumsy initializations.	5
1.2	Contents of the research and their relatedness to short-term/long-term memory and how data is treated (from pixel-wise to patch-wise).	19
2.1	Example of our method running on a mobile device.	23
2.2	Framework of the proposed method	24
2.3	Illustration of the effects of using dual-mode kernel model.	27
2.4	Illustration of the proposed motion compensation by mixing models.	30
2.5	Example critical frames for each method.	34
2.6	Example result with the kernel model and the dual-mode kernel model.	38
2.7	Comparison between nearest neighbor warping and proposed warping.	39
2.8	Example detection results of our method on a mobile device.	41
3.1	Example of each model.	44
3.2	Example of the pixel-wise likelihood values.	46

3.3	Example of each model and the observation warped into the target domain.	48
3.4	Example of the pixel-wise likelihood values in the target domain.	49
3.5	Precision plots for all sequences.	61
3.6	Bounding box results for critical frames.	62
3.7	Pixel-wise tracking results.	62
4.1	Overall scheme of the proposed method.	68
4.2	Illustration of GHT voting with SURF features	68
4.3	Illustration of the descriptor saliency in action.	71
4.4	Example of motion saliency obtained for the woman sequence.	74
4.5	Box plots for % correctly tracked with all initializations.	78
4.6	Critical frames for tracking results.	82
5.1	Example of elastic structure of local patches used to describe the target object.	88
5.2	Example of a 21 dimensional feature descriptor for a single local patch.	90
5.3	Example of neighboring local patches connected together.	91
5.4	Illustrative example of hierarchical diffusion performed for a single sample l	95
5.5	Example showing the effect of β parameter on non-rigid object tracking.	98
5.6	Example showing the effect of β parameter on tracking objects with partial occlusions.	98
5.7	Mean errors for each sequence.	102

5.8	Tracking results for the Dudek sequence and the Sylvester sequence.	103
5.9	Tracking results for the Face sequence, the Woman sequence, and the Caviar sequence.	105
5.10	Tracking results for the High Jump sequence, the Motocross 1 sequence, and the Mtn. Bike sequence.	107
5.11	Tracking results for the Robot sequence, the Dove sequence, the Pedestrian sequence, and the Nemo sequence.	109
5.12	Mean error obtained using hierarchical diffusion and simple Gaussian diffusion.	111
5.13	Example tracking results and their local patch structures.	113
5.14	Example tracking results with scale and orientation changes.	114
5.15	Change in the coordinates of the top-left corner point of the estimated bounding box for each frame.	115

List of Tables

2.1	Average computation time for each method.	33
2.2	Quantitative results for each method.	36
3.1	Area under precision plots.	59
3.2	F-measure for pixel-wise tracking results.	63
4.1	Results for all sequences.	80
5.1	The mean error values and the percentage of meaningful tracking results with all frames in all image sequences for each algorithm.	108
5.2	The mean error and the percentage of meaningful tracking results for each algorithms with all sequences.	108

Chapter 1

Introduction

Detecting and tracking of moving objects in a scene is an important task for many vision based systems. Intelligent visual surveillance systems require the motion detection and object tracking as a preliminary for advanced inference tasks (Kim et al., 2010). Over the last two decades, many object tracking methods have been proposed (Yilmaz et al., 2006; Kalal et al., 2010; Babenko et al., 2011; Adam et al., 2006; Godec et al., 2013) and various motion detection approaches have been explored (Stauffer & Grimson, 1999; Elgammal et al., 2002; Sheikh & Shah, 2005; Georgiadis et al., 2012; Sheikh et al., 2009). However, the practicability of these methods has been limited due to many reasons. In case of motion detection, errors arising from compensating the motion of the camera and the computational load greatly limit the applicability. For object tracking, problems such as occlusions, background clutters, deformations, sensitivity to initializations, and drifting are just a few examples.

In this thesis, we focus on overcoming the limitations of the current state-of-the-art methods for motion detection and tracking through bio-mimetic ideas inspired from human visual perception mechanism. Although the mechanism of

human visual perception is not perfectly known, recent research suggests there are four different forms of visual memory (Irwin, 1992; Palmer, 1999) which are used to form a representation of a scene; visible persistence, informational persistence, visual short-term memory (VSTM), and visual long-term memory (VLTM). Inspired from this memory structure, we view our problem as a problem of modeling and representing the observed scene with temporary short-term models (not to be confused with VSTM) and conservative long-term models (not to be confused with VLTM). The temporary short-term model (TSTM) takes roles similar to what visible persistence, informational persistence, and VSTM do for human visual perception. The TSTM is focused on building the temporal integration of the observation and the instantaneous motion and changes within the scene. The conservative long-term model (CLTM) learns object-wise information, such as the target object information in tracking, throughout the whole given sequence. This is similar to how VLTM works for human visual perception. Through appropriate use of each type of model (short-term and long-term), we aim to obtain robust results outperforming the state-of-the-art.

1.1 Background and Research Issues

1.1.1 Issues in Motion Detection

Detection of moving objects in a scene is without doubt an important topic in the field of computer vision. It is one of the basic steps in many vision-based systems. For example, applications such as human computer interface (HCI), robot visions, and intelligent surveillance systems (Kim et al., 2010) require detection of moving objects. Various methods have been proposed and have proven to be successful for detection of moving objects in case of stationary cameras (Stauffer & Grimson, 1999; Elgammal et al., 2002; Ko et al., 2010; Sheikh & Shah, 2005), but in case

of mobile or pan-tilt-zoom (PTZ) cameras these methods do not work well due to many unaccounted factors that arise when using of movable cameras.

Many methods have been proposed for moving object detection with a non-stationary camera, but the applicability of them is still doubtful. The most critical reason restricting the applicability of these methods is the amount of computation they require to work. In (Georgiadis et al., 2012), the authors noted that it took 30 to 60 seconds per frame with their un-optimized MATLAB implementation, and also noted that (Sheikh et al., 2009) takes over 2.6 seconds. The method proposed in (Kwak et al., 2011) also requires much computation and cannot run in real-time since they use dense optical flows and nonparametric belief propagation (BP) to optimize a Markov random field (MRF). Therefore, even though these algorithms may provide promising results off-line, in real-time, they are unusable unless a machine with great computation power is provided. The methods presented in (Kim et al., 2013, 2012) work in real-time, but they are still not enough when considering that other visual inference task are usually performed after detection, or when considering platforms with less computation power. Smart phones or embedded platforms, such as robots or head mount displays, would be examples of platforms with relatively low computational power which could benefit much from fast motion detection.

To reduce the computation load of methods targeted for non-stationary cameras, it is important that the model design itself also considers the computation load required for applying the model, such as the computation load arising from motion compensation. For example, the method proposed by Barnich and Droogenbroeck (Barnich & Van Droogenbroeck, 2011) is one of the well-known fast background subtraction algorithms showing robust performances. However, when applied to non-stationary cameras, the motion compensation procedure for the algorithm requires computation load proportional to the number of samples

used for a pixel. This could slow down the method in significant amounts (usually requiring more computation than the detection algorithm itself), unless there is some sort of hardware support.

Besides the computation load, when modeling the scene, it is also important that the model considers not only the errors and noises that arise in stationary cameras, but also the errors that arise when compensating for the motion of the camera. This is a critical reason that we cannot just simply apply background subtraction algorithms for stationary cameras with simple motion compensation techniques. Stationary camera background modeling algorithms usually focus on building a precise model for each pixel. But for non-stationary case, we cannot guarantee that the model used to evaluate a pixel is actually relevant to that pixel. Even the slightest inaccuracy in motion compensation could end up in making the algorithm use wrong models for some pixels. To account for such motion compensation errors, in (Rao et al., 2007; Kim et al., 2013; Mittal & Huttenlocher, 2000), small nearby neighborhoods are considered. However, considering neighborhoods increases the necessary computation, slowing down the whole algorithm.

1.1.2 Issues in Object Tracking

Object tracking is an important computer vision problem which can be used for various applications. Example applications of object tracking include, robot vision, video analysis, behavior recognition, and home automation, visual surveillance (Kim et al., 2010). In order for the whole system to work properly for these applications, accurate tracking results are required. Various tracking methods have been tried during the last decade and they have proven to be successful for these applications (Yilmaz et al., 2006). However, the applicability of these algorithms are somewhat limited to “lab environments” and do not show satisfying performances when applied on real-world scenarios.

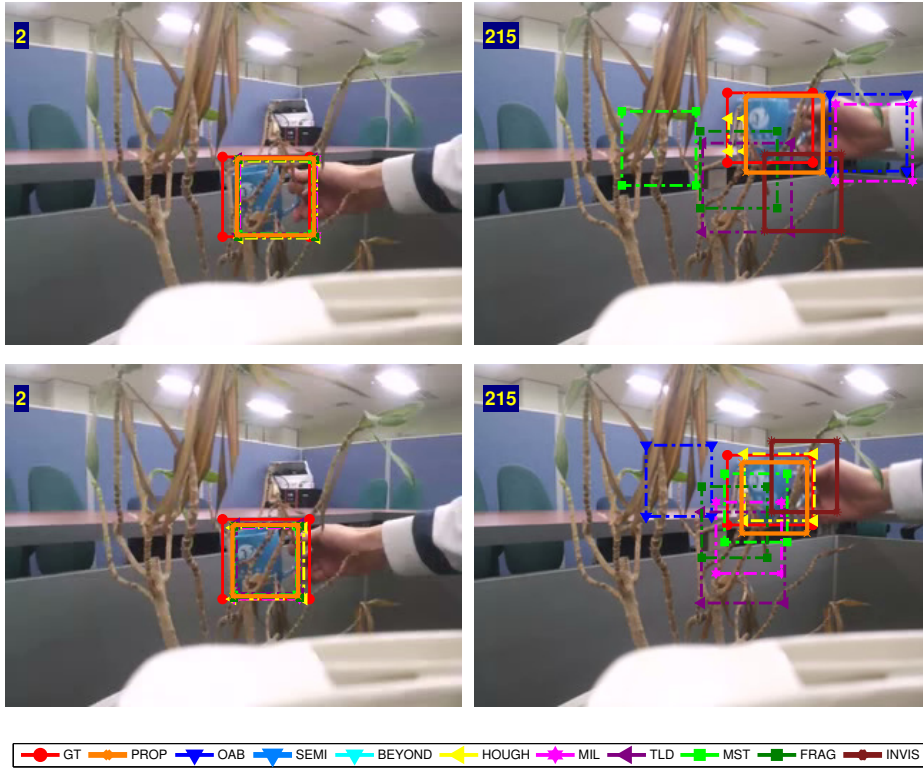


Figure 1.1: Example of tracking in case of occlusions and with clumsy initializations. Different initializations for the **occCup** sequence (left column), leading to different results for the same frame (right column). *Best viewed in color.*

The limitations of conventional methods arise from the fact that they have strong assumptions about the input video sequence, such as constant movements of the target object and consistent views. In real-world scenarios, occlusions may occur frequently with the target object showing non-rigid deformations, degrading the performance of conventional methods. These major issues in object tracking include severe occlusions, uncertain initializations, drifting problems, background clutter problems, and tracking objects with non-rigid deformations. Especially, in case of automatic initialization, the applicability of conventional methods are still limited (illustrated in Figure 1.1.)

Drifting and Background Clutter

The problem of tracking under occlusions and background clutters has been tackled by various methods. The methods for dealing with occlusions described in the preceding subsection (Adam et al., 2006; Mei & Ling, 2009) have been used to deal with occlusion from backgrounds. In case of clutter problems, Grabner et al. (Grabner et al., 2006) proposed a method to train a discriminative classifier in an online manner using online boosting to separate the object from the background. Their method falls into the group of methods treating object tracking as a “two-class” problem (Avidan, 2007), which try to solve background clutter problems. The method is further extended by Stalder et al. (Stalder et al., 2009) to solve drifting problems as well. The methods stated above however focus only on either occlusion problems or background clutter problems, making the performance of the methods limited when both problems occur simultaneously.

Drifting is another well-known problem of traditional tracking methods. As trackers must adapt to various appearance changes of the target object, the target model must be updated on-line. However, traditional methods use the tracking result from previous frames to update the model, which may be incorrect. False-positives and false-negatives may be used in this update process, which makes the model “drift away” from correct answer. Babenko et al. (Babenko et al., 2011) extended the results of Grabner et al. (Grabner et al., 2006) by employing multiple instance learning, which updates the classifier with multiple positive examples rather than just one, and reduced the risk of false positives getting involved in the learning process. Also, in (Stalder et al., 2009), a multiple classifier system composed of learned and un-learned parts is proposed to prevent the tracker from drifting. Kalal et al. (Kalal et al., 2010) proposed a method using both tracker and classifier to create training sets with structural constraints. Their method collects training samples and uses them only when the structural constraint meets (*i.e.*

only when the classifier and the tracker agrees). Through this procedure, their method becomes robust to drifting problems. However, these methods do not consider occlusions, making the methods vulnerable to them.

Initializations Sensitivity and Occlusions

One of the major assumptions many conventional trackers have is that the target object is given rather accurately. Therefore, the trackers are sensitive to how they were initialized at the first frame. Practically, this is not a trivial task and leads to loss in tracking performances. Also, in actual environments, severe occlusions exist where almost all of the target object is occluded, which not many trackers are capable of dealing with.

The performance degradation of conventional trackers under inaccurate initializations is a problem which has not been well addressed yet. There are methods with automatic initialization for trackers such as the method by Mahadevan and Vasconcelos (Mahadevan & Vasconcelos, 2009), but still, they do not account for cases when these initializations fail to give an accurate initialization. Performance degradation from inaccurate initializations is closely related to drifting problems. In many tracking methods, such as (Comaniciu et al., 2003; Adam et al., 2006; Grabner et al., 2006, 2008; Stalder et al., 2009), the model for the target object is constructed using the initial target information given at the first frame. Then, the methods *adapt* the target model with the tracking results for each frames. During the adaptation, rather than learning about the target object and enhancing the model, noise gets involved in the learning process and the performance of the tracker degrades. This drifting phenomenon is evident even in online boosting based methods (Grabner et al., 2006) or methods which create classifiers (Avidan, 2007; Mahadevan & Vasconcelos, 2009). This drifting would cause trackers to be more sensitive to initializations, since having more noise from the beginning

would cause faster drifting. In (Grabner et al., 2008), (Stalder et al., 2009), and (Babenko et al., 2011), the authors tackle the drifting problem with sophisticated learning strategies. Both methods show robust results against drifting, but still are vulnerable against inaccurate initializations. A recent method by Kalal et al. (Kalal et al., 2010) uses P-N learning scheme, which incorporates results from both detector and tracker to avoid such problems, but still, their work is mainly focused on the problem of trackers drifting. Even without considering drifting problems, a small inaccuracy in initialization can cause much problem.

A major cause for conventional trackers being over sensitive to initializations is much based on the fact that they treat the given initialization as a fixed prior. When we only have a single frame to use, constructing the target model solely based on the initialization is reasonable. However, as more frames or more data is given, it is obvious that we would need to *figure out* what the target object is like, rather than just trying to *adapt* the model we learned from the initialization. This means that we need to question the given prior and not believe it thoroughly. Godec et al. (Godec et al., 2013) recognize this problem as a limitation of bounding-box based approaches. Using the initialization by a bounding-box, they build a classifier to roughly classify pixels into foreground and background. With the rough classification result, they again perform Grab-cut (Rother et al., 2004) to segment the target object. Though their aim was to well-describe non-rigid deformations and reduce drifting effects, their method is closely related to the initialization problem of trackers. Unfortunately, their method suffers from randomness of the performance due to the purely random nature of their classifier.

Tracking objects undergoing occlusions have been the aim of many research, but still have problems in case of severe occlusions which happens in actual environments. Adam et al. (Adam et al., 2006) modeled the target object with partial representations, making their method robust to partial occlusions. Their

method, however, is an extension of kernel-based methods (Comaniciu et al., 2003) and therefore weak against scale changes or background clutters. In (Mei & Ling, 2009) the authors built a sparse representation of the target object to track the target object. They used trivial templates to describe occluded parts and thus becoming robust to occlusions. However, both methods assume that at least some parts or majority of the object is visible. This becomes critical when severe occlusion occurs. Grabner et al. (Grabner et al., 2010) considered nearby features as well as the target object to track the target object even when the target object is fully occluded. Their method denotes these nearby feature points as *supporters*, and the supporters help track the object when the primary tracker fails to track. However, the performance of their method relies much on the primary tracker being used since the primary tracker results are considered to be accurate when learning supporters.

Non-Rigid Deformations

To solve the problem of non-rigid object tracking, Kwon and Lee (Kwon & Lee, 2009) proposed a method which models the target object as a collection of local patches. In (Kwon & Lee, 2009), the target object is described with a star model of local patches, and Adaptive Basin Hopping Monte Carlo (A-BHMC) sampling is used to minimize the energy of the model. The patches in the model used to describe target object are consistently updated through a heuristic scheme. This enables the tracker to be able to adapt to drastic changes in the appearance and the shape of the target. A-BHMC reduces the number of particles required for tracking, making the computation time tractable. However, their method tends to have trouble when tracking objects showing large displacements, and still requires large amount of computation even with A-BHMC. Godec *et al.* (Godec et al., 2013) proposed a tracking method based on the generalized Hough transform.

They extended the idea of Hough Forests to the online domain and coupled the voting based detection and back-projection with a rough segmentation based on GrabCut (Rother et al., 2004). Their method gets rid of the bounding-box limitation and returns tracking results which contains only the target object. However, their method is limited to handling non-rigid deformations only without considering occlusions.

Limitations of Bounding Box Representation

Many modern trackers use bounding boxes for initializations and representation of their results. This is due to the ease in initialization and the prominent use in many object detection methods. However, using bounding boxes bring limitations, as noted by Godec et al. (Godec et al., 2013). Trackers representing the target object with a bounding box inevitably include non-target information in their models and even use them when estimating the target object information. This leads to many problems such as the performance degradation when tracking objects under occlusions, background clutter problems, and drifting of models. Also, representing tracking results with bounding boxes only gives limited information about the target.

Various research has been done to overcome these limitations of bounding box based trackers. Adam et al. (Adam et al., 2006) used fragments-based representations to obtain robust results even with occlusions, and “tracking-by-detection” approaches using classifiers (Grabner et al., 2006, 2008; Stalder et al., 2009) were proposed to deal with background clutter problems and drifting problems. Babenko et al. (Babenko et al., 2011) further investigated the drifting problem and proposed a learning strategy using multiple instance learning to solve drifting. Kalal et al. (Kalal et al., 2010) also proposed a tracking method which uses both tracker and a detector to avoid drifting and obtain robust tracking results.

For accurate representations of the target object, Godec et al. (Godec et al., 2013) obtains accurate segmentation of the target by integrating the Hough forest classifier with a segmentation method. Though these methods have good tracking performances when applied to their specific tasks, as demonstrated by Yi et al. (Yi et al., 2012) they tend to show not as good results for problems they have not dealt with. Yi et al. (Yi et al., 2012) tries to overcome these problems simultaneously by modeling both the background and the target object and using a likelihood measure constructed from both models. Their method shows good performances in many cases, but is still limited to bounding boxes, and has problems when dealing with occlusions caused by other moving objects.

1.2 The Human Visual Memory

To resolve the issues raised in Section 1.1, we draw our attention to how the human visual memory works. In case of human, recognizing objects in a scene and tracking them is not a hard thing to do. Even the state-of-the-art methods consider the performance of human visual inference as the goal of the research. Therefore, it is natural that understanding of how human visual system works and mimicking it would help solving problems related to visual inference tasks such as tracking and detection.

Though it is still not clearly known how human visual system works, Hollingworth suggested that there are four different forms of visual memory (Hollingworth et al., 2004). According to the theory, there are four different forms of visual memory (Irwin, 1992; Palmer, 1999) which are used to form a representation of a scene; visible persistence, informational persistence, visual short-term memory (VSTM), and visual long-term memory (VLTM). The former two persistences together, are also referred to as the sensory memory (or iconic memory). They

are directly related to low-level sensory traces point-by-point, decay very quickly, and are affected by masking. In case of computer vision, visible persistence can be understood as raw pixel data, or pixel data with some basic pre-processing, and informational persistence are related to local features. The VSTM holds representations abstracted from precise sensory information. It is known to have limited capacities from three to four objects (Luck & Vogel, 1997; Pashler, 1988), and to be less precise in terms of spatial accuracy. However, VSTM is considerably more robust than sensory memories in that it is not significantly affected by masking or blinking, and can be maintained for longer durations. Finally, the VLTM is known to maintain visual representations in a similar form as VSTM but has massive capacity. The exact capacity of VLTM is not clearly known, but is able to hold image representations of thousands of objects for long periods of time (Brady et al., 2008).

1.2.1 Sensory Memory

The sensory memory, or the iconic memory in case of visual sense, is described as a very brief (less than one second), pre-categorical, and high capacity memory store (Sperling, 1960; Dick, 1974). Iconic memory is thought to act as input data for VSTM, by providing a coherent representation of the scene for a brief period of time. However, iconic memory stores a great deal more information than can be stored normally in short-term memory, having virtually unlimited availability (Dick, 1974). Classic experimental investigations such as the one by Sperling (Sperling, 1960) was one of the early work giving insights to this component of the human visual memory. Iconic memory assists in explaining phenomena such as change blindness, continuity of experience during saccadic eye movements, and the well-known temporal integration of still images (for example, movies).

The evidence for the existence of iconic memory was investigated through

experiments in early days (Dick, 1974). One of the classic experiments include the “partial-report” experiment by Sperling (Sperling, 1960). In partial-report experiments, a target stimulus is given tachistoscopically, followed by a cue stimulus. Then, the subject is required to identify the portion of the target specified by the cue. The target is typically an alphanumeric character, or some simple mark such as arrows or lines. When the cue was given immediately, the accuracy of report is high, and as the cue id delayed, accuracy declined monotonically. The asymptote of the delay curve occurred at point between 250 and 1000 ms after the termination of the stimulus (Dick, 1974). These results provide clues of a memory system lasting very briefly.

1.2.2 Visual Short-Term Memory

The visual short-term memory is a memory system which keeps the visual information for a few seconds, and is used for other ongoing cognitive tasks (Luck, 2007). In contrast to iconic memory, VSTM representations are longer lasting, more abstract, and more durable. They can survive eye movements, blinks, and other visual interruptions, maintaining continuity (Luck, 2007). However, VSTM has highly limited storage capacity and creates largely schematic representations rapidly.

In case of actual storage, VSTM representations are stored by sustained firing of action potentials (Luck, 2007). This can be observed directly in monkeys by recording the activity of neurons in VSTM tasks. When a monkey is exposed to a to-be-remembered stimulus, specific neurons began to and continue to fire during the delay interval. Neural activity during the delay period of a VSTM task can also be seen in neuroimaging studies (Cohen et al., 1997) and event-related potential studies (Vogel & Machizawa, 2004). It is also thought that delay activity is related with recurrent neural networks. This means that the activities

in sensory neurons ultimately flows back to them and making them continue to fire even when the stimulus has been removed (Raffone & Wolters, 2001).

The capacity and how representations are stored in VSTM is not clearly known but it is well accepted that the capacity of VSTM is limited to 3 to 4 objects. In early studies of VSTM using alphanumeric characters, it was suggested that the capacity limit was 4-5 items (Sperling, 1960), but it was not clear that they were stored visually or verbally. Luck and Vogel (Luck & Vogel, 1997) used basic visual features to limit the contributions from verbal short term memory and estimated capacity of 3 to 4 objects. In fact, in their work they added verbal loads to limit the recoding of visual information into verbal forms. The verbal load was given by making participants to consistently speak out loud the given two digits for each trial. Large number of studies (Besner et al., 1981; Henson, 1998; Levy, 1971; Murray, 1967) report that this procedure dramatically impairs the recoding of visual information into verbal form.

Though the capacity of visual is limited to 3 to 4 objects, it is not yet clear if the capacity depends on the amount of information or the number of objects. One view considers objects as the storage unit for VSTM. This means that the VSTM acts as if it consists of 3-4 high-resolution “slots” and therefore the number of objects is what affects the storage capacity. In the study by Luck and Vogel (Luck & Vogel, 1997), they demonstrated that the performance of the observer’s memory does not change significantly with the number of features. In their experiment, the performance of the test subjects remained unchanged when given both color and orientations as features, or only one of the two. They further showed that objects defined by four features could be remembered in the same performance as an object defined by only one feature. As an alternative, Magnussen et al. (Magnussen et al., 1996) suggested that each feature dimension is represented in separate memory store. This was further investigated to find that

VSTM performance is worse when multiple features are drawn from the same feature dimension (Luck, 2007).

Another view on VSTM proposes that the capacity limit of VSTM is related to the amount of information not the number of objects. In this view, it is considered that VSTM has limited “resources” which is divided among the memorized items. Thus the resolution of the memory representation decreases as the number of items increase (Alvarez & Cavanagh, 2004; Vogel et al., 2001; Wilken & Ma, 2004). In Alvarez and Cavanagh’s work (Alvarez & Cavanagh, 2004), they found that estimated capacity of VSTM decreases as the difficulty among discriminating items increased. They also estimated that the maximum capacity with the simplest items is approximately 4-5 items. Their results support the theory that when more details are required VSTM becomes more limited.

VSTM can be thought as a buffer for temporary information storage. VSTM takes role in bridging the sensory gaps coming from eye movements and blinks. Saccadic eye movements occur a few times every second, which introduces gaps in the sensory data. This results in a series of spatially shifted snapshots of the overall scene, separated with brief gaps (Luck, 2007). When constructing a scene representation with VSTM and VLTM (Hollingworth et al., 2004), VSTM is thought to bridge the gaps between these snapshots (Irwin, 1991) and to allow the relevant portions of each subsequent snapshots to be aligned with each other (Currie et al., 2000). VSTM is also thought to play an important role in keeping track of previously attended locations when searching for an object in complex scenes (Luck, 2007). Results from *inhibition-of-return* experiments have shown that after attention has visited a place, it tends not to revisit the same location for some time (Klein, 2000; Peterson et al., 2001; Posner & Cohen, 1984). Further study showed that visual system can exhibit inhibition at several previously attended locations over a short time (Snyder & Kingstone, 2001), and the inhibition

is reduced when spatial VSTM is occupied by a concurrent task (Castel et al., 2003), hinting the role of VSTM in inhibition.

1.2.3 Visual Long-Term Memory

Besides the iconic memory and the visual short-term memory, long-term memory plays an important role in constructing a visual representation of natural scenes (Hollingworth et al., 2004). It is obvious that human visual system has a long-term memory, but its interference with scene representation has had some doubts. For example, Irwin and Zelinsky (Irwin & Zelinsky, 2002) proposed that higher level visual representations of previously attended objects get stored in VSTM as the eye moves from fixation to fixations according to attention. However, from Hollingworth and Henderson's experiment (Hollingworth & Henderson, 2002), it was shown that more than eight objects (which exceeds the known capacity of VSTM (Luck, 2007)) can be retained in visual memory over relatively long periods of time, showing the possibility of long-term memory component in online scene representation. In Hollingworth's work (Hollingworth et al., 2004), more evidence is provided that VSTM and VLTM both contribute to the online scene representation.

A general assumption is that VLTM lacks in detail and holds only the gist of a certain object (Brainerd & Reyna, 2005). It is well known from Standing's work (Standing, 1973) that VLTM can store massive number of items. In his work, he demonstrated that after viewing 10,000 scenes for a few seconds each, people could distinguish which of the two images was in the 10,000 with 83% accuracy (Standing, 1973). However, this experiment did not provide insights on whether only the gist of the scene was stored in VLTM, or the fine details were also stored. Hollingworth (Hollingworth et al., 2004) showed that when requiring memory for more than hundred objects, observers remain significantly above chance at

remembering which exemplar of an object they have seen. This result hinted that VLTm is capable of storing detailed representation of observed objects. Hollingworth also argued that shares similar visual representations as VSTM and demonstrated that the capacity of VLTm is not exhausted by retention of visual properties of hundreds of objects (Hollingworth, 2005).

Contrary to the general assumption, recent work by Brady et al. (Brady et al., 2008) showed that VLTm is capable of storing a massive number of objects with details from the image. In their experiment, participants viewed pictures of 2,500 objects over the course of 5.5 hours, which then the participants were to choose which of the two test images they had seen. The previously viewed item was paired with either an object from a novel category, an object from the same basic-level category (*e.g.* mirrors, starfish), or the same object in different state or pose. The results from these tests were remarkably high, giving evidence to the theory that VLTm holds detailed representation of the scene. They further examine the results by investigating the lower bound of VLTm in the traditional information theory concept and showed that the estimated memory capacity grows massively according to the size of the representation stored in VLTm. This implies that human VLTm has incredibly large storage with high fidelity.

1.3 Bio-mimetic Framework for Detection and Tracking

Inspired from the human vision perception model, we tackle the problem of detecting and tracking moving objects through bio-mimetic models. We represent the scene with a temporary short-term model (TSTM) and a conservative long-term model (CLTM). The TSTM exists to model the quick changes in the scene and to connect the observations from multiple time instances. The underlying

philosophy for TSTM is that through whatever method, we would like to establish a model which is able to describe the scene in a short-term and bottom-up driven way. Thus, by utilizing TSTM, the entire modeling becomes robust against such short-term noises and changes. An example of TSTM in our study can be a conventional background model with a relatively fast learning rate. In this case, the background model can be used to find quick changes in the scene, similar to a way VSTM can be used in human visual perception. Also, TSTM is not restricted to such background models only. Rather, TSTM can be in a totally different form such as the distribution of optical flows for a single shot of the scene.

CLTM on the other hand, exists to model the long-term information required to achieve a given visual inference task. Again, the philosophy here is that through whatever method, we want to have a top-down driven long-term modeling component solely designed for the task at hand. For object tracking, CLTM can be the target model in the traditional sense. Though it is only about one object in case of tracking single object, the target model needs to be effective throughout the whole given sequence, which can be relatively long to be dealt with short-term models. In case of multiple object tracking, CLTM, which is composed of models for multiple objects, plays the role of VLTM in case of human visual perception. Also, the use of the concept of CLTM is not restricted to object tracking. In case of object recognition, the models used for the recognition task are actually CLTMs. However, in this thesis, we focus on using TSTM and CLTM for the purpose of motion detection and object tracking.

1.4 Contents of the Research

The thesis is mainly focused on developing bio-mimetic models for solving the issues addressed in Section 1.1. Figure 1.2 is a summary of the contents and their

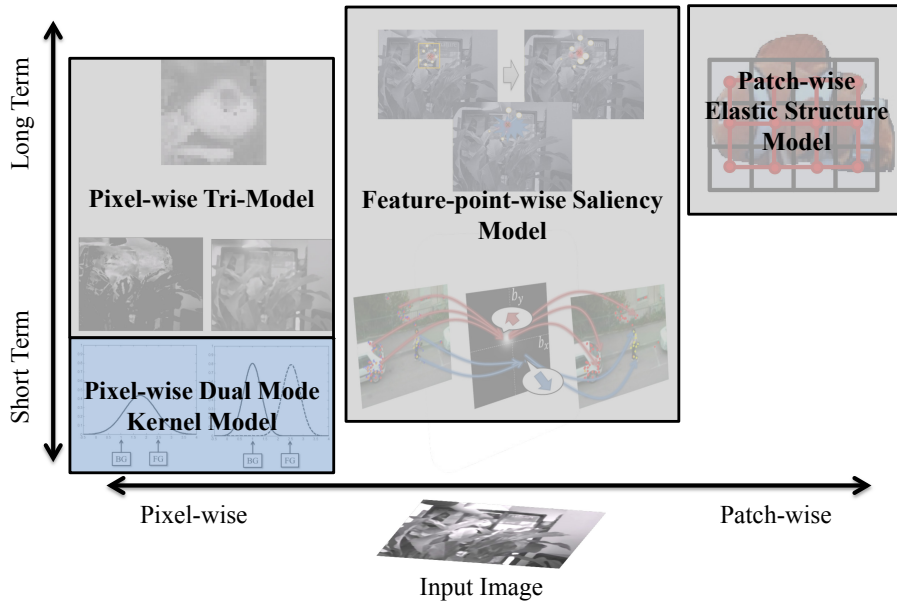


Figure 1.2: Contents of the research and their relatedness to short-term/long-term memory and how data is treated (from pixel-wise to patch-wise).

relatedness to short-term/long-term memory and how data is treated (from pixel-wise to patch-wise). Firstly, to build a robust motion detection scheme capable of running real-time even on a mobile device, we propose an efficient pixel-wise background model based on the concept of TSTM (bottom-left on Figure 1.2). The current state-of-the-art methods used for such purpose are mostly with much computation load (Sheikh et al., 2009; Kwak et al., 2011), limiting the applicability of these methods greatly. Although Kim et al. (Kim et al., 2013) provide real-time performance, for a detection method to be used with other higher-level inference methods, the computation load can still be of burden. To achieve real-time capability with satisfying performance, the proposed method models the background through dual-mode kernel model (DMKM) and compensates the motion of the camera by mixing neighboring models. Modeling through DMKM prevents the background model from being contaminated by foreground pixels,

while still allowing the model to be able to adapt to changes of the background. Mixing neighboring models reduces the errors arising from motion compensation and their influences are further reduced by keeping the age of the model. The details about our scheme is described in Chapter 2.

Next, to deal with the issues related to drifting, background clutter, occlusions, and the limitation of bounding box representation in tracking, we propose a tracking scheme using both pixel-wise TSTM and CLTM (mid to top-left in Figure 1.2). The proposed scheme employs a novel tri-model to represent the scene while tracking. The tri-model consists of the target object model (CLTM), the background model (TSTM), and the foreground model (TSTM). The target object model learns the object of interest as CLTM, the background model learns the stationary parts of the scene in TSTM sense, and the foreground model learns moving objects in the scene other than the target object. The proposed scheme performs tracking by finding the best explanation of the scene with the three learned models. By utilizing the information in the background and the foreground models as well as the target object model, our scheme is robust against occlusions, background clutters, and drifting. Furthermore, our scheme is not restricted to bounding-boxes when representing the target object, and is able to give pixel-wise tracking results. The scheme is explained in detail in Chapter 3.

In Chapter 4, to resolve the initialization sensitivity issue as well as the occlusion issue, we go beyond the pixel-wise representation and apply TSTM and CLTM using local features (center in Figure 1.2). The instantaneous velocities of local features are modeled as TSTM and used to obtain saliencies related to motions. The importance of local features when tracking are learned as CLTM and used to obtain saliencies related to descriptors. The two saliencies are combined through Hough voting to obtain promising tracking results even under occlusions and clumsy initializations.

Lastly, to deal with non-rigid deformations and occlusions simultaneously, we focus on building robust representation for CLTM (top-right in Figure 1.2). We go beyond the pixel-wise or local feature-wise representations and use local patches. Moreover, we consider the relationship among neighboring patches as well by modeling the target object as an elastic structure of local patches. This modeling allows more accurate description of the target object giving robust performances in case of tracking objects showing non-rigid deformations and partial occlusions. We also propose a hierarchical diffusion scheme to efficiently use this model. This scheme is presented in Chapter 5. In Chapter 6, we offer some broad closing remarks as well as future research directions.

Chapter 2

Detection by Pixel-wise Dual-Mode Kernel Model

In this chapter, we focus on the computation efficiency problem of motion detection algorithms. Thus, we propose an efficient pixel-wise representation for TSTM to be used for detecting moving objects on a non-stationary camera in real-time. Our motion detection scheme is not only aimed to work in real-time for PC environments, but to work in real-time for mobile devices as well. Our background model is designed in a way that minimizes computational requirements while still showing robust detection performances. The motion compensation is performed in a way so that not much warping computation is required, and the model tries to learn compensation errors within the model itself. Experimental results show that our method requires average of **5.8** milliseconds to run for a 320×240 image sequence on a desktop PC, with acceptable detection performance compared to other state-of-the-art methods. Also, our implementation of the proposed method on a smart phone is able to run in real-time.



Figure 2.1: Example of our method running on a mobile device (foreground pixels highlighted in red). Our implementation runs approximately 20 frames per second, comfortably in real-time.

2.1 Proposed Method

The proposed method consists of three major parts; pre-processing to reduce noise, background modeling through the dual-mode kernel model (DMKM) with age, and motion compensation for the background movements by mixing models. Figure 2.2 is an illustration of the framework. Pre-processing on the image is performed with simple spatial Gaussian filtering and median filtering on the image. To reduce the amount of computation required, the number of DMKMs used is smaller than the number of pixels in the input frame, meaning that the same model can be used for multiple pixels. The motion compensation is performed in a simple manner, with traditional KLT (Tomasi & Kanade, 1991). However, rather than *moving* the background model to its correct positions, similar to (Kwak et al., 2011), we mix the models from the previous frame to construct a

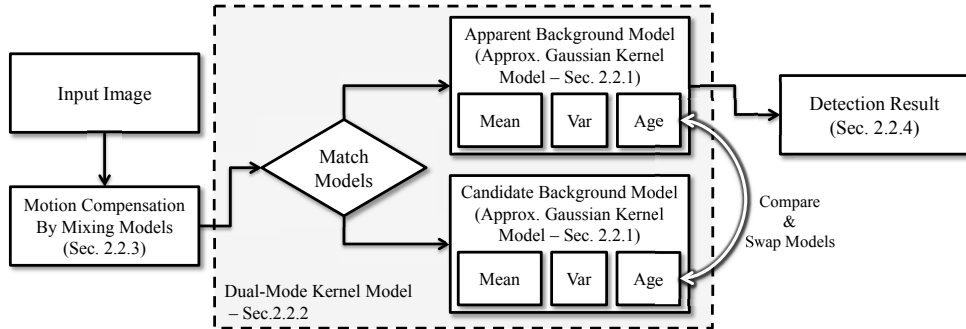


Figure 2.2: Framework of the proposed method

model for the present frame. Finally, we obtain the detection results using the trained model.

2.1.1 Approximated Gaussian Kernel Model

One of the main reasons for background subtraction methods with statistical models failing for non-stationary cameras is that they usually have a fixed learning rate. Having a fixed learning rate means that for a pixel, the first observation of the pixel is being considered as the mean of an infinitely learned model. This does not cause critical problems for stationary camera since for many cases the pixel value of a certain pixel does not change much for background pixels. However, for non-stationary cameras, motion compensation errors are apt to exist no matter how accurate the compensation is, and we cannot assume that the first observation of a pixel would be similar to the mean value we would actually get by acquiring further observations. Therefore, we need a varying learning rate. Although the authors of (KaewTrakulPong & Bowden, 2003) thought it to be a problem with initialization and fast adaptation, the notion of constructing a model with expected sufficient statistics works nicely for non-stationary cameras as well. In (Kim et al., 2013), the authors also use the *age* of a pixel to define

a variable learning rate, which is actually the same as using expected sufficient statistics.

To model the scene, we use a newly proposed kernel model which can be understood as an approximated version of SGM. Similar to using SGMs, with the notion of sufficient statistics to use only the observed data to form a model, as in (Kim et al., 2013) we keep the age of kernel model as well as the mean and the variance. Also, to reduce the computation load, we divide the input image into equal grids of size $N \times N$ and keep one kernel model for each grid. Here, to apply a single model to all pixels inside the grid, our kernel model needs to account for all statistics inside the grid. In case of the mean, we just need to observe the mean value of the input observations inside the grid. However, in the case of the variance, to get an exact update, we would need to obtain the variance of the observations inside the grid and as well as complex update equations compared to conventional SGM. This would harm the benefits of using grid representations, and therefore we approximate the variance calculation in a simple way without significant performance degradation.

If we denote the group of pixels in grid i at time t as $\mathbf{G}_i^{(t)}$, the number of pixels in $\mathbf{G}_i^{(t)}$ as $|\mathbf{G}_i^{(t)}|$, and the observed pixel intensity of a pixel j at time t as $I_j^{(t)}$, then the mean $\mu_i^{(t)}$, the variance $\sigma_i^2(t)$, and the age $\alpha_i^{(t)}$ of the kernel model applied to $\mathbf{G}_i^{(t)}$ is updated as

$$\mu_i^{(t)} = \frac{\tilde{\alpha}_i^{(t-1)}}{\tilde{\alpha}_i^{(t-1)} + 1} \tilde{\mu}_i^{(t-1)} + \frac{1}{\tilde{\alpha}_i^{(t-1)} + 1} M_i^{(t)} \quad (2.1)$$

$$\sigma_i^2(t) = \frac{\tilde{\alpha}_i^{(t-1)}}{\tilde{\alpha}_i^{(t-1)} + 1} \tilde{\sigma}_i^2(t-1) + \frac{1}{\tilde{\alpha}_i^{(t-1)} + 1} V_i^{(t)} \quad (2.2)$$

$$\alpha_i^{(t)} = \tilde{\alpha}_i^{(t-1)} + 1 \quad (2.3)$$

where $M_i^{(t)}$ and $V_i^{(t)}$ are defined as

$$M_i^{(t)} = \frac{1}{|\mathbf{G}_i|} \sum_{j \in \mathbf{G}_i} I_j^{(t)} \quad (2.4)$$

$$V_i^{(t)} = \max_{j \in \mathbf{G}_i} \left(\mu_i^{(t)} - I_j^{(t)} \right)^2 \quad (2.5)$$

and $\tilde{\mu}_i^{(t-1)}$, $\tilde{\sigma}_i^{2(t-1)}$, and $\tilde{\alpha}_i^{(t-1)}$ denote the kernel model of time $t-1$ compensated for use in time t which will be discussed in detail at Section 2.1.3. Here, the update equations regarding the mean (2.1) and (2.4) are the same as conventional SGMs. However, in the case of the variance in (2.2) and (2.5), the equations are rough approximations. To be exact, (2.2) should be the weighted mixing equations for Gaussian distributions and (2.5) should be the observation variance. Calculating these exact equations would actually negate the reduction of computation load introduced from using grids. Therefore, we take advantage of the fact that the kernel model is going to be used to find outliers (foreground pixels) and approximate the model using the observation which is furthest away from the mean as in (2.5). This approximated model is no longer a Gaussian model, but the model is still in a radial kernel form, similar to the Gaussian. The advantage of the proposed kernel model for grids is that it allows the motion compensation errors to be learned in the model properly with a variable learning rate based on the age of a model. Also, having the number of kernel models less than the number of pixels reduces computation load.

2.1.2 Dual-Mode Kernel Model (DMKM)

Using the proposed kernel model to model the scene usually works well in simple cases, however, when fast learning rates are used, the kernel model suffers from getting contaminated with the data coming from foreground pixels. In our

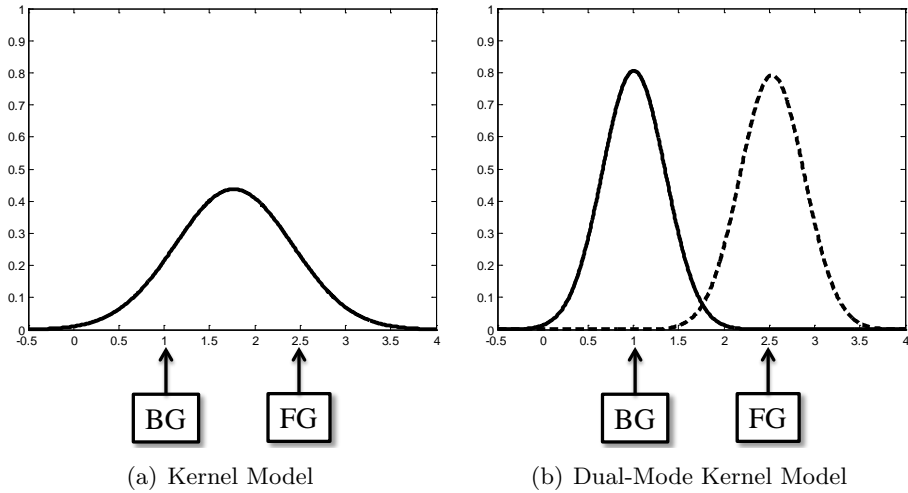


Figure 2.3: Illustration of the effects of using dual-mode kernel model. Learning results of (a) kernel model and (b) dual-mode kernel model with the same data. In (b), the solid line denotes the apparent background model whereas the dotted line denotes the candidate background model.

method, since we have a variable learning rate, having a fast learning rate is a common case. For example at initialization, all pixels start with age of one, meaning that the learning rate of these pixels at next frame would be 0.5. As illustrated in Figure 2.3 (a), the fast learning rate causes the background model to describe some portion of the foreground as well. This can be seen easily in the case of large objects passing through the scene. A naive solution to this problem would be to update with only the pixels determined as the background. However, in this case, a single misclassification of pixel would have an everlasting effect on the model since false foregrounds would never be learned.

To overcome this defect, we use another kernel model which acts like a candidate background model. The candidate background model remains ineffective until its age becomes older than the apparent background model, when, at that time, the two models are swapped. This dual-mode kernel model is different from Gaussian mixture models (GMM) (Stauffer & Grimson, 1999) with two modals,

considering the fact that using a bi-modal GMM would still have the foreground data contaminating the background whereas our method does not. If we denote the mean, variance, and age of the candidate background model and the apparent background model at time t for grid i as $\mu_{C,i}^{(t)}$, $\sigma_{C,i}^2(t)$, and $\alpha_{C,i}^{(t)}$, and $\mu_{A,i}^{(t)}$, $\sigma_{A,i}^2(t)$, and $\alpha_{A,i}^{(t)}$, respectively, then, if the squared difference between the observed mean $M_i^{(t)}$ and $\mu_{A,i}^{(t)}$ is less than a threshold with respect to $\sigma_{A,i}^2(t)$, *i.e.*

$$\left(M_i^{(t)} - \mu_{A,i}^{(t)}\right)^2 < \theta_s \sigma_{A,i}^2(t), \quad (2.6)$$

we update $\mu_{A,i}^{(t)}$, $\sigma_{A,i}^2(t)$, and $\alpha_{A,i}^{(t)}$ according to (2.1), (2.2), and (2.3), where θ_s is a threshold parameter. Also if the above condition does not hold and if the observed mean matches the candidate background model,

$$\left(M_i^{(t)} - \mu_{C,i}^{(t)}\right)^2 < \theta_s \sigma_{C,i}^2(t), \quad (2.7)$$

then we update $\mu_{C,i}^{(t)}$, $\sigma_{C,i}^2(t)$, and $\alpha_{C,i}^{(t)}$ according to (2.1), (2.2), and (2.3). If none of the conditions hold, we initialize the candidate background model with the current observation. Also, the two kernel models for grid i are swapped if the age of the candidate exceeds the apparent meaning,

$$\alpha_{C,i}^{(t)} > \alpha_{A,i}^{(t)}. \quad (2.8)$$

The candidate background model is initialized after swapping. Finally, we only use the apparent background model, which is now an uncontaminated pure background model, when determining foreground pixels in Section 2.1.4. Through the dual-mode kernel model, we can prevent our background model from being corrupted by the foreground data. As in Figure 2.3 (b), the foreground data is learned by the candidate background model rather than the apparent background

model. Also, we do not have to worry about false foregrounds never being learned into the model since if the age of the candidate background model becomes larger than the apparent background model, the models will be swapped and correct kernel model will be used.

2.1.3 Motion Compensation by Mixing Models

For image sequences obtained from a non-stationary camera, the model learned until time $t - 1$ cannot be used directly for detection in time t . To use the model, motion compensation is required. However, since we use a single model for all the pixels inside a grid (*i.e.* a single model for all j such that $j \in \mathbf{G}_i^{(t)}$), simple warping on the background model based on interpolation strategies would cause too much error. Thus, instead of simply warping the kernel model, we construct the compensated kernel model at time t by merging the statistics of the model at time $t - 1$. For obtaining the background motion, we divide the input image at time t into 32×24 grids, and perform KLT (Tomasi & Kanade, 1991) on every corner of the grid with the image from time $t - 1$. With these point tracking results, we perform RANSAC (Fischler & Bolles, 1981) to obtain a homography matrix $\mathbf{H}_{t:t-1}$ which warps all pixels in time t to pixels in time $t - 1$ through a perspective transform. We consider this to be the movement of the background. For further explanation, we will denote the position of pixel j as \mathbf{x}_j , the position of the center of gravity for $\mathbf{G}_i^{(t)}$ as $\bar{\mathbf{x}}_i^{(t)}$, and the perspective transform of \mathbf{x} according to $\mathbf{H}_{t:t-1}$ as $f_{PT}(\mathbf{x}, \mathbf{H}_{t:t-1})$.

As in Figure 2.4, for each grid i , we consider $\bar{\mathbf{x}}_i^{(t)}$ to have moved from $f_{PT}(\bar{\mathbf{x}}_i^{(t)}, \mathbf{H}_{t:t-1})$. Then, assuming that there was no significant scale change, if we think of an $N \times N$ square region \mathbf{R}_i having $f_{PT}(\bar{\mathbf{x}}_i^{(t)}, \mathbf{H}_{t:t-1})$ as the center, the model at time $t - 1$ for \mathbf{R}_i would be the motion compensated kernel model for pixels $\mathbf{G}_i^{(t)}$. \mathbf{R}_i overlaps with multiple grids and we mix the kernel models

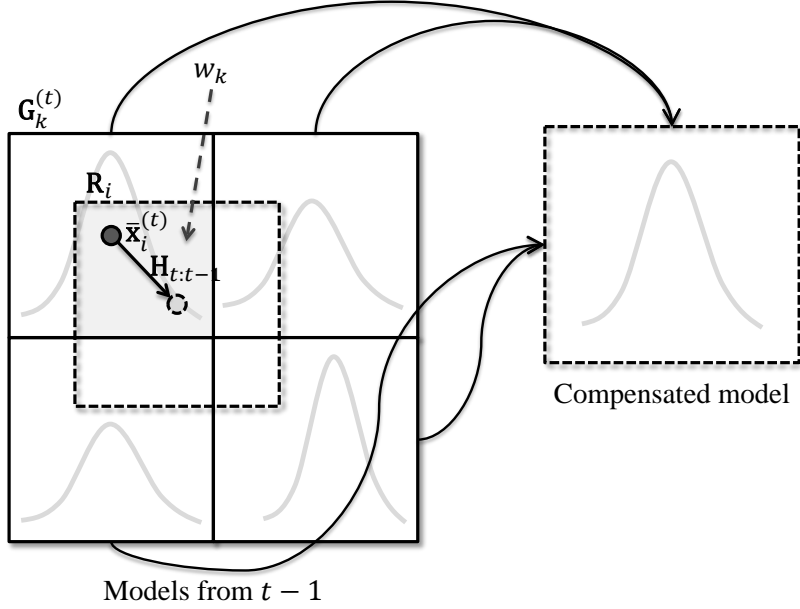


Figure 2.4: Illustration of the proposed motion compensation by mixing models. Models of overlapping regions are mixed together to a single model.

at time $t - 1$ of these overlapping grids to obtain $\tilde{\mu}_i^{(t-1)}$, $\tilde{\sigma}_i^{2(t-1)}$, and $\tilde{\alpha}_i^{(t-1)}$ of the motion compensated kernel model. The mixture weights are assigned so that they are proportional to the overlapping area. If we denote the group of grids overlapping with \mathbf{R}_i as $\mathbf{O}_i^{(t)}$, and the mixture weights as w_k where $k \in \mathbf{O}_i^{(t)}$, then we obtain the compensated kernel model simply by mixing the Gaussian models

$$\tilde{\mu}_i^{(t-1)} = \sum_{k \in \mathbf{O}_i^{(t)}} w_k \mu_k^{(t-1)} \quad (2.9)$$

$$\tilde{\sigma}_i^{2(t-1)} = \sum_{k \in \mathbf{O}_i^{(t)}} w_k \left[\sigma_k^{2(t-1)} + \left\{ \mu_k^{(t-1)} \right\}^2 - \left\{ \tilde{\mu}_i^{(t-1)} \right\}^2 \right] \quad (2.10)$$

$$\tilde{\alpha}_i^{(t-1)} = \sum_{k \in \mathbf{O}_i^{(t)}} w_k \alpha_k^{(t-1)}, \quad (2.11)$$

where the mixing weights are

$$w_k \propto \text{Area} \left\{ \mathbf{R}_i \cap \mathbf{G}_k^{(t)} \right\} \quad (2.12)$$

and

$$\sum_k w_k = 1. \quad (2.13)$$

Note that in (2.10) $\left\{ \mu_k^{(t-1)} \right\}^2 - \left\{ \tilde{\mu}_i^{(t-1)} \right\}^2$ exists, since when we merge multiple Gaussian models into one, it is the expectation for the square of the observation that is merged with weight w_k and not the variance itself. During the mixing process, as in Figure 2.4, models where nearby regions differ a lot (*e.g.* edges) will have excessively large variances after compensation (as in (2.10)). Normally, a kernel model would not have too large variances, and having such large variance would mean that the model has not learned much of the target object. Therefore, after compensation, if the variance is over a threshold θ_v , *i.e.* $\tilde{\sigma}_i^2{}^{(t-1)} > \theta_v$, we reduce the age of the model as

$$\tilde{\alpha}_i^{(t-1)} \leftarrow \tilde{\alpha}_i^{(t-1)} \exp \left\{ -\lambda \left(\tilde{\sigma}_i^2{}^{(t-1)} - \theta_v \right) \right\}, \quad (2.14)$$

where λ is a decaying parameter. This decaying of age prevents the model from having a model with too large variance not having any significant meaning near edges, in case of consistent camera movements.

2.1.4 Detection of Foreground Pixels

After obtaining the kernel model for time t as in Section 2.1.1, 2.1.2, and 2.1.3, we select pixels that have distances larger than threshold from the mean as foreground pixels. This is not an exact solution to the problem theoretically, since to be exact, we should find pixels with lower probability of being the background

with respect to the learned kernel model. However, finding such pixels require much computation, due to the square-root and natural logarithms operations in the exact equation (Stauffer & Grimson, 1999). We found empirically that the results are useable even with simple thresholding with respect to the variance, without complicated computation. Mathematically, for each pixel j in group i , we classify the pixel as a foreground pixel if

$$\left(I_j^{(t)} - \mu_{A,i}^{(t)}\right)^2 > \theta_d \sigma_{A,i}^2{}^{(t)}, \quad (2.15)$$

where θ_d is a threshold parameter. Note that we only use the apparent background model in determining the foreground. Through this way, we can avoid false negatives arising from contamination of the kernel model.

2.2 Experimental Results

For the experiments, the proposed method was implemented using C++ with the KLT from OpenCV¹ library. For the parameters, the grid size $N = 4$, the threshold for matching $\theta_s = 2$, the decaying parameter for age $\lambda = 0.001$, the threshold for decaying age $\theta_v = 50 \times 50$, and the threshold for determining detection $\theta_d = 4$. For the initialization of the variance, we simply set variance to be a moderate value (*e.g.* 20×20). The age was truncated at 30 to keep a minimum learning rate. The method was experimented with eleven image sequences, where six are identical to the sequences used in the work by Kim et al. (Kim et al., 2013), two are provided by Kwak et al. (Kwak et al., 2011), one from the work by Godec et al. (Godec et al., 2013), and others are newly made datasets (Yi et al., 2013). To evaluate the effectiveness of the proposed method, the method is compared against three other methods by Barnich and Van Droogenbroeck

¹<http://opencv.com/downloads.html>

Method	Pre-Processing	Motion Comp.	Modeling	Post-Processing	Total
Proposed with $N = 4$ and OpenMP	0.82ms	2.00ms	0.88ms	-	3.71ms
Proposed with $N = 4$	0.84ms	2.65ms	2.31ms	-	5.81ms
Proposed with $N = 1$	0.84ms	18.72ms	6.40ms	-	25.96ms
ViBe with simple motion comp.	-	7.20ms	4.15ms	-	11.35ms
ViBe with proposed motion comp.	-	39.22ms	3.64ms	-	42.86ms
Kim et al. (2013)	1.15ms	5.90ms	2.32ms	7.85	17.86ms
Kwak et al. (2011)	NA	NA	NA	NA	> 30s

Table 2.1: Average computation time for each method. “NA”: Not available, “-”: No such process.

(ViBe) (Barnich & Van Droogenbroeck, 2011), by Kim et al. (Kim et al., 2013), and by Kwak et al. (Kwak et al., 2011). ViBe (Barnich & Van Droogenbroeck, 2011) was implemented with the pseudo code provided the authors and simple warping of the background was applied based on the $\mathbf{H}_{t:t-1}$ in Section 2.1.3. For the method by Kim et al. and Kwak et al., the implementations were provided by the authors. However, for Kwak et al.’s method, for the sequences which were not included in their implementation, it was not possible obtain the preliminary labeling and therefore the ground truth label was used as the initial labeling at the first frame, and the result of the method proposed in this paper was used as preliminary labeling for other frames. Some critical frames are shown in Figure 2.5.

2.2.1 Runtime Comparison

To demonstrate the computational efficiency of the proposed method, computation time required for each major steps were measured and compared to other state-of-the-art methods. Also, for comparison, the computation time for the proposed method with simple parallel processing, and with parameter setting of $N = 1$, which means that all pixels have their own dual-mode kernel model, were measured. All experiments were performed on a Dual-core 3.4GHz PC with 320×240 image sequences. Table 2.1 is the average runtime required for each

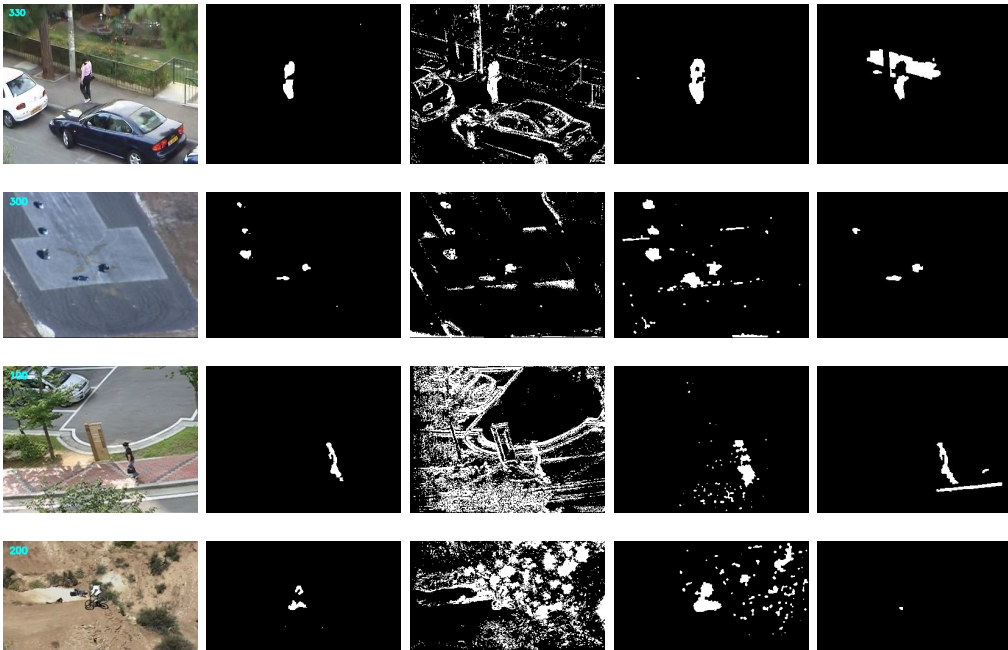


Figure 2.5: Example critical frames for each method. Input frames (first column), proposed method (second column), ViBe (Barnich & Van Droogenbroeck, 2011) with simple motion compensation (third column), method by Kim et al. (Kim et al., 2013) (fourth column), and method by Kwak et al. (Kwak et al., 2011) (last column). All datasets shown are non-commercial and publicly available (Kim et al., 2013; Kwak et al., 2011; Yi et al., 2013; Godec et al., 2013)

algorithm. As shown in the table, the proposed method outperforms other methods with respect to computation load. Even the proposed method with $N = 1$ runs 25.96ms in average, which assures real-time performance. The computation time of ViBe (Barnich & Van Droogenbroeck, 2011) is comparable to our method (11.35ms) but shows relatively poor performance considering the quality of detection results (discussed in detail in Section 2.2.2). Also, the influence of the computation load arising from motion compensation increases when a more precise motion compensation method, such as the proposed compensation method, is used rather than simple warping. The method by Kim et al. (Kim et al., 2013) also require computation time suitable for real-time performance on a PC, but is still computationally expensive to run on a machine with relatively less computation power. In case of Kwak et al. 's method (Kwak et al., 2011), which is a state-of-the-art method focusing on the detection performance, takes more than a 30 seconds per frame. The proposed method, however, can be further reduced to run 3.71ms on average with simple parallelization.

2.2.2 Qualitative Comparison

Figure 2.5 is an example of qualitative comparison among each compared methods. As shown in the third column of Figure 2.5, results of ViBe (Barnich & Van Droogenbroeck, 2011) with simple motion compensation have many false foregrounds. Most error arises near the edges, showing that simply using a method designed for stationary cameras in case of non-stationary cameras is not sufficient even with motion compensation. The proposed method (second column) generally outperforms or is comparable to the method proposed by Kim et al. (Kim et al., 2013) (fourth column) and the method by Kwak et al. (Kwak et al., 2011) (last column). As shown in the second row, in cases where the parts of the foreground is similar to the background, the proposed method does show false backgrounds.

Method	Cycle	Mtn. Bike	Football
Proposed with $N = 4$	0.53 / 0.80 / 0.64	0.72 / 0.26 / 0.39	0.74 / 0.35 / 0.47
Proposed with $N = 1$	0.26 / 0.90 / 0.40	0.57 / 0.63 / 0.60	0.54 / 0.63 / 0.58
ViBe with simple motion comp.	0.22 / 0.86 / 0.35	0.02 / 0.75 / 0.04	0.44 / 0.69 / 0.53
Kim et al. (2013)	0.71 / 0.88 / 0.79	0.13 / 0.69 / 0.22	0.70 / 0.84 / 0.76
Kwak et al. (2011)	0.78 / 0.88 / 0.82	0.54 / 0.12 / 0.20	0.89 / 0.83 / 0.86
Method	UAV	Woman	
Proposed with $N = 4$	0.68 / 0.34 / 0.46	0.36 / 0.56 / 0.46	
Proposed with $N = 1$	0.57 / 0.63 / 0.60	0.11 / 0.82 / 0.19	
ViBe with simple motion comp.	0.06 / 0.58 / 0.12	0.06 / 0.79 / 0.12	
Kim et al. (2013)	0.29 / 0.80 / 0.43	0.49 / 0.77 / 0.60	
Kwak et al. (2011)	0.66 / 0.34 / 0.45	0.19 / 0.55 / 0.28	

Table 2.2: Quantitative results for each method. Results are shown in the form [Precision / Recall / F-measure].

Still, the performance of the proposed method is acceptable even in such cases. Also, for this sequence, Kwak et al. ’s method (last column) fails to detect newly appearing cars on the top, whereas the proposed method succeeds.

2.2.3 Quantitative Comparison

To evaluate the proposed method quantitatively, pixel-wise precision, recall, and F-measure for five sequences were measured. The **Football** and **Cycle** sequences are provided by Kwak et al. (Kwak et al., 2011) with the results from the author’s exact implementation. The **UAV** and **Woman** sequence is from the work by Kim et al. (Kim et al., 2013), and **Mountain Bike** (Mtn. Bike) is from the work by Godec et al. (Godec et al., 2013). For the **UAV** sequence, the first 40 frames were cut out since the movements of objects in those frames were too small and Kwak et al.’s method (Kwak et al., 2011) was not well initialized with the results of the proposed method. Also, **Mtn. Bike** was resized to the half of its original size. The ground truth label was generated manually for every ten frames. As shown in Table 2.2, the proposed method shows good results in terms of precision (best results for **UAV** and **Mtn. Bike**, comparable results for others). In terms of recall, the proposed method tends to give results not as good as Kim et al. ’s

method (Kim et al., 2013). However, when both precision and recall are considered using the F-measure, the proposed method gives comparable results to the state-of-the-art. Results of the proposed method is not as good as Kwak et al.’s (Kwak et al., 2011) or Kim et al.’s (Kim et al., 2013) in case of **Cycle**, **Football**, and **Woman**, but is still acceptable (having F-measure above 0.4). In case of the other two, the proposed method outperforms all other methods. Furthermore, the proposed method requires significantly less computation load and does not perform any post-processing. Note that since the foreground model of Kwak et al.’s method (Kwak et al., 2011) tend to behave similar to tracking methods, the foreground model of their method suffer from drifting problem, which is why they give bad results for **Mtn. Bike** and **Woman**.

2.2.4 Effects of Dual-Mode Kernel Model

As noted in Section 2.1.2, using one kernel model causes the background model to be contaminated by the foreground. This causes problems when the scene contains objects that are relatively large or moving slowly, since they will contaminate the background significantly and affect the final detection result. An example of this is shown in Figure 2.6. In Figure 2.6 (a), it can be seen that traces of moving objects are left in the background model when using one kernel model, whereas in (b) and (c), with dual-mode kernel model, the traces are learned in the candidate background model (c) and the apparent background model (b) is preserved and clear. As in (d) and (e), this degradation of the background model decreases the performance of the detection algorithm.

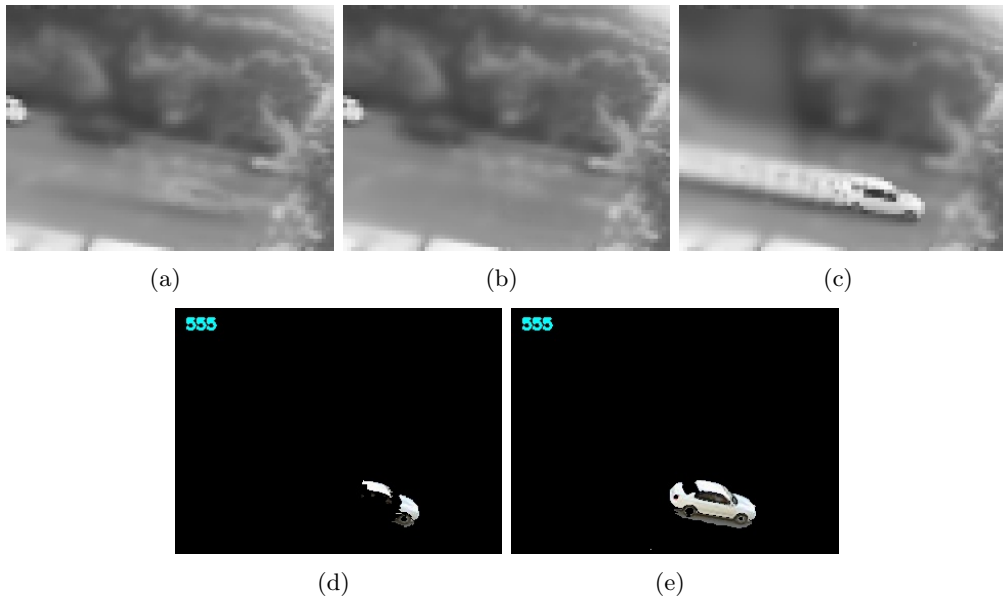


Figure 2.6: Example result with the kernel model and the dual-mode kernel model. (a) mean of the background model for the kernel model, (b) mean of the apparent background model of the dual-mode kernel model, (c) mean of the candidate background model of the dual-mode kernel model, (d) the kernel model result, and (e) the dual-mode kernel model result. The mean of the background model of the kernel model (a) is contaminated by the foreground, whereas the mean of the apparent background model of dual-mode kernel model (b) remains unharmed. This leads to different results as in (d) and (e).

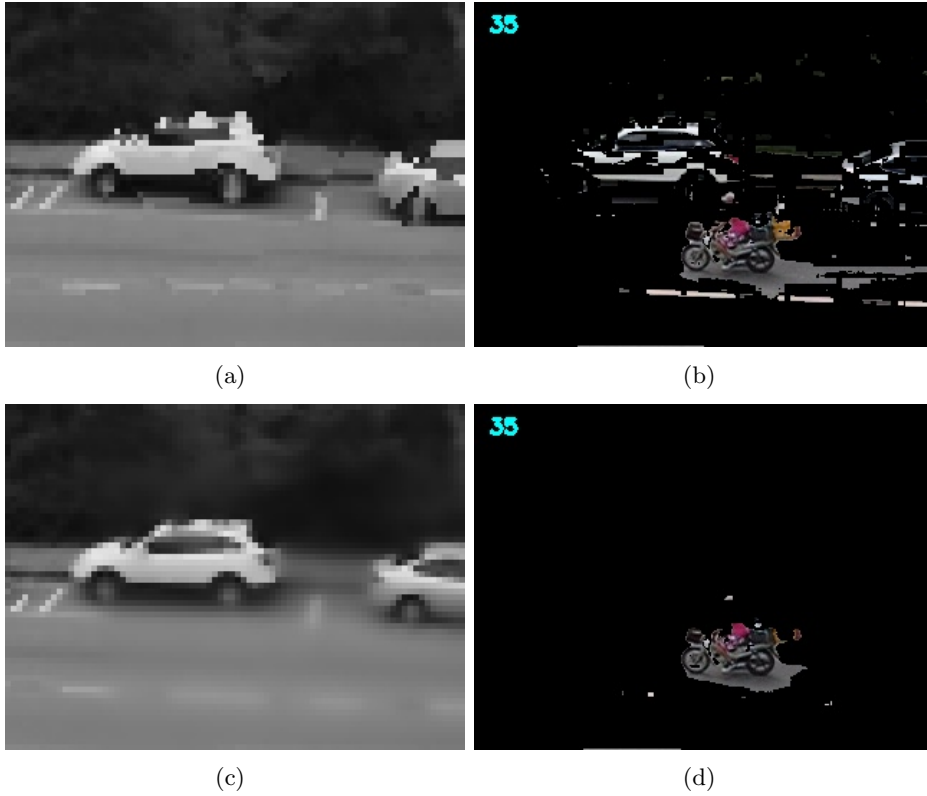


Figure 2.7: Example results when simple nearest neighbor warping is performed ((a) and (b)) and when the proposed motion compensation by mixing models is performed ((c) and (d)). (a) and (c) are the means of the apparent background model, and (b) and (d) are the detection results.

2.2.5 Effects of Motion Compensation

Since we use the same dual-mode kernel model for multiple pixels inside a grid, the motion compensation method proposed in Section 2.1.3 plays a critical role for the performance of the whole algorithm. If we simply apply a nearest neighbor warping to *move* the pixels, even the kernel model is not able to cope with such compensation errors. Figure 2.7 is an example showing the effectiveness of the proposed compensation scheme. In the case of simple nearest neighbor warping, as shown in (a), the background model becomes distorted due to quantization effects. This results in bad detection performance as shown in (b). However, with the proposed motion compensation scheme, the background motion is well compensated as in (c). Also, as in (d), detection performance is unharmed.

2.2.6 Mobile Results

We have also implemented our method on a mobile device to further test its real-time capability. The implementation was done on an Quad-core 1.4 GHz Cortex-A9 android device with OpenCV for android² used to implement KLT. The implementation does not have any optimization techniques used and is basically the same code for PC tweaked so that it matches the android interface. Our method runs approximately 20 frames per second with the capture resolution set to 160×120 . Figure 2.8 shows actual still shots of our algorithm running on a mobile device. Although some small details are not detected due to low resolution, it is possible to see that our method performs well in real-time even on a mobile device.

²<http://opencv.org/platforms/android.html>



Figure 2.8: Example detection results of our method on a mobile device (foreground pixels highlighted in red). Our implementation runs approximately 20 frames per second, assuring real-time performance.

2.3 Remarks and Discussion

A novel computationally efficient scheme for detecting moving objects in a scene with non-stationary cameras was proposed. The proposed scheme modeled the background through DMKM to cope with motion compensation errors and to prevent the background model from being contaminated by the foreground. A single DMKM was applied to multiple pixels to reduce the required computation load, without performance degradation. To reduce errors arising from motion compensation, models were mixed together in the compensation process. Experimental results showed that our scheme requires significantly less amount of computation, running within **5.8ms**, and yet with robust detection performances. Also, our scheme was implemented on a mobile device, confirming its real-time capability.

Chapter 3

Tracking by Pixel-wise Tri-Model Representation

In this chapter, we focus on resolving the problems of drifting, background clutter, occlusions, and the limitation of bounding box representation in tracking simultaneously through a pixel-wise approach. Our method starts from the idea that if we can estimate which pixels of the observed image are of the target object and which are not, and apply these estimation results to the model updates and the estimation of the target object position, we can easily overcome these limitations. For example, in case of occlusions and background clutters, we can easily overcome those problems by disregarding the pixels in the scene which are from the background or from other moving objects during the tracking process. Moreover, disregarding those pixels during the update process would prevent drifting problems as well. To achieve this aim, we build three pixel-wise models which are dedicated in learning different aspects (the target, the background, and other moving objects) of the scene.

The significant difference of our method compared to previous tracking meth-

ods is that we not only focus on modeling the target in a robust way, but also focus on learning the background and other moving objects which may interfere with tracking. Thus, if occlusions or background clutters are present in the scene, the other two models can cope with them. The learning of the models are performed similar to traditional background subtraction methods (Stauffer & Grimson, 1999; KaewTrakulPong & Bowden, 2003). In fact, our framework combines background subtraction based detection with traditional tracking, benefiting from them both. Also, when estimating the position of the target object, our method simultaneously finds which model fit best for each pixel (*i.e.* pixel labels). As a result, we not only obtain the position of the target object, but also the pixel-wise estimation of the target object, which provides more information about the target than bounding box representations. These pixel labels are also used in the learning process of the three models to prevent drifting.

3.1 Tri-Model Framework

3.1.1 Overall Scheme

The proposed tracking scheme is formulated as a maximization problem with a newly defined tri-model likelihood, representing the degree of matching between the observation and the tri-model w.r.t. the hypothesized position of the target object. A key idea of our method is the way the tri-model likelihood is defined using the likelihoods of the three models; the target model (T), the background model (B), and the foreground model (F). Each model is represented using single Gaussian Models (SGM) and learns the probability of each pixel being a certain color, as shown in the example in Figure 3.1. For each pixel in the model, SGM learns the mean and the variance of the observed data (colors), and keeps track of how much data was observed as the age of the model. The age term is used to

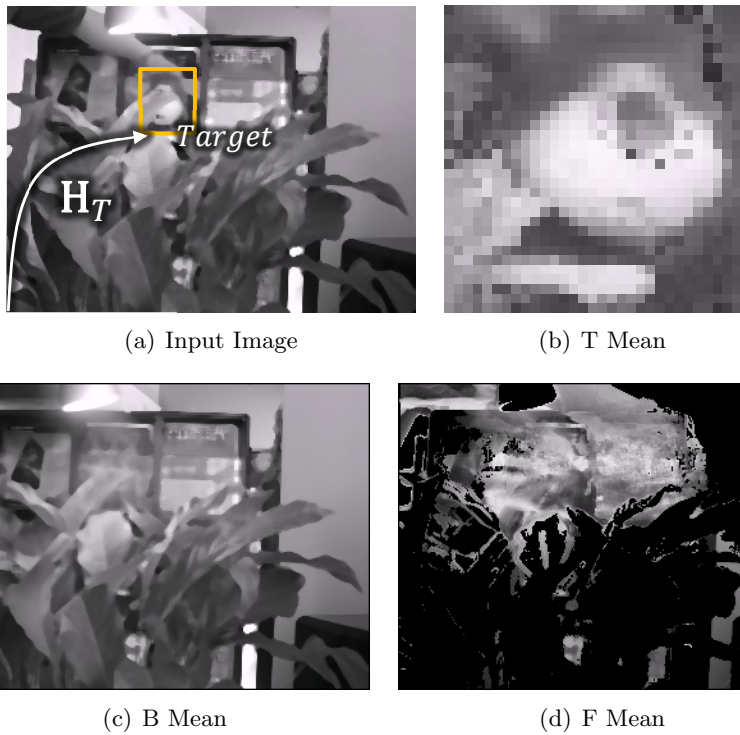


Figure 3.1: Example of each model. (a) The input image. The relationship between the target model and the input image denoted with \mathbf{H}_T . Target candidate region denoted with orange bounding box. (b) The mean values for the target model, (c) the background model, and (d) the foreground model.

derive the variable learning rate, which in turn is employed to use the estimated sufficient statistics for the SGM (KaewTrakulPong & Bowden, 2003). This allows appropriate adaptation for pixels which have only a few observed data (having small age). The target model is kept in a user-defined fixed size ($M_T \times N_T$, where M_T and N_T is the width and height of the target model) and the relationship between the target model and the input image is kept with a homography matrix \mathbf{H}_T (Example shown in Figure 3.1(a)). The other two models are in the same size as the input image ($M_I \times N_I$, where M_I and N_I is the width and height of the input image). For further explanation, we will denote the mean, the variance, and the age of the SGM with \mathbf{M} , \mathbf{V} , and \mathbf{A} , respectively. We will also use subscripts T , B , and F to denote the three models; the target, the background, and the foreground.

With the three models, we obtain the tri-model w.r.t. \mathbf{H}_T and then find $\hat{\mathbf{H}}_T$ which gives maximum tri-model likelihood. As the first step of calculating the tri-model likelihood, we match the observation to the three models in a pixel-wise manner. Then, the likelihood of the best fitting model for each pixel is used to obtain the overall tri-model likelihood. For a hypothesized \mathbf{H}_T , using log-likelihoods for ease in notation, if we denote the log-likelihood for the i^{th} pixel using the target model, the background model and the foreground model with $L_{T,i}(\mathbf{H}_T)$, $L_{B,i}$, and $L_{F,i}$, respectively (details on the likelihoods in Section 3.2.1), then the tri-model log-likelihood for the i^{th} pixel $L_{Tri,i}(\mathbf{H}_T)$ is defined so that

$$L_{Tri,i}(\mathbf{H}_T) = \max \{L_{T,i}(\mathbf{H}_T), L_{B,i}, L_{F,i}\}. \quad (3.1)$$

Then, since we are using SGM and assuming independence among pixels, the tri-model log-likelihood for the entire observed image is simply the sum of $L_{Tri,i}(\mathbf{H}_T)$

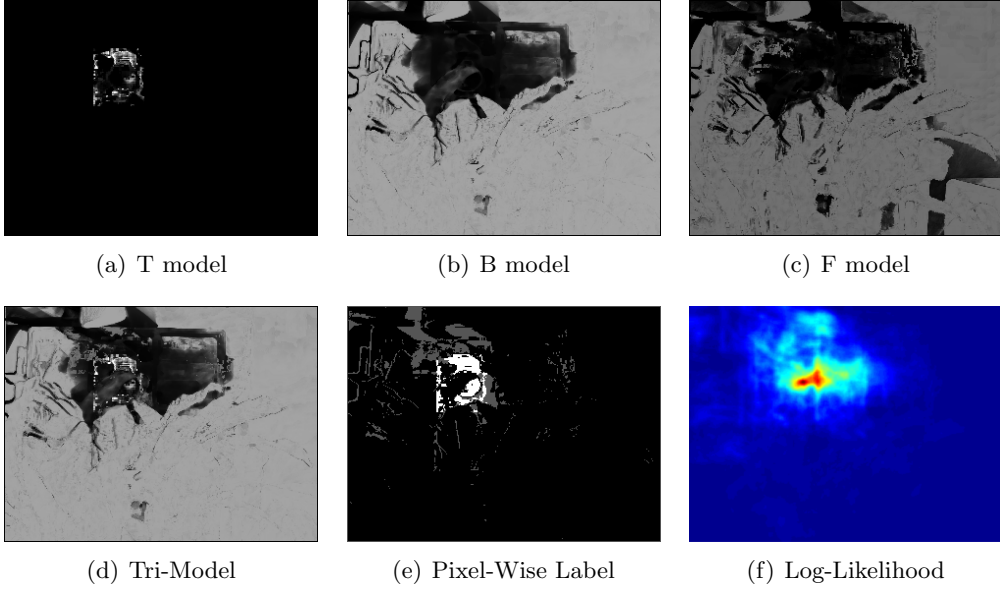


Figure 3.2: Example of the pixel-wise likelihood values (a) - (d) using the models and \mathbf{H}_T in Figure 3.1, and the selected model for each pixel (e). All results are in size $M_I \times N_I$. (f) is the tri-model log-likelihood w.r.t. the hypothesized position of the target object. In (e), white denotes T , gray denotes F , and black denotes B . In (f), red denotes high log-likelihood value and blue denotes low value. Note that for the target model in (a), the target model is warped to the target candidate region in Figure 3.1(a) with \mathbf{H}_T and then applied.

over all pixels. Thus, the tracking problem is to find the estimate $\hat{\mathbf{H}}_T$ so that

$$\hat{\mathbf{H}}_T = \arg \max_{\mathbf{H}_T} \sum_i^{M_I \times N_I} L_{Tri,i}(\mathbf{H}_T). \quad (3.2)$$

When estimating $\hat{\mathbf{H}}_T$, as a bi-product, we also obtain which model fits best for each pixel. During the calculation of $L_{Tri,i}(\hat{\mathbf{H}}_T)$, the estimated pixel-wise label $\hat{\mathbf{R}}_i$ can be obtained as

$$\hat{\mathbf{R}}_i = \arg \max_{T,B,F} \left\{ L_{T,i}(\hat{\mathbf{H}}_T), L_{B,i}, L_{F,i} \right\}. \quad (3.3)$$

Figure 3.2 shows examples of pixel-wise likelihood maps when applying the tri-model setting in Figure 3.1. With the hypothesized \mathbf{H}_T in Figure 3.1(a), pixel-wise likelihood maps, which indicate the probabilities of the observed colors of each pixel being generated from the models, are obtained as shown in Figures 3.2(a), 3.2(b), and 3.2(c). Then, we use the pixel-wise maximum among the three likelihood maps as the likelihood map of the tri-model, as shown in Figure 3.2(d). This means that with the three models, our tri-model likelihood is designed so that we try to best describe the given observation in a pixel-wise manner. When obtaining the estimate for \mathbf{H}_T by maximizing the tri-model likelihood (Figure 3.2(f)), we can also easily obtain pixel-wise labels by observing which of the three models fits best (Figure 3.2(e)). Details of the log-likelihoods for each model and the maximization process are presented in Section 3.2. Also, for clarification, from now on we will denote pixel indices with i when dealing with the pixels in the original observation dimension and with j when dealing with the pixels in the target model dimension.

3.1.2 Advantages

The proposed pixel-wise tri-modeling has distinct advantages. In our framework, occlusions and background clutters are naturally considered. With the proposed likelihood, we consider all three models in a pixel-wise manner when finding $\hat{\mathbf{H}}_T$. For each pixel, we choose the model which fits best, meaning that we simultaneously estimate pixel-wise labeling when likelihood maximization is performed. In other words, if there are occlusions or background clutters, we are considering them during the maximization process. For example, as shown in Figures 3.2(e) and 3.4(e), the leaf occluding the target object is described with the background model. As a result, the occluding leaves do not interfere with the estimation of $\hat{\mathbf{H}}_T$.

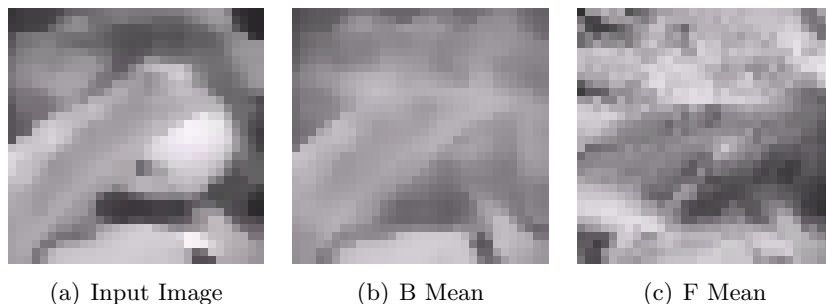


Figure 3.3: Example of each model and the observation warped into the target domain. The target object is a hand held tiger doll, which the left side of the tiger doll is partially occluded by a leaf of a plant in the background at the shown frame.

The proposed method is robust against drifting problems as well. Learning of the three models is performed considering estimated pixel labels. This prevents the target model from learning the background or other moving objects, reducing the chance of drifting. Again, as in the example shown in Figures 3.2(e) and 3.4(e), only the pixels estimated as the target object (denoted with white) is learned in the target model. In case of pixels with occlusions, the target models for those pixels remain unharmed. Moreover, the pixel-wise labeling further provides more accurate target object information to be used for higher-level inference tasks.

3.1.3 Practical Approximation

Unfortunately, an exact solution to the above problem formulation would require too much computation since all pixels need to be considered for all hypothesized \mathbf{H}_T . To reduce computation load, we take advantage of the fact that we are mostly interested on the target object, and the assumption that the target object moves little between consecutive frames. Instead of considering all pixels, we approximate by considering the pixel-wise likelihoods within the target candidate region, in the target model domain. As shown in the example in Figures 3.2(f),

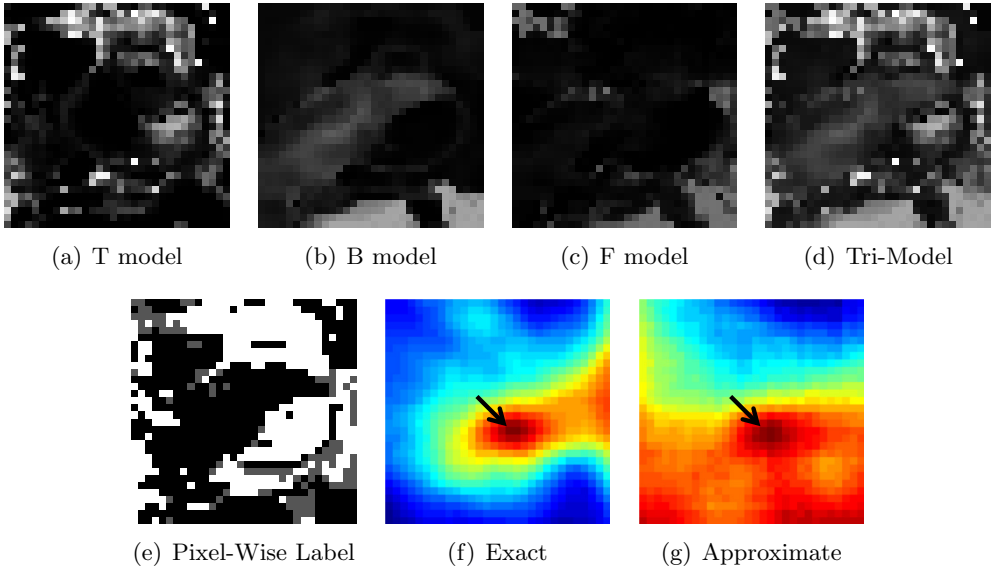


Figure 3.4: Example of the pixel-wise likelihood values in the target domain (a) - (d) using the models and \mathbf{H}_T in Figure 3.1, and the selected model for each pixel (e). All results are in size $M_T \times N_T$. In (e), white denotes T , gray denotes F , and black denotes B . In (f) and (g), red denotes high log-likelihood value and blue denotes low value. (f) is the exact tri-model log-likelihood in Figure 3.2(f) near the target object, and (g) is the approximated log-likelihood in the same region. Note that (f) and (g) share the same local maxima (marked with arrows).

3.4(f), and 3.4(g), this approximation shares the same local maxima around the target object. Thus, as in Figures 3.3 and 3.4, we warp the current input, the background model, and the foreground model according to \mathbf{H}_T^{-1} and apply them to obtain the likelihoods in the warped domain ($M_T \times N_T$). If we denote the pixel indices of the target model with j , where $j = 1, 2, \dots, M_T \times N_T$, the pixel-wise log-likelihood for the three models in the target domain as $\tilde{L}_{T,j}(\mathbf{H}_T)$, $\tilde{L}_{B,j}(\mathbf{H}_T)$, and $\tilde{L}_{F,j}(\mathbf{H}_T)$, then the pixel-wise log-likelihood of the tri-model in the target domain $\tilde{L}_{Tri,j}(\mathbf{H}_T)$ is defined as

$$\tilde{L}_{Tri,j}(\mathbf{H}_T) = \max \left\{ \tilde{L}_{T,j}(\mathbf{H}_T), \tilde{L}_{B,j}(\mathbf{H}_T), \tilde{L}_{F,j}(\mathbf{H}_T) \right\}. \quad (3.4)$$

With (3.4), we redefine $\hat{\mathbf{H}}_T$ in (3.2) as

$$\hat{\mathbf{H}}_T = \arg \max_{\mathbf{H}_T} \sum_j^{M_T \times N_T} \tilde{L}_{Tri,j}(\mathbf{H}_T). \quad (3.5)$$

3.2 Tracking with the Tri-Model

3.2.1 Likelihood of the Tri-Model

For a hypothesized \mathbf{H}_T , as in (3.4), we define the likelihood for each pixel as the maximum value among the pixel-wise likelihoods given by each model. This can be understood as if we are trying to reconstruct the observed image by selecting the best model for each pixel, and then using the negative of squared reconstruction error as the log-likelihood. As noted in Section 3.1.3, we consider this likelihood in the target domain for computational reasons. In other words, we warp the observation, the background model, and the foreground model with \mathbf{H}_T^{-1} , and then consider the likelihood. Then, if we let \mathbf{I} denote the observed data, and $\check{\mathbf{M}}_B, \check{\mathbf{V}}_B, \check{\mathbf{M}}_F, \check{\mathbf{V}}_F$, and $\check{\mathbf{I}}$ denote $\mathbf{M}_B, \mathbf{V}_B, \mathbf{M}_F, \mathbf{V}_F$, and \mathbf{I} warped

with \mathbf{H}_T^{-1} , respectively, the log-likelihoods of each model for pixel j in the target domain, $\tilde{L}_{T,j}(\mathbf{H}_T)$, $\tilde{L}_{B,j}(\mathbf{H}_T)$, and $\tilde{L}_{F,j}(\mathbf{H}_T)$ are defined as,

$$\tilde{L}_{T,j}(\mathbf{H}_T) = \mathcal{L}_N(\mathbf{M}_{T,j}, \mathbf{V}_{T,j}, \check{\mathbf{I}}_j) \quad (3.6)$$

$$\tilde{L}_{B,j}(\mathbf{H}_T) = \mathcal{L}_N(\check{\mathbf{M}}_{B,j}, \check{\mathbf{V}}_{B,j}, \check{\mathbf{I}}_j), \quad (3.7)$$

$$\tilde{L}_{F,j}(\mathbf{H}_T) = \mathcal{L}_N(\check{\mathbf{M}}_{F,j}, \check{\mathbf{V}}_{F,j}, \check{\mathbf{I}}_j), \quad (3.8)$$

where $\mathcal{L}_N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2, \mathbf{o})$ is the log-likelihood of a normal distribution with mean $\boldsymbol{\mu}$, variance $\boldsymbol{\sigma}^2$, and observation \mathbf{o} . Note that these log-likelihoods are log-likelihoods of SGM using the corresponding models with correct warping.

3.2.2 Likelihood Maximization

To find $\hat{\mathbf{H}}_T$ which maximizes the log-likelihood in (3.5), we perform an iterative process based on mean-shift (Comaniciu et al., 2003) and random sampling. The iterative process begins with random sampling of alterations on \mathbf{H}_T , then uses the samples to estimate the mean-shift vector. The mean-shift vector is in fact an estimate of the gradient (Comaniciu et al., 2003), thus iteratively updating \mathbf{H}_T with the mean-shift vector gives $\hat{\mathbf{H}}_T$. For further explanation, we will denote the alterations on \mathbf{H}_T with vector $\mathbf{h}^{[s]}$, where $s = 1, 2, \dots, S$ and S is the number of random samples used. Each dimension of vector $\mathbf{h}^{[s]}$ corresponds to a tracked dimension (e.g. translation in vertical direction, translation in horizontal direction, change in scale, etc.). For example, in case of translation only tracking, if we use the notation $g(\mathbf{H}_T, \mathbf{h}^{[s]})$ for the result of applying $\mathbf{h}^{[s]} = [dx^{[s]} \quad dy^{[s]}]^\top$ to \mathbf{H}_T , where $dx^{[s]}$ and $dy^{[s]}$ are scalar values denoting alterations in x and y directions

for sample s ,

$$g(\mathbf{H}_T, \mathbf{h}^{[s]}) = \begin{bmatrix} 1 & 0 & dx^{[s]} \\ 0 & 1 & dy^{[s]} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{H}_T. \quad (3.9)$$

To obtain the mean-shift vector, we first generate random samples using a Gaussian distribution for each tracked dimension. In our implementation, the tracked dimension is limited to translational movements only, but sampling can be done for any affine movement as long as a sufficient number of samples is provided. When considering translational movements only, $\mathbf{h}^{[s]} = [dx^{[s]} \quad dy^{[s]}]^\top$ is sampled as

$$\mathbf{h}^{[s]} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} (0.1M_T)^2 \\ (0.1N_T)^2 \end{bmatrix}\right), \quad (3.10)$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ denotes a Gaussian distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$. In the sampling process, we balance the sampling by first sampling $S/2$ samples and assigning $\mathbf{h}^{[s]} = -\mathbf{h}^{[S-s]}$ to the remaining half to ensure easy calculation of the mean-shift vector. If we denote the k^{th} element of \mathbf{h} as h_k , then the equation for obtaining the mean-shift vector $\Delta\mathbf{h}$ with the samples is

$$\Delta h_k = \frac{\sum_s h_k^{[s]} \exp\left\{\frac{\lambda}{M_T N_T} \sum_j \tilde{L}_{Tri,j}(g(\mathbf{H}_T, \mathbf{h}^{[s]}))\right\}}{\sum_s \exp\left\{\frac{\lambda}{M_T N_T} \sum_j \tilde{L}_{Tri,j}(g(\mathbf{H}_T, \mathbf{h}^{[s]}))\right\}}, \quad (3.11)$$

where λ is a parameter controlling the convergence speed.

If we apply (3.11) directly, we need to apply $g(\mathbf{H}_T, \mathbf{h}^{[s]})$ to all three models for every sample, which requires large amount of computation load. Thus, to ensure real-time performance, we approximate this process as in Section 3.1.3. Instead of applying $\mathbf{h}^{[s]}$ directly to \mathbf{H}_T and calculating $\tilde{L}_{Tri,j}(g(\mathbf{H}_T, \mathbf{h}^{[s]}))$ with

(3.6), (3.7), and (3.8), we first apply \mathbf{H}_T to the three models and apply $\mathbf{h}^{[s]}$ only to the target model. In other words, instead of warping the background model, the foreground model, and the input image according to the inverse of $g(\mathbf{H}_T, \mathbf{h}^{[s]})$ for each sample to calculate (3.6), (3.7), and (3.8), we warp them only once according to \mathbf{H}_T^{-1} , and warp the target model according to $\mathbf{h}^{[s]}$ s. If we denote the mean and the variance of the target model warped with $\mathbf{h}^{[s]}$ as $\mathbf{M}_T^{[s]}$ and $\mathbf{V}_T^{[s]}$, and the approximation of $\tilde{L}_{Tri,j}(g(\mathbf{H}_T, \mathbf{h}^{[s]}))$ as $\hat{L}_{Tri,j}(\mathbf{H}_T, \mathbf{h}^{[s]})$,

$$\hat{L}_{Tri,j}(\mathbf{H}_T, \mathbf{h}^{[s]}) = \max \left\{ \hat{L}_{T,j}(\mathbf{H}_T, \mathbf{h}^{[s]}), \tilde{L}_{B,j}(\mathbf{H}_T), \tilde{L}_{F,j}(\mathbf{H}_T) \right\}, \quad (3.12)$$

where

$$\hat{L}_{T,j}(\mathbf{H}_T, \mathbf{h}^{[s]}) = \mathcal{L}_N(\mathbf{M}_{T,j}^{[s]}, \mathbf{V}_{T,j}^{[s]}, \tilde{\mathbf{I}}_j). \quad (3.13)$$

Note that in this approximation using (3.12) and (3.13), only the target model needs to be warped for each sample, which reduces the amount of warping required to one third of the exact method.

This approximation means that we are evaluating each $\mathbf{h}^{[s]}$ by applying the change to the target model only, and not changing the target candidate region (defined by \mathbf{H}_T). In other words, we keep the target candidate region from previous iteration (or frame) and evaluate how much each sampled $\mathbf{h}^{[s]}$ would make the current observation plausible. To summarize, instead of $\Delta \mathbf{h}$, we use $\Delta \hat{\mathbf{h}}$, which

$$\Delta \hat{h}_k = \frac{\sum_s h_k^{[s]} \exp \left\{ \frac{\lambda}{M_T N_T} \sum_j \hat{L}_{Tri,j}(\mathbf{H}_T, \mathbf{h}^{[s]}) \right\}}{\sum_s \exp \left\{ \frac{\lambda}{M_T N_T} \sum_j \hat{L}_{Tri,j}(\mathbf{H}_T, \mathbf{h}^{[s]}) \right\}}, \quad (3.14)$$

where $\Delta \hat{h}_k$ is the k^{th} element of $\Delta \hat{\mathbf{h}}$. Finally, with $\Delta \hat{\mathbf{h}}$, we update \mathbf{H}_T at the end

of every iteration.

$$\mathbf{H}_T \leftarrow g\left(\mathbf{H}_T, \Delta \hat{\mathbf{h}}\right) \quad (3.15)$$

Usually, this iteration should be done until convergence but we found out experimentally that 10 iterations for each frame are enough to get an acceptable result.

3.2.3 Estimating Pixel-Wise Labels

With the estimated position of the target object $\hat{\mathbf{H}}_T$, we estimate the pixel-wise labels by finding the best matching model of the three. However, unlike Sections 3.2.1 and 3.2.2, we obtain results with respect to the pixels in the same dimension as the observation, denoted with indices i . In case of the background and the foreground model, we can easily obtain the log-likelihoods according to (3.7) and (3.8) without warping.

$$L_{B,i} = \mathcal{L}_N(\mathbf{M}_{B,i}, \mathbf{V}_{B,i}, \mathbf{I}_i) \quad (3.16)$$

$$L_{F,i} = \mathcal{L}_N(\mathbf{M}_{F,i}, \mathbf{V}_{F,i}, \mathbf{I}_i). \quad (3.17)$$

For the target model, we warp the SGMs with $\hat{\mathbf{H}}_T$ to match the observation. For pixels which are out of bounds, we simply consider the log-likelihood as minus infinite, so that the other two models take care of those pixels. If we denote the mean and the variance of the target model warped with $\hat{\mathbf{H}}_T$ as $\check{\mathbf{M}}_T$ and $\check{\mathbf{V}}_T$ (note that warping is performed with $\hat{\mathbf{H}}_T$), then, for i where $\check{\mathbf{M}}_{T,i}$ and $\check{\mathbf{V}}_{T,i}$ is within bounds,

$$L_{T,i}(\hat{\mathbf{H}}_T) = \mathcal{L}_N(\check{\mathbf{M}}_{T,i}, \check{\mathbf{V}}_{T,i}, \mathbf{I}_i). \quad (3.18)$$

With $L_{T,i}(\hat{\mathbf{H}}_T)$, $L_{B,i}$, and $L_{F,i}$, we choose the model which gives the largest log-likelihood among the three for each pixel. The estimated pixel-wise labeling result $\hat{\mathbf{R}}_i$ is obtained as in (3.3).

3.3 Learning the Tri-Model

Another key idea of our method is the learning strategy for the three models. In order for the proposed method to work robustly under occlusions and background clutters, it is important that the three models learn the desired different aspects of the scene. The target model learns the object that is tracked, similar to the models of conventional tracking methods. The background model learns the whole scene with respect to the global movement. The foreground model learns the moving objects in the scene except for the object learned by the target model. To achieve this aim, we use the pixel-wise labels, the estimated position of the target object, and the current observation, with different learning strategies applied to each of the three models.

3.3.1 Target Model

With the estimation results for frame t , i.e. the estimates $\hat{\mathbf{H}}_T^{(t)}$ and $\hat{\mathbf{R}}^{(t)}$, we update the target model using the observed image. To make the target model only learn the target object, we update the model only when we are sure that the observation is the target object. In other words, for the j^{th} pixel of the target model, we only update mean $\mathbf{M}_{T,j}$, variance $\mathbf{V}_{T,j}$, and age $\mathbf{A}_{T,j}$ when the labeling result in the target domain (obtained by warping $\hat{\mathbf{R}}^{(t)}$ with $\hat{\mathbf{H}}_T^{-1}$) denotes the target model.

For j which the condition is met,

$$\mathbf{M}_{T,j}^{(t+1)} = \left(1 - \alpha_{T,j}^{(t)}\right) \mathbf{M}_{T,j}^{(t)} + \alpha_{T,j}^{(t)} \check{\mathbf{I}}_j^{(t)} \quad (3.19)$$

$$\mathbf{V}_{T,j}^{(t+1)} = \left(1 - \alpha_{T,j}^{(t)}\right) \mathbf{V}_{T,j}^{(t)} + \alpha_{T,j}^{(t)} \left(\check{\mathbf{I}}_j^{(t)} - \mathbf{M}_{T,j}^{(t+1)}\right)^2 \quad (3.20)$$

$$\mathbf{A}_{T,j}^{(t+1)} = \mathbf{A}_{T,j}^{(t)} + 1, \quad (3.21)$$

where $\alpha_{T,j}^{(t)} = \frac{1}{\mathbf{A}_{T,j}^{(t)} + 1}$ is the varying learning rate. For pixels which do not meet the update condition, they are preserved and not changed. To keep the model from becoming overly too stiff, we cap the age at γ_T . This is to ensure that when learning is performed, we have learning rate at least over $\frac{1}{\gamma_T + 1}$. We also set a minimum value σ_T for the variance to keep the model from being too sensitive to noise.

3.3.2 Background Model

The background model learns the whole scene similar to background models used in conventional background subtraction methods (Stauffer & Grimson, 1999; KaewTrakulPong & Bowden, 2003). Thus, the dimension of the background model is the same as the input observation. To correctly apply the model to a moving scene, we continuously transform the learned model according to the estimated global motion of the scene. We estimate the global motion from time t to time $t + 1$ simply by observing the optical flows of the whole scene and obtaining the global affine movement. We divide the input image at time $t + 1$, i.e. $\mathbf{I}^{(t+1)}$, into 32×24 grids and perform optical flow (Tomasi & Kanade, 1991) on each corners of the grid with $\mathbf{I}^{(t)}$. With the point tracking results, we perform RANdom SAMpling and Consensus (RANSAC) (Fischler & Bolles, 1981) to obtain the homography matrix $\hat{\mathbf{H}}_{t+1:t}$ which matches pixels in image $\mathbf{I}^{(t+1)}$ to pixels from time $\mathbf{I}^{(t)}$. The inverse of this matrix is $\hat{\mathbf{H}}_{t:t+1}$, which is the estimated global

motion.

With the estimated global motion $\hat{\mathbf{H}}_{t:t+1}$, if we let $f_{\hat{\mathbf{H}}_{t:t+1}}(i)$ denote the result of the coordinate transform $\hat{\mathbf{H}}_{t:t+1}$ for pixel i , then the SGM for the i^{th} pixel of the background model is updated as

$$\mathbf{M}_{B, f_{\hat{\mathbf{H}}_{t:t+1}}(i)}^{(t+1)} = \left(1 - \alpha_{B,i}^{(t)}\right) \mathbf{M}_{B,i}^{(t)} + \alpha_{B,i}^{(t)} \mathbf{I}_i^{(t)} \quad (3.22)$$

$$\mathbf{V}_{B, f_{\hat{\mathbf{H}}_{t:t+1}}(i)}^{(t+1)} = \left(1 - \alpha_{B,i}^{(t)}\right) \mathbf{V}_{B,i}^{(t)} + \alpha_{B,i}^{(t)} \left(\mathbf{I}_i^{(t)} - \mathbf{M}_{B, f_{\hat{\mathbf{H}}_{t:t+1}}(i)}^{(t+1)}\right)^2 \quad (3.23)$$

$$\mathbf{A}_{B, f_{\hat{\mathbf{H}}_{t:t+1}}(i)}^{(t+1)} = \mathbf{A}_{B,i}^{(t)} + 1, \quad (3.24)$$

where $\alpha_{B,i}^{(t)} = \frac{1}{\mathbf{A}_{B,i}^{(t)} + 1}$ is the varying learning rate. Note that since we do not have $\mathbf{I}^{(t+1)}$ at time t , the update is performed immediately after the new observation at time $t + 1$ is given, and before other tracking process is carried out. The update equations are similar to the update equations for the target model (equations (3.19), (3.20)), and (3.21)), but for the background model, we learn all pixels without considering the pixel label $\mathbf{R}^{(t)}$. Also, we use $\mathbf{I}^{(t)}$ instead of $\check{\mathbf{I}}^{(t)}$. We cap the age for a pixel at γ_B as we do for the target model. We also keep a lower bound σ_B for the variance. However, we set σ_B to be larger than that for the target model σ_T , so that the target model would provide a better description of the observation when learning with the same data. For example, when a moving target object stops, the background model would also learn the target object, but would not provide as accurate explanation as the target model. Figure 3.1(c) is an example of the mean values of the background model.

3.3.3 Foreground Model

The foreground is designed to have the same dimension as the observed image and learns observation from pixels that are estimated to be from moving objects

(excluding the target object). When looking for pixels that are from moving objects, we look for pixels which do not fit well to the background model. In other words, we design the foreground model to learn the background subtraction results, i.e. pixels i which $\left(\mathbf{M}_{B,i}^{(t)} - \mathbf{I}_i^{(t)}\right)^2 > \mathbf{V}_{B,i}^{(t)}$. However, since we do not want the target object to be learned in this model, we use the pixel-wise label $\mathbf{R}_i^{(t)}$ to exclude observations from the target object from being learned. Since the foreground is learning the outliers of the background model, motion compensation is performed with the same $\hat{\mathbf{H}}_{t,t+1}$ as the background model. Thus, the update is performed as in (3.22), (3.23), and (3.24), but with subscripts B s replaced with F s, and only for the pixel i satisfying the condition of

$$\left[\mathbf{R}_i^{(t)} \neq T\right] \wedge \left[\left(\mathbf{M}_{B,i}^{(t)} - \mathbf{I}_i^{(t)}\right)^2 > \mathbf{V}_{B,i}^{(t)}\right]. \quad (3.25)$$

As in the case of other models, $\mathbf{A}_{F,i}^{(t)}$ is capped at γ_F . However, for the foreground model, we keep the age cap γ_F of the foreground model to be a relatively small value (5 in our experiments). The reason we keep γ_F low to have a fast learning rate is because of the temporary nature of the foreground model. Since we are compensating the foreground model with the global movement and not the movements of each individual object, positions of moving objects are temporary and change consistently. Thus, the foreground model needs a fast learning rate to deal with these changes. We also keep the lower bound for variance σ_F to be higher than the other two models, i.e. $\sigma_F > \sigma_B$ and $\sigma_F > \sigma_T$. If either of the two models can explain the observed scene better, due to the fast adaptation property of the foreground model, we trust the two models more than the foreground model. Figure 3.1(d) is an example of objects being learned which are not learned in the background model in Figure 3.1(c).

	FRAG	OAB	BEYOND	MIL	HOUGH	TLD	DM	Proposed
Caviar	0.728	0.765	0.057	0.533	0.871	0.303	0.939	0.935
FaceOcc	0.965	0.920	0.883	0.785	0.657	0.902	0.952	0.956
Woman	0.180	0.546	0.128	0.186	0.364	0.582	0.802	0.802
Sylvester	0.723	0.581	0.453	0.738	0.165	0.821	0.894	0.894
Tiger1	0.552	0.431	0.175	0.397	0.207	0.412	0.858	0.860
Motocross1	0.269	0.221	0.042	0.165	0.138	0.124	0.400	0.879
CupOcc	0.216	0.193	0.069	0.298	0.265	0.487	0.932	0.913
Cheetah	0.141	0.792	0.054	0.856	0.081	0.386	0.833	0.897
GreenMan	0.410	0.461	0.056	0.426	0.313	0.306	0.417	0.898

Table 3.1: Area under precision plots in Figure 3.5. Best result denoted with bold, sequences with large performance improvement marked with red.

3.4 Experimental Results

3.4.1 Experimental Settings

We implemented our method with C++ using the `opencv`¹ library. All parameters were fixed during the experiments. $M_T = N_T = 32$, $\gamma_T = \gamma_B = 60$, $\gamma_F = 5$, $\sigma_T = 20$, $\sigma_B = 25$, and $\sigma_F = 30$. These parameters are not optimized parameters but we found that they work well for most cases. Also to reduce effects from noise, we applied simple spatial median filtering with window size five to the input image. We compared our method with seven representative trackers with respect to the bounding box based results. We will denote these methods as FRAG (Adam et al., 2006), OAB (Grabner et al., 2006), BEYOND (Stalder et al., 2009), MIL (Babenko et al., 2011), HOUGH (Godec et al., 2013), TLD (Kalal et al., 2010), and DM (Yi et al., 2012). Also, to evaluate the performance of the pixel-wise results of our method, we compared it with HOUGH (Godec et al., 2013) which is a state-of-the-art method also able to give pixel-wise tracking results. We have tested the methods using nine image sequences. **Caviar** is from CAVIAR² data set, **FaceOcc** and **Woman** are from (Adam et al., 2006), **Sylvester** and

¹<http://opencv.com/downloads.html>

²EC Funded CAVIAR project/IST 2001 37540, found at URL: <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>

Tiger1 are from (Babenko et al., 2011), **Motocross1** is from (Godec et al., 2013), **CupOcc** and **Cheetah** are from (Yi et al., 2012), and **GreenMan** is from a part of the festival dataset in (Krausz & Bauckhage, 2011). Sequences contain various situations including occlusions from background objects, occlusions from other moving objects, and background clutters. For the experiments, we used the implementation provided by the authors of each paper. Some critical frames are shown in Figures 3.6 and 3.7.

3.4.2 Tracking Accuracy: Bounding Box

To evaluate the performance of the proposed method, we compared each tracker using precision plots used in (Babenko et al., 2011). The plots show ratio of correctly tracked frames w.r.t. thresholds. However, to consider the different sizes of the target objects, we used relative thresholds. This means that if we let w_{GT} and h_{GT} denote the ground truth object width and height, and let ϵ_x and ϵ_y denote the horizontal and vertical center point errors, the precision is the ratio of frames with results satisfying $(\epsilon_x < w_{GT}) \wedge (\epsilon_y < h_{GT})$. This relative threshold measure is more strict in case of evaluating sequences with target objects having small width or height (**Caviar**, **Tiger1**, **Sylvester**, **CupOcc**, and **Cheetah**). Precision plots for all sequences are shown in Figure 3.5, and the summary of these plots using the area under the precision plots are presented in Table 3.1.

As shown in Table 3.1, our method outperforms or is comparable to the compared methods. Note that the performances of the compared methods are not as good as they were reported in other papers (Babenko et al., 2011; Yi et al., 2012; Godec et al., 2013) due to the strictness of the used measure. For example, in the **Tiger1** sequence, the width of the ground truth is only about 40 pixels at start, and is even smaller than that in some occasions. Also for HOUGH, due to the pure random nature of their classifier, their performance differs from

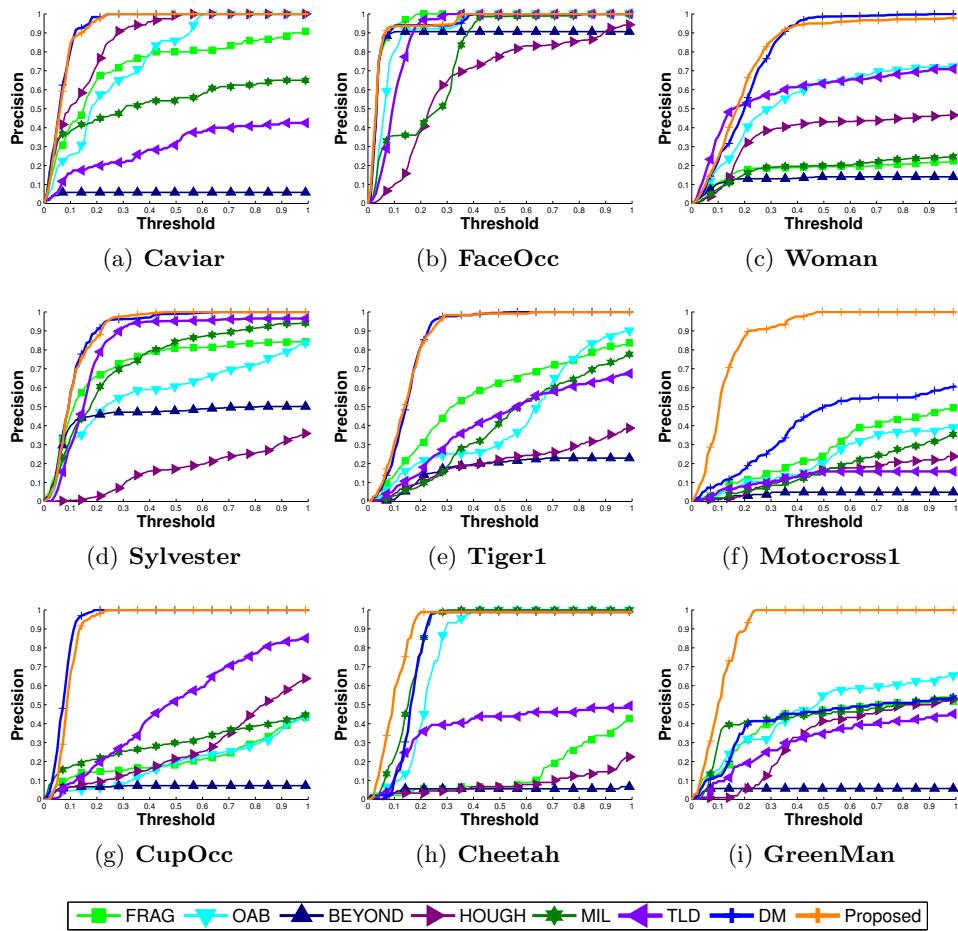


Figure 3.5: Precision plots for all sequences. See text for details. *Best viewed in color.*

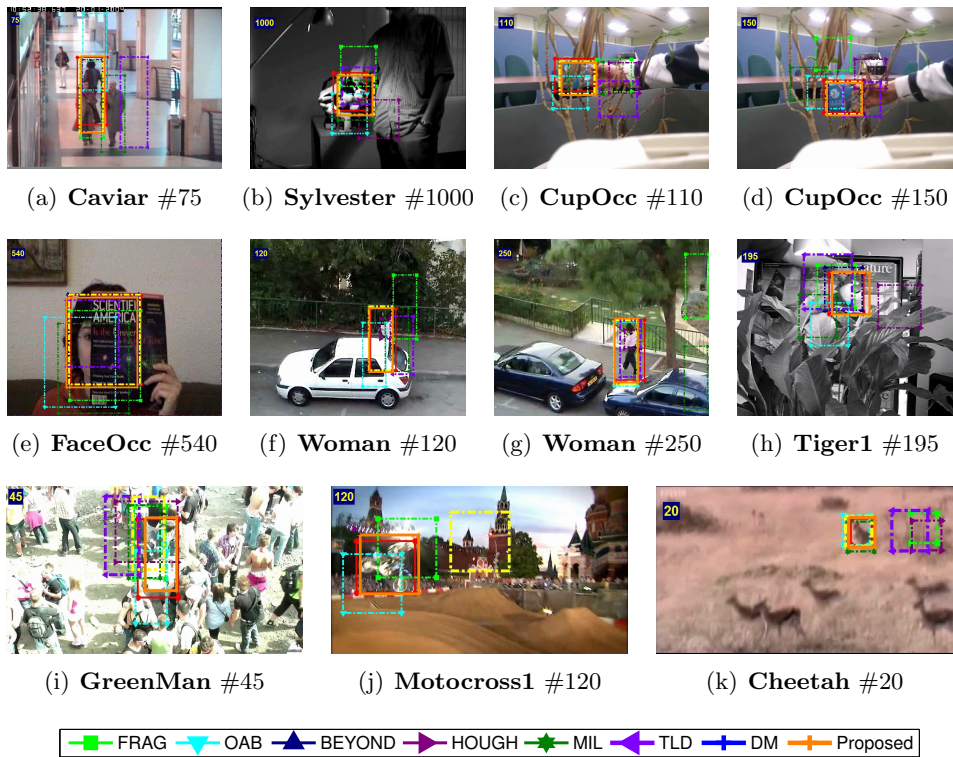


Figure 3.6: Bounding box results for critical frames. *Best viewed in color.*

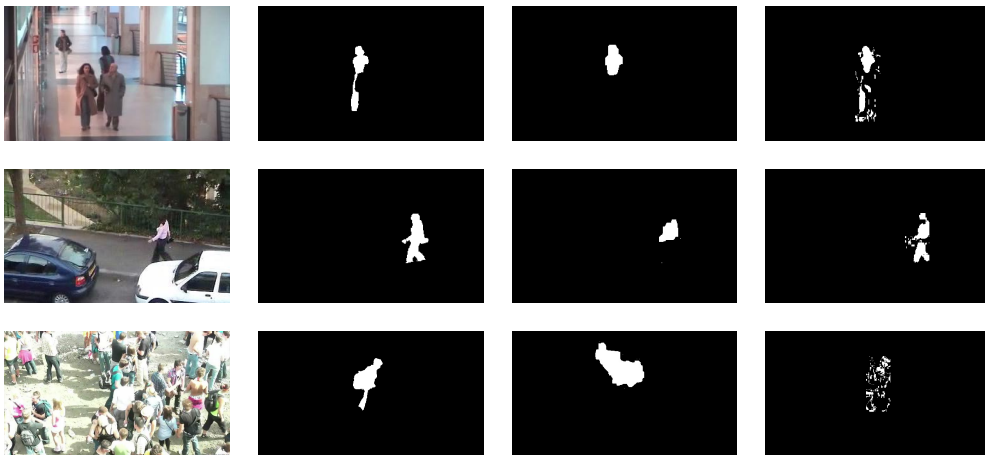


Figure 3.7: Pixel-wise tracking results. Input image (first column), ground truth (second column), results from HOUGH (third column), and the results of our method (last column). Each row is from **Caviar** frame 61 (first row), and **Woman** frame 161 (second row).

	HOUGH (Godec et al., 2013)	Proposed
Caviar	0.70	0.71
FaceOcc	0.73	0.75
Woman	0.32	0.70
Sylvester	0.07	0.52
Tiger1	0.10	0.53
Motocross1	0.44	0.41
CupOcc	0.13	0.56
Cheetah	0.004	0.62
GreenMan	0.22	0.44

Table 3.2: F-measure for pixel-wise tracking results. Best result denoted with bold, sequences with large performance improvement marked with red.

the results reported by Godec et al. (Godec et al., 2013). Our method shows promising results even with the strict measure. Yi et al.’s method (Yi et al., 2012) shows good results as well but in case of **Motocross1** and **GreenMan**, where the movement of the entire scene is complex or there are many moving objects, our method outperforms all others by significant amounts.

3.4.3 Tracking Accuracy: Pixel-Wise

To evaluate the performance of the pixel-wise results quantitatively, we obtained the F-measure for each sequence using manually annotated ground truth. The ground truth was made for every ten frames in the sequence. Table 3.2 is the comparison result. Except for **Motocross1**, which is a sequence from the work which HOUGH was introduced (Godec et al., 2013), our method shows better results. Especially for sequences **Woman**, **CupOcc**, and **Tiger1**, which have numerous occlusions, our method outperforms HOUGH (Godec et al., 2013) by large amounts. This is also true for **Cheetah** as well, where the target object is not so distinctive from the background. Some qualitative comparison of the pixel-wise results are shown in Figure 3.7.

3.5 Remarks and Discussion

A new pixel-wise visual tracking method using the novel tri-model representation has been proposed. The proposed tri-model was composed of three models, which each learned the different aspects of the scene. Using the tri-model, the proposed method estimated both the holistic position of the target object and the pixel-wise labels simultaneously. The contributions of the proposed method are summarized as follows. First, the proposed method not only focused on building a robust model, but also focused on learning the background and other moving objects in the scene which may interfere with tracking. Thus, in case background clutters or occlusions exist in complex scenes, the background and foreground models stopped them from hampering the tracking performances. Second, the proposed method was able to give pixel-wise labels outperforming the state-of-the-art. This pixel-wise labeling was also used in the learning process to prevent drifting. Third and last, experimental validation regarding the bounding box representation shows that the proposed method outperforms the state-of-the-art with promising results.

Chapter 4

Tracking by Feature-point-wise Saliency Model

In this chapter, to track objects robustly with inaccurate initializations and severe occlusions, we propose a method employing *motion saliency* and *descriptor saliency* of local features to learn and track the target object based on GHT (a voting based method which combines partial solutions effectively to obtain a global solution) (Ballard, 1981). In order to achieve good tracking results even with inaccurate initializations, we define the two saliencies related to motions and descriptors (explained in detail in Section 4.1). With the two proposed saliencies, our model learns to put more weight on good partial results when obtaining the global solution with GHT voting. In other words, rather than just trying to find what was given at initialization, the two saliencies work together to learn the salient characteristics of the target object, which also results in change of the influences of initial features. Method in (Mahadevan & Vasconcelos, 2009) also uses the concept of saliency, but their definition of saliency is a criterion for selection of features (features such as colors or or edges, not to be confused with

feature points) similar to (Avidan, 2007). Also, the bottom-up saliency they use for initialization is a center-surround saliency (unlike ours which considers target and non-target rather than center and surround), and does not always guarantee that it highlights the target object.

With the two saliencies, we use GHT in order to properly combine the estimates from multiple local feature points. GHT is a powerful method for combining partial estimates into a whole used in many tracking methods (Godec et al., 2013; Grabner et al., 2010; Asadi et al., 2007). Unlike them, in our case, the mapping table of GHT containing partial solutions (votes) acts as a model learning descriptor saliencies. Since we use GHT voting to combine multiple estimates from local features, our method gives robust results even if some features of the target object become occluded. Furthermore, we learn the model using all local features in the scene and keep local features which move along with the target object, thus giving robust results even when all of the target object is occluded (and in case of severe occlusions). This is similar to the method by Grabner et al. (Grabner et al., 2010), which creates *supporters* with nearby features, and use them to aid tracking in case of severe occlusions. However, the performance of their method relies much on the primary tracker being used, which is not always accurate and suffers from initialization problems, whereas our method successfully deals with both problems simultaneously. Dinh et al. (Dinh et al., 2011) also use supporters to aid tracking, but their method still treats initialization to be accurate.

4.1 Proposed Method

The overall scheme of the proposed method is depicted in Figure 4.1. Our method is based on GHT. For each frame, we extract local features. Then, using the learned feature database (DB), we match each feature to obtain partial solutions

for the center of the target object. We then combine the partial results with GHT to obtain a global solution. When collecting the partial results using GHT, in order to deal with inaccurate initializations, each solution is weighted according to the two proposed saliencies (the descriptor saliency and the motion saliency).

The feature DB is learned *on-the-fly* during the tracking process. The feature DB keeps track of distinctive features w.r.t. their descriptors (descriptor saliency), and where the center of the target object would be for each item. The proposed motion saliency is obtained using the learned descriptor saliency of features and the optical flow of the local feature. The motion saliency is designed so that the features showing distinctive motion characteristics of the target object have higher values. With the two saliencies, the salient characteristics of the target object are learned in the model (feature DB). Details of the proposed method are explained in the following subsections.

4.1.1 Tracking based on GHT

The proposed tracking scheme using GHT starts by building a likelihood map for the center position of the target object and obtains the result by finding the maximum on this map. The likelihood map is created through GHT by combining the center estimates (votes) from each local feature point. When combining the estimates, we weight them w.r.t. their saliencies so that salient features are more accounted for. Figure 4.2 is an illustration of this process. Mathematically, if we denote the estimated center for the j^{th} feature in the current observation as $(x_{c,j}, y_{c,j})$ and its weight as w_j , the likelihood map \mathbf{A} is defined as

$$\mathbf{A}(x, y) = \sum_j w_j \exp \left\{ -\frac{(x_{c,j} - x)^2 + (y_{c,j} - y)^2}{2\sigma_{\mathbf{A}}^2} \right\}, \quad (4.1)$$

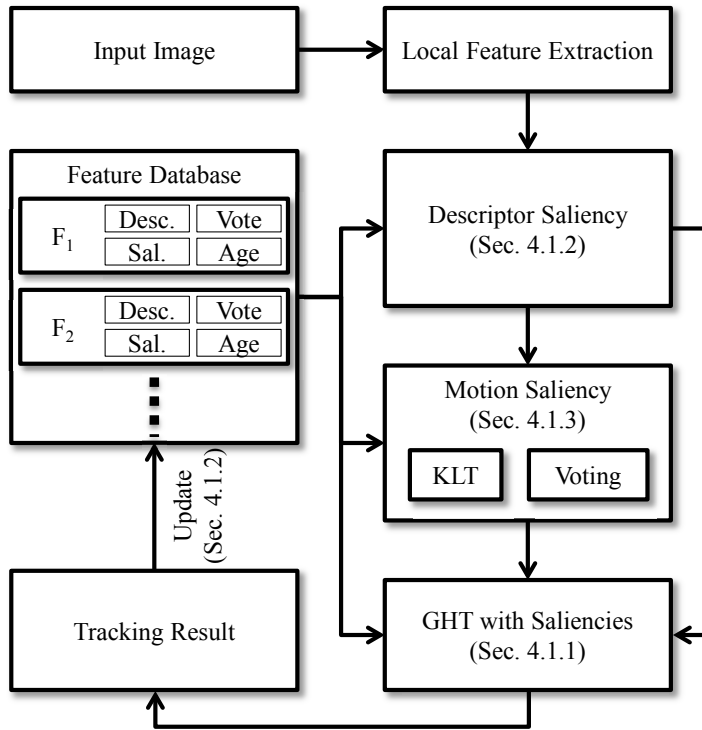


Figure 4.1: Overall scheme of the proposed method.

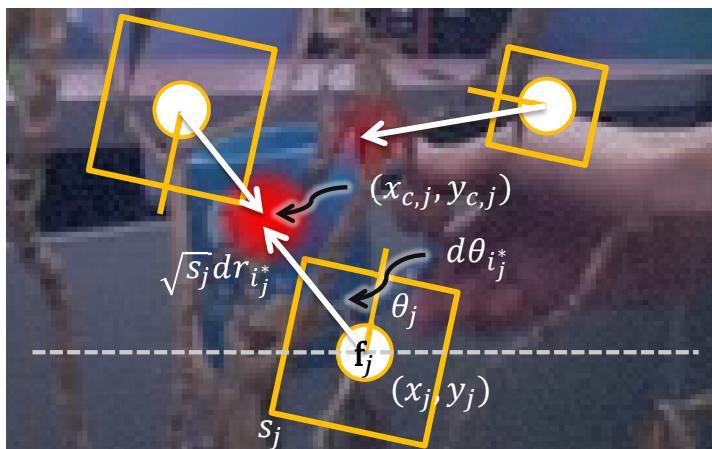


Figure 4.2: Illustration of GHT voting with SURF features

where $\sigma_{\mathbf{A}}$ is the standard deviation of the Gaussian kernel used for combining estimates, which is a parameter controlling the smoothness of the likelihood map. This combining process is referred to as voting in GHT, each partial estimates as votes, w_j as voting weights, and the resultant likelihood map as the vote map. Since the target object position changes little between frames, we apply temporal low-pass filtering (temporal weighted averaging) to the vote map to take advantage of this fact. Then, with this vote map, we can find the target object position $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = (\hat{x}, \hat{y}) = \arg \max_{x,y} \mathbf{A}(x, y). \quad (4.2)$$

Local Estimates. The estimates from each local feature point are obtained by matching with the feature DB. The feature DB consists of multiple items, of which each item contains descriptor information (\mathbf{d}), distance to the estimated object center normalized with the square-root of the size of the feature (dr), angle difference between the major orientation and the vector to the estimated center ($d\theta$), saliency of the item (ζ), and the age (α) of the item. If we denote i^{th} item of the feature DB \mathbf{F} as \mathbf{F}_i then,

$$\mathbf{F}_i = (\mathbf{d}_i, dr_i, d\theta_i, \zeta_i, \alpha_i). \quad (4.3)$$

For the j^{th} local feature of the current scene, we denote it as $\mathbf{f}_j = (\mathbf{x}_j, \mathbf{d}_j, s_j, \theta_j)$, where $\mathbf{x}_j = (x_j, y_j)$, \mathbf{d}_j , s_j , and θ_j is the pixel position, the descriptor, the size (area), and the major orientation of the feature, respectively. Then, we find the best matching item using the learned feature DB $\mathbf{F}^{(t)}$ at time t (learning strategy detailed in Section 4.1.2) in terms of \mathbf{d}_j and $\mathbf{d}_i^{(t)}$, and use this match for voting. In other words, if there exists i_j^* such that

$$i_j^* = \arg \min_i \left\{ \left\| \mathbf{d}_i^{(t)} - \mathbf{d}_j \right\| \mid \left\| \mathbf{d}_i^{(t)} - \mathbf{d}_j \right\| < \epsilon_d \right\}, \quad (4.4)$$

where ϵ_d is a threshold, we consider feature j to be matched with $\mathbf{F}_{i_j^*}^{(t)}$ and vote to $(x_{c,j}, y_{c,j})$ with weight w_j as in Figure 4.2, where

$$x_{c,j} = \sqrt{s_j} dr_{i_j^*} \cos(d\theta_{i_j^*} + \theta_j) + x_j \quad (4.5)$$

$$y_{c,j} = \sqrt{s_j} dr_{i_j^*} \sin(d\theta_{i_j^*} + \theta_j) + y_j. \quad (4.6)$$

Any type of affine invariant feature can be used such as SIFT (Lowe, 2004) or SURF (Bay et al., 2008) but we used SURF for ease in implementation. Note that in (4.5) and (4.6), we take advantage of the affine invariant properties of the local features and compensate the voting vector to fit the current observation with $\sqrt{s_j}$ and θ_j . This lets the proposed method to be able to vote regardless of the scale and rotation change of the target object.

Voting Weights. The weight w_j is designed to account both the motion saliency η_j and the descriptor saliency $\zeta_{i_j^*}^{(t)}$. We use the multiplication of the two saliencies to emphasize features which have both saliency values high. Therefore for the j^{th} detected local feature, the weight is defined as

$$w_j = \eta_j \zeta_{i_j^*}^{(t)}. \quad (4.7)$$

Details about these saliencies and their effects are presented in Section 4.1.2 and Section 4.1.3.

4.1.2 Descriptor Saliency and Feature DB Update

To learn the salient features of the target object w.r.t. the shape of the target object, we define the descriptor saliency as how much the descriptor coincided with past consensus. For each item in $\mathbf{F}^{(t)}$, the descriptor saliency $\zeta^{(t)}$ is learned to hold how good the partial (voting) results were when using the item, *i.e.* the

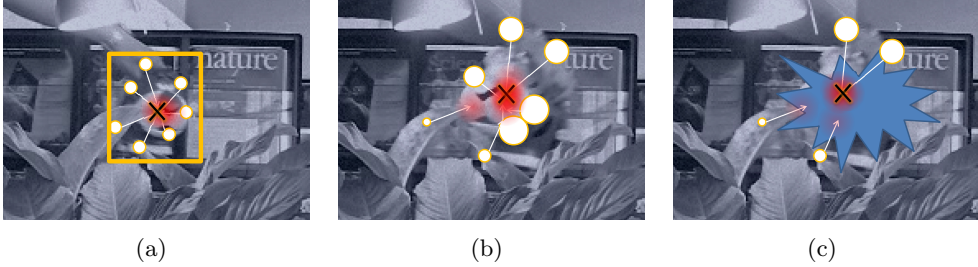


Figure 4.3: Illustration of the descriptor saliency in action. Detected local features are depicted with circles, having their sizes as their descriptor saliency values (larger means high). Red means having higher vote map value \mathbf{A} . (a) Initial voting for the target object position, (b) voting after t frames, and (c) voting in case of occlusion. Note that these are not actual experimental results and are only illustrations.

value of the vote map \mathbf{A} . Since our framework is based on GHT, we can simply achieve this by looking at each votes and DB matches (back projection). For example, the feature items matched with the features pointing to the center of the cup in Figure 4.2 would be updated with high saliency values, whereas items matched with the feature pointing at the wrong direction (*e.g.* feature point on the hand) would be updated with low saliency value.

Initially, $\mathbf{F}^{(0)}$ is an empty set without any elements. As the target object information is provided with a bounding box in the first frame, feature points inside the bounding box are added to $\mathbf{F}^{(1)}$ with descriptor saliency $\zeta = 1$, and feature points outside the bounding box are added to $\mathbf{F}^{(1)}$ with descriptor saliency $\zeta = 0$ (Figure 4.3(a)). Then, we continuously update the descriptor saliencies using the back projection result of each vote on $\mathbf{A}^{(t)}$. This means that at time t , if item $\mathbf{F}_i^{(t)}$ has been matched with M_i features (*i.e.* there exists M_i number of j such that $\|\mathbf{d}_i^{(t)} - \mathbf{d}_j\| < \epsilon_d$), the descriptor saliency of this item $\zeta_i^{(t)}$ is updated

with the average of the vote map value for all matches.

$$\zeta_i^{(t+1)} = \beta_i \zeta_i^{(t)} + (1 - \beta_i) \frac{1}{M_i} \sum_{m=1}^{M_i} \frac{\mathbf{A}(x_{c,m}, y_{c,m})}{\max \mathbf{A}(\cdot)} \quad (4.8)$$

where m is the index of the matched local feature in the current frame, β_i is the variable learning rate $\beta_i = \alpha_i^{(t)} / (\alpha_i^{(t)} + 1)$. Similarly, we update the voting information of the item with the average normalized distance and angle from each feature to the obtained tracking result, and also increment the age of the item. At time t , with the tracking result $\hat{\mathbf{x}}^{(t)}$ and position of the matched features \mathbf{x}_m

$$dr_i^{(t+1)} = \beta_i dr_i^{(t)} + (1 - \beta_i) \frac{1}{M_i} \sum_{m=1}^{M_i} \frac{\|\hat{\mathbf{x}}^{(t)} - \mathbf{x}_m\|_2}{\sqrt{s_m}} \quad (4.9)$$

$$d\theta_i^{(t+1)} = \beta_i d\theta_i^{(t)} + (1 - \beta_i) \frac{1}{M_i} \sum_{m=1}^{M_i} \left[\angle(\hat{\mathbf{x}}^{(t)} - \mathbf{x}_m) - \theta_m \right] \quad (4.10)$$

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + 1, \quad (4.11)$$

where $\|\cdot\|_2$ and $\angle(\cdot)$ is the Euclidean norm and the angle of a vector, respectively. If an element in $\mathbf{F}^{(t)}$ has no match, *i.e.* $M_i = 0$, we do not update that element. Also, the elements of $\mathbf{F}^{(t)}$ which were added in the first frame are treated as an exception and we never update $dr_i^{(t)}$ and $d\theta_i^{(t)}$ for them to prevent the tracker from drifting. Still, since we update $\zeta_i^{(t)}$, the effects of initial elements can change. Figure 4.3(a) and Figure 4.3(b) is an example of some local features (features on leaves) having high descriptor saliency at initialization, but becoming low after learning them correctly as tracking is performed.

Practical Implementation. Ideally, it would be best if we add all new unmatched features into the database, but this would not be practical due to computational and memory requirements. Thus, to keep $\mathbf{F}^{(t)}$ in a reasonable size without harming the overall performance, we apply an update strategy inspired

by the work of Avidan (Avidan, 2007). Except for the elements of $\mathbf{F}^{(t)}$ which were added in the first frame, we keep the K best elements in terms of $\zeta_i^{(t)}$ and get rid of others from $\mathbf{F}^{(t)}$. After removing bad elements from $\mathbf{F}^{(t)}$, we add L most motion-salient unmatched feature points into the DB. The L best considering the motion saliency η_j are added to $\mathbf{F}^{(t+1)}$ with its motions saliency as initial descriptor saliency, *i.e.* the following 5-tuple

$$(\mathbf{d}_j, dr_j, d\theta_j, \eta_j, 1) \quad (4.12)$$

is added, where

$$dr_j = \frac{\|\hat{\mathbf{x}}^{(t)} - \mathbf{x}_j\|_2}{\sqrt{s_j}}, \quad (4.13)$$

$$d\theta_j = \angle(\hat{\mathbf{x}}^{(t)} - \mathbf{x}_j) - \theta_j. \quad (4.14)$$

With this update strategy, if we denote the number of elements initially added to $\mathbf{F}^{(1)}$ as K_{init} , the size of $\mathbf{F}^{(t)}$ will always be smaller than $K_{init} + K + L$. Note that during the feature DB update, all current local features are considered. This means that if there are background local features which help in estimating the target object position, they will also be learned. These un-occluded salient features can act as supporters similar to (Grabner et al., 2010), aiding in case of severe occlusions as in Figure 4.3(c).

4.1.3 Motion Saliency

To capture the characteristics of the target object in terms of motion, we define the motion saliency of a feature point with its descriptor saliency and optical flow (Figure 4.4(b)), and emphasize motions which are distinctive (Figure 4.4(d)). By distinctive, we expect the motion of the target object to stand-out from back-

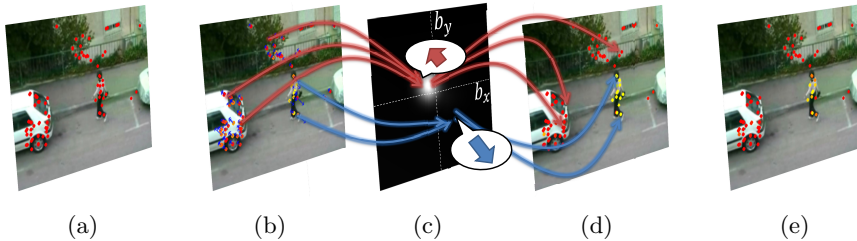


Figure 4.4: Example of motion saliency obtained for the **woman** sequence. (a) Detected local feature points, (b) descriptor saliency of matched local features and their optical flows (denoted yellow if high saliency and red if low saliency, optical flows displayed 3 times their original magnitude), (c) motion vote map \mathbf{B} , (d) motion saliency for each detected local feature, and (e) final voting weight from both saliencies. Arrows from (b) to (c) and (c) to (d) illustrate where motions are mapped. *Best viewed in color.*

ground motion (including motion from other non-target objects). For obtaining motions for each feature points, we use backward optical flow from time t to time $t - 1$. The backward optical flow $\mathbf{b} = (b_x, b_y)$, where b_x and b_y are backward optical flows in horizontal and vertical direction, can be easily obtained through KLT (Tomasi & Kanade, 1991).

The way we measure the distinctiveness is by constructing a likelihood map of background motions through consensus (Figure 4.4(c)). This likelihood map is constructed using voting strategy similar to the GHT voting used for tracking. When constructing the likelihood map, we weight each motion of the detected local feature points so that the resultant likelihood map would have higher values if the motion is likely to be from background. Therefore, with the likelihood map we are able to know which motions are similar to background motions and which are distinct. We will refer to this likelihood map as the *motion vote map*. We obtain the motion saliency of feature point by simply using the inverse of the motion vote map value. By doing so, the motion saliency value will be high when the vote map value is low, *i.e.* not similar to background motions. If we denote

the motion vote map as \mathbf{B} , then for a feature point \mathbf{f}_j in the current observation with backward optical flow \mathbf{b}_j , the motion saliency for this feature η_j can be defined as

$$\eta_j = 1 - \frac{\mathbf{B}(\mathbf{b}_j)}{\max \mathbf{B}(\cdot)}, \quad (4.15)$$

where the motion vote map \mathbf{B} is constructed by all features in the current frame having a matched item i_j^* from (4.4) as

$$\mathbf{B}(\mathbf{b}) = \sum_{\forall j | \exists i_j^*} \left(1 - \zeta_{i_j^*}^{(t)}\right) \exp \left\{ -\frac{\|\mathbf{b} - \mathbf{b}_j\|^2}{2\sigma_{\mathbf{B}}^2} \right\}, \quad (4.16)$$

where $\sigma_{\mathbf{B}}$ is the standard deviation of the Gaussian kernel used for voting. Note that for the motion vote map, we weight the votes with $\left(1 - \zeta_{i_j^*}^{(t)}\right)$, which is the inverse of descriptor saliency so that the vote map is about background motions. η_j from (4.15) has a value in range $[0, 1]$, representing how distinctive the backward optical flow of a feature point is from background motions.

The advantage of our method using motion vote map is that we are able to capture distinctive motions regarding the target object robustly without any sophisticated motion grouping. As in Figure 4.4(d), even though there are other motions due to the camera movement, only the distinctive motions similar to the target object is emphasized.

4.2 Experimental Results

We implemented our method (PROP) in C++ with the OpenCV¹ library for SURF and KLT. For all experiments, the threshold for determining feature point matches $\epsilon_d = 0.1$, the variances of the Gaussian kernels for voting $\sigma_{\mathbf{A}}$ and $\sigma_{\mathbf{B}}$ are both set to 10, the number of elements to keep $K = 500$, and the num-

¹<http://opencv.com/downloads.html>

ber of features added $L = 100$. We also considered tracking results having $\mathbf{A}^{(t)}(\hat{\mathbf{x}}^{(t)}) < 0.05$ as tracking failures.

We tested our method against nine other representative trackers. MST (Comaniciu et al., 2003) and FRAG (Adam et al., 2006) are kernel-based trackers, where FRAG is mainly focused on solving occlusion problems. OAB (Grabner et al., 2006), SEMI (Grabner et al., 2008), BEYOND (Stalder et al., 2009), and MIL (Babenko et al., 2011) are boosting based methods. HOUGH (Godec et al., 2013) is a method based on Hough forests and TLD (Kalal et al., 2010) is a method with P-N learning strategy combining the result of both tracker and detector. SEMI, BEYOND, MIL, and TLD are mostly targeted for solving drifting issues and HOUGH is a method targeted to overcome inaccuracies arising from a bounding box representation of the target. Finally INVIS (Grabner et al., 2010) is a method using GHT similar to ours. For the experiments, we used the implementation provided by the authors of each paper except for MST and INVIS. For MST and INVIS, we implemented them in C++ according to the papers. We performed our experiments with nine image sequences. Sequences **coke**, **tiger1**, and **tiger2** are from (Babenko et al., 2011), **syvester** is from (Ross et al., 2008), **occFace** and **woman** are from the (Adam et al., 2006), **motocross1** and **mtn.bike** are from (Godec et al., 2013), and **occCup** is our own. Implementation of the proposed method and the datasets used are available at the first author’s website². Results for critical frames are shown in Figure 4.6.

To compare results quantitatively, we used manually annotated bounding box representation of the target object as the ground truth. Two measures were used to evaluate the algorithms; mean error between the ground truth center point and the tracking result, and the percentage of correctly tracked frames. By correctly tracked frames, we counted the tracking result as correct if the center of the

²<https://sites.google.com/site/homekmyi>

tracking result was inside the ground truth bounding box. The reason we applied this measure instead of the commonly used overlap criterion is because we are using random initializations which may have little overlap even from the beginning. This measure can be understood as a weak condition of tracking success. When the overlap measure is used, results for all trackers become degraded since some initializations would be counted as failures even at the first frame. Still, the relative performances of trackers remain similar since this happens equally for all trackers. For trackers being able to detect tracking failures, we did not use these frames for computing the mean error. However, they are considered as tracking failures in the percentage of correctly tracked frames for fair comparison.

4.2.1 Tracking with Inaccurate Initializations

To validate the robustness of our method against clumsy initializations, we have tested trackers with 100 random initializations. Of the 100, the first initialization is identical to the ground truth and 20 contain initializations having the center point of the bounding box fixed at the ground truth but having different width and height (sampled uniformly having maximum difference to be 20% of the original width or height). Another 20 contain initializations having the same width and height as the ground truth, but with the center point differing (sampled uniformly having maximum difference to be 50% of the width or height of the target object.) Finally, the remaining 59 have both the width and height, and the center point differing from the ground truth in the same sense above.

Results for all sequences with all initializations are shown in Table. 4.1 (PROPm and PROPd are the results of our method using only motion saliency and descriptor saliency, respectively). Initialization overlap in the table is defined as the percentage ratio between the intersection and the union of the initial bounding box and the ground truth. In Table. 4.1, it can be observed that as

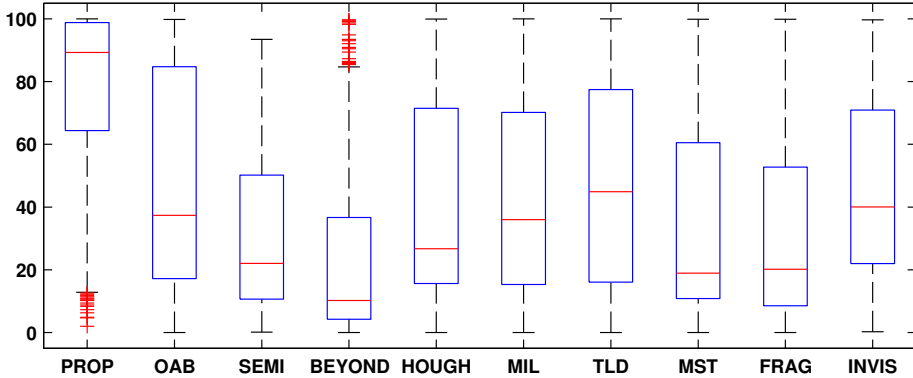


Figure 4.5: Box plots for % correctly tracked with all initializations.

initialization overlap decreases, the average performance of trackers generally degrade (lower percentage of correctly tracked frames and larger mean errors). For percentage of correctly tracked frames, when considering best results, our method shows 94.2% whereas the second best except the results of our method is TLD with 60%~40% overlap showing 89.3% (denoted by bold blue text). However, when considering the average performance, with the same condition, our method shows 84.5% whereas TLD shows 51.3%. Note that the gap between the best performance and the average performance our method is relatively small compared to other methods. This shows that our method is less sensitive to initializations. Also, in terms of average performance, our method significantly outperforms other methods including INVIS, which uses BEYOND as a primary tracker and uses GHT similar to ours. In case of mean error in the correctly tracked frames, our method is not best but shows comparable results against other methods. Occasionally, BEYOND shows best results in terms of mean error, but the percentage of correctly tracked frames shows that only a limited number of frames were tracked. Note that using only one of the two saliencies degrades performance (PROPM and PROPD).

Figure 4.5 is a box plot demonstrating the performance of trackers against

different initializations. The whiskers (denoted with red pluses) are data points having values more than 1.5 inter quartile range away from the median (red line). In Figure 4.5, it can be seen that the best performances (dotted lines) do not differ much. This implies that with a particular initialization designed for each algorithm, the performances may not differ much. However, this is not a trivial task, and when considering average performance, our method outperforms all other compared methods by significant amounts.

	Algorithm	Initialization Overlap				
		100% ~ 80%	80% ~ 60%	60% ~ 40%	40% ~ 20%	20% ~ 0%
Correctly Tracked Frames (%)	PROP	87.7±15.3 (90.5)	88.0±15.7 (92.6)	84.6±18.6 (94.2)	72.8±27.0 (92.9)	55.3±32.9 (75.8)
	PROPm	80.7±20.7 (82.2)	77.2±23.8 (85.4)	70.5±27.2 (88.5)	64.1±29.5 (87.7)	46.9±34.3 (70.0)
	PROPd	41.3±39.9 (42.6)	40.8±34.9 (47.9)	38.2±32.5 (48.5)	34.2±30.5 (48.1)	22.8±25.3 (34.0)
	OAB	55.0±35.1 (76.3)	55.4±34.7 (80.6)	50.5±34.6 (88.6)	43.0±32.6 (78.4)	34.8±31.2 (62.9)
	SEMI	28.3±23.0 (46.2)	34.3±24.6 (54.7)	30.8±23.0 (65.9)	31.2±24.6 (67.9)	23.9±23.4 (42.2)
	BEYOND	28.1±29.7 (44.1)	23.9±27.8 (55.7)	22.9±24.8 (57.6)	21.7± 23.8 (56.8)	18.9±21.7 (39.9)
	HOUGH	50.0±34.0 (63.9)	46.9±33.1 (71.3)	44.1±33.1 (79.1)	37.7±33.6 (77.1)	26.4±29.6 (51.7)
	MIL	56.0±32.2 (62.8)	52.5±30.6 (70.3)	46.7±32.3 (75.4)	37.8±31.6 (74.1)	24.4±25.7 (55.7)
	TLD	61.7±32.2 (72.0)	54.3±30.1 (81.8)	51.3±33.7 (89.3)	41.3±31.7 (88.5)	26.6±26.4 (58.5)
	MST	43.6±38.1 (50.5)	43.7±37.2 (55.2)	40.1±35.1 (62.4)	29.5±27.3 (56.2)	17.8± 17.5 (31.9)
	FRAG	43.8±33.9 (57.5)	46.1±31.4 (64.2)	35.6±28.8 (59.1)	25.7±28.5 (46.6)	16.0±23.9 (31.4)
	INVIS	50.2±31.4 (71.5)	51.5±29.0 (78.2)	47.9±27.5 (77.5)	43.2±27.2 (71.1)	33.4±26.1 (56.7)
	Mean Error (pixels)	PROP	21.1±19.7 (17.8)	20.7±17.8 (13.4)	25.2±19.2 (15.5)	34.5±24.4 (19.8)
OAB		38.4±41.6 (14.8)	41.1±35.8 (18.3)	45.4±33.5 (18.6)	50.4±34.5 (23.3)	63.6±40.0 (35.6)
SEMI		19.6±21.7 (9.6)	23.0±17.7 (8.0)	41.0±37.7 (16.6)	52.5±36.8 (23.1)	72.7±45.7 (45.4)
BEYOND		12.0±12.9 (6.1)	14.5±12.3 (4.2)	25.9±19.3 (7.2)	38.8±27.1 (14.0)	57.0±34.7 (37.8)
HOUGH		58.3±45.5 (31.5)	56.8±41.9 (26.6)	57.8±42.4 (24.8)	64.2±44.7 (24.6)	89.1±55.7 (44.7)
MIL		39.3±34.4 (20.5)	43.6±35.6 (14.9)	48.9±38.7 (17.3)	60.5±43.2 (20.7)	73.4±49.4 (39.1)
TLD		85.7±95.1 (52.3)	87.0±89.2 (40.4)	92.1±93.8 (30.7)	101.5±97.1 (38.9)	124.2±103.4 (75.4)
MST		75.1±56.6 (51.6)	69.0±51.5 (48.5)	75.6±54.3 (38.4)	83.8±56.4 (49.2)	100.7±58.6 (76.9)
FRAG		66.8±47.4 (44.7)	65.3±46.2 (40.7)	81.4±55.9 (49.0)	95.3±66.1 (72.9)	113.7±70.7 (86.9)
INVIS		60.1±52.0 (34.6)	60.8±50.5 (32.1)	63.2±48.0 (36.3)	68.8±49.4 (42.5)	83.5±50.9 (54.3)

Table 4.1: Results for all sequences. [average±standard deviation (best average)]. For best average, best results for all sequences were averaged. Bold denotes best result among the compared algorithms.

4.2.2 Tracking Under Occlusions

To evaluate the performance of our method against occlusions, we have tested our method on sequences **coke**, **tiger1**, **tiger2**, **occFace**, **woman**, and **occCup**. In these sequences, occlusion of the target object exist, expecially with the **coke** sequence having the object fully occluded at occasions, and **occCup** having the object occluded even at initialization. Critical frames for sequences with occlusions are shown in Figures 4.6(a) to 4.6(i). Figure 4.6(b) is an example of severe occlusion where the target object gets fully occluded. Our method successfully tracks the target object even in such case, by learning the features of the hand which moves together with the target object. In Figure 4.6(e), the target object is occluded even at initialization. As in Figure 4.6(f), many compared methods fail to recognize the cup as the target object and fail. However, our method successfully tracks the target object.

4.3 Remarks and Discussion

A new visual tracking method for tracking objects in case of severe occlusions and uncertain initialization has been proposed. The proposed method used *motion saliency* and *descriptor saliency* of local features and obtained the target position through GHT. The motion saliency of a local feature emphasized features having distinctive motions, compared to the motions coming from local features which are not from the object. The descriptor saliency emphasized features which are likely to be of the object in terms of its feature descriptors. Through these saliencies, the proposed method *learned and found* the target object in the image sequence. The saliencies and GHT combined allowed the tracker to have robust performances under occlusions and clumsy initializations.

The proposed method was extensively tested against nine other methods,

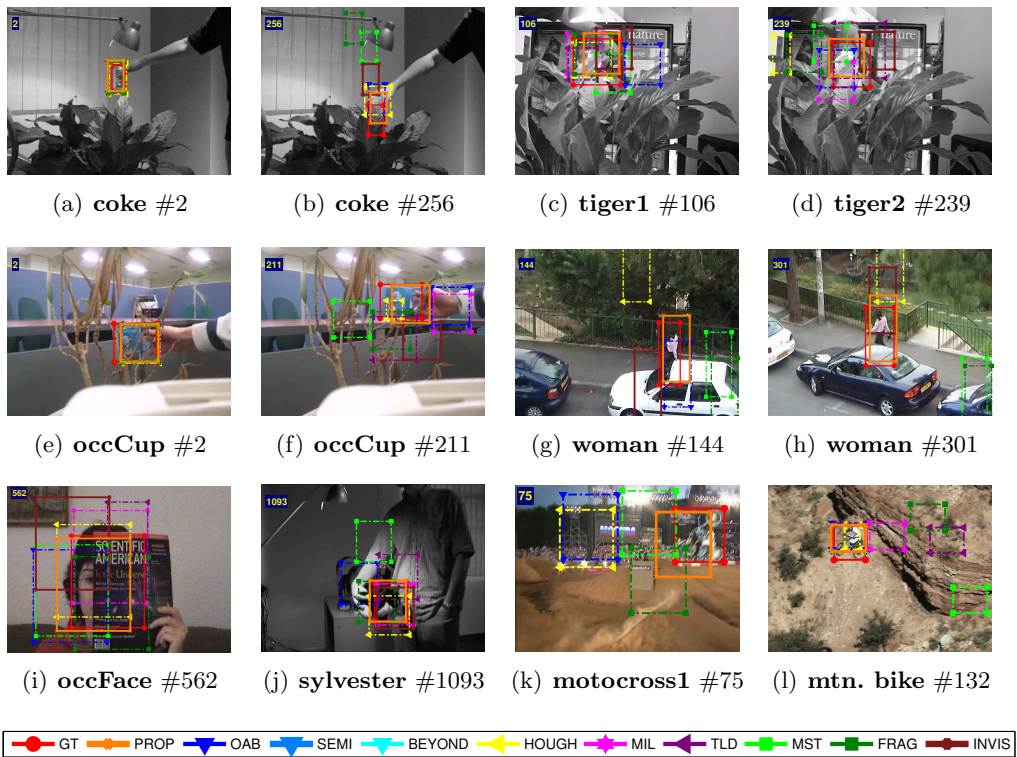


Figure 4.6: Critical frames for tracking results. Subcaptions denote sequence names and frame numbers. *Best viewed in color.*

using nine image sequences, and with hundred random initializations. The experimental results demonstrated the robustness of our method against occlusions and initializations, outperforming the other compared methods significantly.

Chapter 5

Tracking by Patch-wise Elastic Structure Model

In this chapter, to solve the issues related to tracking of non-rigid objects with occlusions in real-time, we focus on building a robust representation for CLTM. The proposed algorithm models the target object through a structure of local patches with spring-like connections, formulated under the Maximum A Posterior - Markov Random Field (MAP-MRF) framework. Each local patch is considered to be connected with its neighbors and, therefore, the local structures of the target object are embedded into the MRF structure. When partial occlusions occur, un-occluded patches will enforce the maintenance of the local structures owing to the spring-like connections among the patches. As a result, the neighboring occluded patches will be directed to their correct positions through the relationship between patches, thus making our method robust against partial occlusions. Non-rigid deformations are also well described since they can be explained as a collection of local movements of patches. Unlike other methods which concentrate on occlusions (Adam et al., 2006; Mei & Ling, 2009), or methods which focus on

non-rigid movements only (Kwon & Lee, 2009; Godec et al., 2013), our method addresses both problems simultaneously. In addition, to achieve real-time performance, which is critical in most tracking applications, a hierarchical diffusion approach is proposed to overcome the curse of dimensionality.

To demonstrate the effectiveness of our method, we have experimented with a number of challenging image sequences. The experimental results show that our method is the most robust against both partial occlusions and non-rigid deformations, compared with other methods. Especially, our method runs in real-time (20 to 50 frames per second), whereas other state-of-the-art methods capable of tracking non-rigid objects proposed in (Kwon & Lee, 2009) and (Godec et al., 2013) runs only a few frames per second.

5.1 Tracking with Elastic Structure of Local Patches

5.1.1 Sequential Bayesian Inference Framework

The proposed tracking method is based on a sequential Bayesian inference framework. We denote the object state at time t as \mathbf{X}_t , where $\mathbf{X}_t = (\mathbf{X}_t^1, \mathbf{X}_t^2, \dots, \mathbf{X}_t^N)$ and \mathbf{X}_t^k denotes the state of the k^{th} local patch of the object (*e.g.*, the position of the patch) among the N local patches used to describe the object. Then, if we denote the observations up to time t as $\mathbf{Y}_{1:t}$, the problem of object tracking can be defined as finding $\hat{\mathbf{X}}_t$ such that,

$$\hat{\mathbf{X}}_t = \arg \max_{\mathbf{X}_t} P(\mathbf{X}_t | \mathbf{Y}_{1:t}). \quad (5.1)$$

For sequential Bayesian inference, the posterior probability $P(\mathbf{X}_t|\mathbf{Y}_{1:t})$ is sequentially updated as the following:

$$P(\mathbf{X}_t|\mathbf{Y}_{1:t}) \propto P(\mathbf{Y}_t|\mathbf{X}_t) \int P(\mathbf{X}_t|\mathbf{X}_{t-1})P(\mathbf{X}_{t-1}|\mathbf{Y}_{1:t-1})d\mathbf{X}_{t-1}. \quad (5.2)$$

Here, $P(\mathbf{Y}_t|\mathbf{X}_t)$ is the likelihood between the current state \mathbf{X}_t and the current observation \mathbf{Y}_t , and $P(\mathbf{X}_t|\mathbf{X}_{t-1})$ is the transition probability from \mathbf{X}_{t-1} to \mathbf{X}_t .

Typically, for object tracking, since we consider many types of movements (*e.g.*, translation, rotation, scale, and affine motions), obtaining an exact analytic solution is not an easy task. Especially, in our formulation, it is more challenging since the dimension of the solution space increases as the number of local patches increases. Therefore as in (Isard & Blake, 1998), we use particle filtering (also known as sequential Monte Carlo sampling) to solve the problem. If we denote the l^{th} sample in particle filtering as $\mathbf{X}_{t,[l]}$, then (5.1) can be re-written as

$$\hat{\mathbf{X}}_t = \mathbf{X}_{t,[\hat{l}]}, \quad (5.3)$$

where

$$\hat{l} = \arg \max_l P(\mathbf{Y}_t|\mathbf{X}_{t,[l]}). \quad (5.4)$$

Note that since we are performing particle filtering, the likelihood of each particle $P(\mathbf{Y}_t|\mathbf{X}_{t,[l]})$ corresponds to the posterior probability. Thus, the problem of object tracking is now the problem of simulating the posterior distribution $P(\mathbf{X}_t|\mathbf{Y}_{1:t})$ with particle filtering, and then taking the particle with the best probability as a solution.

Our method differs from the traditional particle filtering methods due to the fact that the likelihood $P(\mathbf{Y}_t|\mathbf{X}_t)$ is obtained through an MRF-style manner. Through this MRF-style method, both the individual likelihood of each patch

and the relationship among them are maximized while tracking. The MRF-style elastic structure of local patches, which will be explained in Section 5.1.2, has an advantage that the resultant posterior distribution considers both the underlying local structures and the non-rigid deformations simultaneously. To allow the proposed method to perform the tracking procedure within the real-time constraint, we adopt a hierarchical diffusion scheme which benefits from the assumption that local deformation is not large between consecutive frames. Details of the proposed hierarchical diffusion is explained in Section 5.1.6.

5.1.2 Elastic Structure of Local Patches

In our work, we treat the target object as a collection of local parts, rather than treating the target object as a whole. Local parts are described with $n \times n$ size local patches, and local patches are assumed to be connected with nearby neighbors forming an elastic structure as in Figure 5.1. This model of the target object is realized using MRF. The likelihood of each local patch is considered to be the unary likelihood of the MRF, and the structure among them is considered to be the neighborhood relationship of the MRF. Since each local patch is connected with its neighbors forming an MRF, our model prefers solutions with the local structure of the target object preserved. Therefore, even if some of the patches are occluded, other un-occluded patches will drive occluded patches to the correct positions, causing the proposed model to be robust against partial occlusions. Also, since we describe the target object using local patches, we are able to represent non-rigid deformations as a collection of movements of local patches. We consider the initial patch positions and connections are given in the first frame. This initial setting can be given manually or automatically by some other detection system or by some certain strategy (*e.g.*, dividing the target bounding box into equal grids and considering each grids to be connected to its direct neighbors). The

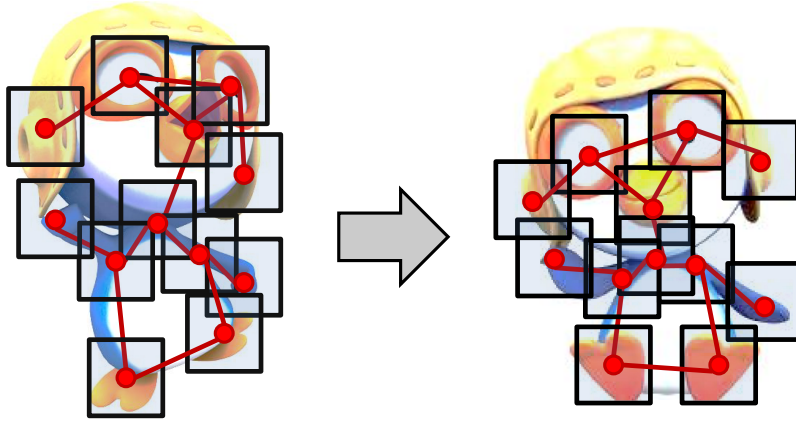


Figure 5.1: Example of elastic structure of local patches used to describe the target object. Black boxes denote each local patch and red lines denote each connection.

given patches should cover most of the target object with connections describing the structure of the target object.

The likelihood $P(\mathbf{Y}_t|\mathbf{X}_t)$ in (5.2) is modeled with the posterior probability of the MRF describing the structure of local patches. Therefore, $P(\mathbf{Y}_t|\mathbf{X}_t)$ is designed as

$$P(\mathbf{Y}_t|\mathbf{X}_t) \propto \prod_{k=1}^N \left[P(\mathbf{Y}_t|\mathbf{X}_t^k) \prod_{j \in N_k} P(\mathbf{X}_t^k|\mathbf{X}_t^j) \right], \quad (5.5)$$

where $P(\mathbf{Y}_t|\mathbf{X}_t^k)$ is the likelihood of a single patch, $P(\mathbf{X}_t^k|\mathbf{X}_t^j)$ is the prior probability describing the relationship among neighboring patches, and N_k denotes the neighbors of the k^{th} patch. Note that we are following the standard MRF configuration and are assuming conditional independence among patches which are not neighbors, as well as the independence among unary likelihoods of each patch $P(\mathbf{Y}_t|\mathbf{X}_t^k)$. In the energy form, if we denote the total energy of the configuration as $E(\mathbf{Y}_t; \mathbf{X}_t)$, the energy of a single patch as $E(\mathbf{Y}_t; \mathbf{X}_t^k)$, the energy from the relationship between patches as $E(\mathbf{X}_t^k, \mathbf{X}_t^j)$, we can write the total energy of the MRF model (which is simply the sum of the observation and neighborhood

energy of all patches) as

$$E(\mathbf{Y}_t; \mathbf{X}_t) \equiv Z + \sum_{k=1}^N \left[E(\mathbf{Y}_t; \mathbf{X}_t^k) + \sum_{j \in N_k} E(\mathbf{X}_t^k, \mathbf{X}_t^j) \right], \quad (5.6)$$

where Z is a normalizing constant. Here, the relationship between (5.5) and (5.6) is that $Probability \propto \exp(-\lambda Energy)$, assuming the Gibbs distribution. Here, λ is a design parameter controlling the smoothness of the posterior distribution. Now, with (5.6), (5.4) can be re-written as

$$\hat{l} = \arg \max_l P(\mathbf{Y}_t | \mathbf{X}_{t,[l]}) = \arg \min_l E(\mathbf{Y}_t; \mathbf{X}_{t,[l]}). \quad (5.7)$$

Also, the sample weights for particle filtering is now

$$w_{(l)} \propto P(\mathbf{Y}_t | \mathbf{X}_{t,[l]}) \propto \exp(-\lambda E(\mathbf{Y}_t; \mathbf{X}_{t,[l]})). \quad (5.8)$$

Generally speaking, if λ is large, the posterior distribution becomes spiky and particles become concentrated near the MAP solution, whereas if λ is small, the posterior distribution becomes smooth and more particles far away from the MAP survive the resampling process. We empirically found that $\lambda = 10$ work well in most cases.

5.1.3 Modeling a Single Patch

In order to obtain the energy of a single patch $E(\mathbf{Y}_t; \mathbf{X}_t^k)$, we model each individual patch using a linear classifier in 21 dimensional space. The first nine dimensions are HOG (Histogram of Oriented Gradients) features. We build our HOG by dividing the orientation into eight equal bins, and one bin to denote gradients with response 0. To obtain image gradients, filter kernels $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$

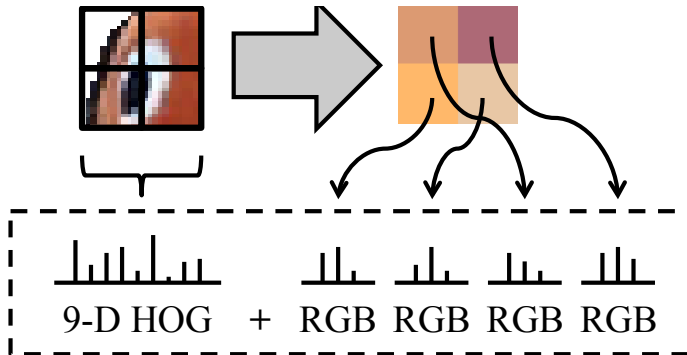


Figure 5.2: Example of a 21 dimensional feature descriptor for a single local patch.

and $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$ are used. When applying these filters, if the responses were below 10 on a 0 to 250 scale, we considered the response to be 0 to increase robustness to noise. We then assigned the image gradients to one of the eight bins according to their orientations, or the ninth bin if both filter responses were 0. For simplicity, we assigned each gradient to one of the nine bins without weighing them. For the remaining 12 dimensions, we divide a single local patch into four equal parts (upper left, upper right, bottom left, and bottom right) and obtain the mean RGB values for each sub region as in Figure 5.2 (3 dimension for each sub region). More sub regions may be used depending on the level of accuracy required for a single local patch tracking. This feature is similar to the one used in (Avidan, 2007), but one feature is assigned to a single patch not a single pixel as in (Avidan, 2007). The advantage of using this 21 dimensional feature is that this feature can be obtained efficiently using integral images.

For each patch, we use the classifier score of the observed 21 dimensional vector to obtain the energy of a single patch, $E(\mathbf{Y}_t; \mathbf{X}_t^k)$ in (5.6). The classifier is trained so that it gives high scores when the observation is likely to be the modeled patch, and gives low scores (possibly negative) when it is not likely. For the classifier, we use linear Support Vector Machine (SVM) (Cortes & Vapnik,

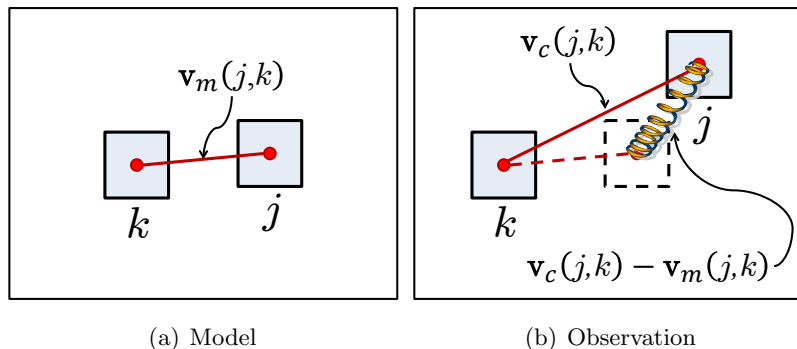


Figure 5.3: Example of neighboring local patches connected together.

1995; Fan et al., 2008) with logistic fitting performed on the classification score (Platt et al., 1999) to perform scaling. Training strategies for both linear SVM and logistic fitting are described in detail in Section 5.1.5. If we let \mathbf{f}_c^k denote the 21 dimension feature vector of the current observation for the k^{th} patch, let $s_k(\mathbf{f}_c^k)$ denote the linear SVM classification score for \mathbf{f}_c^k , and let A_k and B_k denote the learned logistic parameters, then

$$E(\mathbf{Y}_t; \mathbf{X}_t^k) = 1 - \frac{1}{1 + e^{A_k s_k(\mathbf{f}_c^k) + B_k}}. \quad (5.9)$$

Note that the logistic fitting (Platt et al., 1999) scales the classifier scores to be in range $[0, 1]$, considering the distribution of scores from the training data. This prevents the problem of certain patches having higher priority than others due to different score range when using raw classifier scores. We also use the classification result of each patch when updating the model to prevent drifting (detailed in Section 5.1.5).

5.1.4 Modeling the Relationship between Patches

The relationship between neighboring patches is modeled so that the local structures among neighboring patches are preserved while tracking. To deal with non-

rigid deformations, patches behave as if they are connected by springs. Also, to be robust to partial occlusions, the springs of each patch behave as if they are connected to the patch’s expected positions from its neighbors. As in Figure 5.3, if we consider an observed patch (patch j in Figure 5.3) and its neighbor (patch k in Figure 5.3), then in our model, the observed patch j tends to return to its expected position from its neighbor k (expected position is denoted by the dotted box) by the restoring force of a virtual spring connection between the expected position and patch j , which is length zero at rest. In other words, the energy of the connection between the two neighboring patches k and j is defined as the elastic energy of this spring. If we denote the vector difference between j^{th} and k^{th} patches of the current observation and the model as $\mathbf{v}_c(j, k)$ and $\mathbf{v}_m(j, k)$, respectively, then the displacement change x of this virtual spring is

$$x = \|\mathbf{v}_c(j, k) - \mathbf{v}_m(j, k)\|_2. \quad (5.10)$$

Also, to make close patches have more effect on one another, the strength of this spring is designed to be inversely proportional to the squared distance between the neighbors. Therefore, the spring constant κ is designed as

$$\kappa = \frac{2}{\|\mathbf{v}_m(j, k)\|_2^2} \beta, \quad (5.11)$$

where the neighbor strength β is a design parameter controlling the trade-off between the flexibility to adapt to non-rigid motion and the ability to keep the structure against occlusion. Details on the effect of β will be addressed in Section 5.2.1. Then, the elastic energy between connected local patches $E(\mathbf{X}_t^k, \mathbf{X}_t^j)$ in (5.6) is defined as

$$E(\mathbf{X}_t^k, \mathbf{X}_t^j) = \frac{1}{2} \kappa x^2 = \beta \frac{\|\mathbf{v}_c(j, k) - \mathbf{v}_m(j, k)\|_2^2}{\|\mathbf{v}_m(j, k)\|_2^2}. \quad (5.12)$$

5.1.5 Model Update

The model of the target object needs to be updated consistently in order for the tracker to be able to adapt to various changes in the target object. Illumination changes and deformations of the target object must be learned by the model to correctly evaluate (5.5). In our case, the model update is performed in two steps: (1) updating the linear classifiers and the logistic parameters for each patch, and (2) updating the neighborhood relationship. To prevent the model from drifting, the update is performed only when the observed patch is classified as the model, *i.e.* for patch k , only when $s_k(\mathbf{f}_c^k) > 0$. Also in case of the neighborhood relationship, we only update when the observation for both patches forming the relationship are classified as the model.

The way we update the linear classifiers is through updating training samples. For each patch, we keep a pool containing positive and negative training samples of size $2M$ (M positive samples and M negative samples). The positive samples represent the target object and the negative samples are simply the 21 dimension feature vectors drawn randomly from nearby. At the initial frame, we initialize the positive pools with M identical copies of the initial patches of the target object. For each frame, after obtaining the local patch tracking results, we add the tracking results to the positive pools and take out the oldest samples from the positive pools. When taking samples out from the positive pools, to prevent drifting, we make sure that at least one sample is from the first frame (*i.e.* one sample representing the patch from the initial frame is never taken out for each pool). Then we completely discard the previous negative pools and refresh negative samples from random nearby patches. Again, for each patch, the classifier from the previous frame is discarded and a new classifier is trained using the new training pools. Note that for each pool, since we update one positive sample at a time, the pool size M acts as a design parameter controlling the update speed

of the target object model. When M is large the model is updated slowly and when M is small the model is updated quickly. In general, we empirically found that $M = 100$ gives good performance as well as low computational cost for the update process.

For the relationship update, we simply update $\mathbf{v}_m(j, k)$ by weighted averaging, but only when both patches are classified as the model. In other words, for all j and k , if

$$\left[s_j \left(\mathbf{f}_c^j \right) > 0 \right] \wedge \left[s_k \left(\mathbf{f}_c^k \right) > 0 \right], \quad (5.13)$$

then

$$\mathbf{v}_m(j, k) \leftarrow \frac{1}{M} \mathbf{v}_c(j, k) + \left(1 - \frac{1}{M} \right) \mathbf{v}_m(j, k). \quad (5.14)$$

Note that the learning rate is $\frac{1}{M}$ so that the update rate would be the same as for the training pools for the classifiers.

5.1.6 Hierarchical Diffusion

In our model, the dimension of the solution space is too large to apply simple motion models such as the random-walk model. For instance, if we were to need 100 particles to track an object with only one patch using the random-walk motion model, then with N patches we would require 100^N particles to track the target object with our model. This would require an infeasible amount of calculation, making our method impossible to run in real-time. Therefore, we propose an efficient hierarchical diffusion method.

To use a small number of samples, we focus on sampling from the region where the actual solution would exist with high probability. In case of tracking situations the deformation of the target object is not large between subsequent frames. Considering this as an assumption, we diffuse particles hierarchically in two steps: globally for the motion of the whole object and locally for the deformations

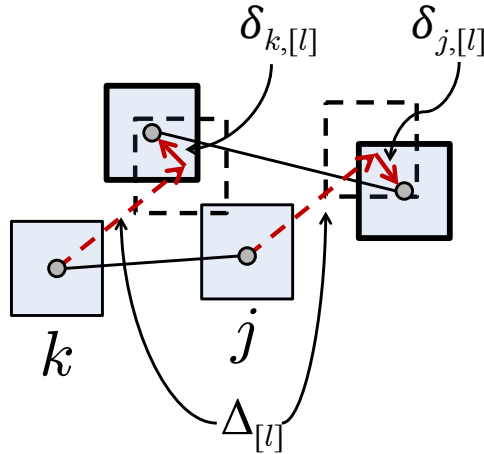


Figure 5.4: Illustrative example of hierarchical diffusion performed for a single sample l . Global movement of the total configuration of patches $\Delta_{[l]}$ is sampled first, then local movements of individual patches $\delta_{k,[l]}$ and $\delta_{j,[l]}$ are sampled.

of the target object. We first diffuse all local patches equally according to the Gaussian distribution with a relatively large variance, and then diffuse each patch separately according to the Gaussian distribution with a relatively small variance (illustrated in Figure 5.4). In the global step, the samples are diffused so that the relative positions between local patches in the sample are preserved. Then, in the local step, each local patch is diffused independently. Mathematically the proposed hierarchical diffusion method can be described as

$$\mathbf{X}_{t,[l]}^k = \mathbf{X}_{t-1,[l]}^k + \Delta_{[l]} + \delta_{k,[l]}, \quad (5.15)$$

where, $\mathbf{X}_{t,[l]}^k$ denotes the state of the k^{th} local patch of the l^{th} sample at time t , $\Delta_{[l]}$ denotes the 2-dimensional global diffusion (translation in x, y direction for the whole object) for the l^{th} sample, and $\delta_{k,[l]}$ denotes the 2-dimensional local diffusion (translation in x, y direction for a local patch) for the k^{th} local patch of

the l^{th} sample. Here,

$$\Delta_{[l]} \sim \mathcal{N}(0, \sigma_G^2), \quad (5.16)$$

and

$$\delta_{k,[l]} \sim \mathcal{N}(0, \sigma_L^2), \quad (5.17)$$

where $\mathcal{N}(0, \sigma^2)$ denotes a Gaussian distribution with zero mean and standard deviation σ . σ_G and σ_L are parameters for the diffusion. The optimal choice of σ_G and σ_L may vary depending on the image sequence but we empirically found that $\sigma_G = 8$ and $\sigma_L = 4$ works well for most cases. The proposed diffusion produces an accurate solution with a relatively less number of particles than the simple random walk approach, which allows the proposed method to achieve real-time performance. Details and discussion on experimental results regarding the effectiveness of hierarchical diffusion are given in Section 5.2.8.

5.1.7 Summary of the Proposed Method

The proposed method uses particle filtering to get an MAP solution for the object tracking problem. Given the initial patches and connections $\{\mathbf{f}_m^k, \mathbf{v}_m(j, k); \forall j, k, \}$, the proposed method can be summarized as **Algorithm 1**.

5.2 Experiments

5.2.1 Parameter Effects

The parameter β in subsection 5.1.2 controls the strength of the neighborhood connections. In other words, large β means that tracking is performed so that the local structure does not change much. In other words, β is a parameter controlling the tradeoff between the tracker’s ability to track non-rigid objects and ability to

Algorithm 1 Tracking with Local Patches (for each frame)

1: For each sample, compute

$$E(\mathbf{Y}_t; \mathbf{X}_t) \equiv Z + \sum_{k=1}^m \left[E(\mathbf{Y}_t; \mathbf{X}_t^k) + \sum_{j \in N_k} E(\mathbf{X}_t^k, \mathbf{X}_t^j) \right] \quad ((5.6))$$

where,

$$E(\mathbf{Y}_t; \mathbf{X}_t^k) = 1 - \left(1 + e^{A_k s_k(\mathbf{f}_c^k) + B_k} \right)^{-1} \quad ((5.9))$$

$$E(\mathbf{X}_t^k, \mathbf{X}_t^j) = \beta \frac{\|\mathbf{v}_c(j,k) - \mathbf{v}_m(j,k)\|_2^2}{\|\mathbf{v}_m(j,k)\|_2^2} \quad ((5.12))$$

2: Find MAP solution

$$\hat{\mathbf{X}}_t = \mathbf{X}_{t, [\hat{l}]} \quad ((5.3))$$

where,

$$\hat{l} = \arg \max_l P(\mathbf{Y}_t | \mathbf{X}_{t, [l]}) = \arg \min_l E(\mathbf{Y}_t; \mathbf{X}_{t, [l]}) \quad ((5.7))$$

3: Update object model (Section 5.1.5)

4: Assign sample weights $w_{[l]}$ according to the likelihood

$$w_{[l]} \propto P(\mathbf{Y}_t | \mathbf{X}_t) \propto \exp(-\lambda \times E(\mathbf{Y}_t; \mathbf{X}_{t, [l]})) \quad ((5.8))$$

5: Re-sample according to weights

6: Diffuse samples

$$\mathbf{X}_{t+1, [l]}^k = \mathbf{X}_{t, [l]}^k + \Delta_{[l]} + \delta_{k, [l]} \quad ((5.15))$$

cope with partial occlusions. Figure 5.5 is an example showing the effect of the β parameter when tracking object with deformation. As in (b) if parameter β is small, then object deformation is well tracked. On the other hand, if β is large, then deformation is less accounted for. For our method, β is the only parameter which requires tuning. In case of other parameters, we found empirically that the parameters noted in the beginning of Section 5.2.2 works well in most situations and does not require tuning. Thus, unlike the case for traditional particle filtering methods (Ross et al., 2008; Pérez et al., 2002; Kwon et al., 2009; Mei & Ling, 2009) which require the tuning of the diffusion parameters in each dimension when changing the tracker’s behaviors, our method only requires tuning of β . Also, tuning β is intuitive and simple. Figure 5.6 shows the effect of parameter β when tracking objects with partial occlusions. The tracker better handles occlusions when β is large as in (c). Generally, for highly deformative scenes $\beta = 0.2$ shows good results, for scenes with heavy occlusion $\beta = 2.0$, and normally $\beta = 1.0$ is

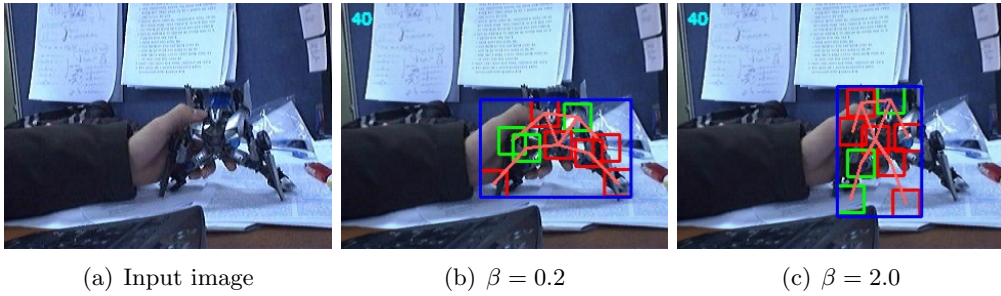


Figure 5.5: Example showing the effect of β parameter on non-rigid object tracking. Tracking results for the **Robot** sequence, frame #40, with $\beta = 0.2$ (b) and $\beta = 2.0$ (c).

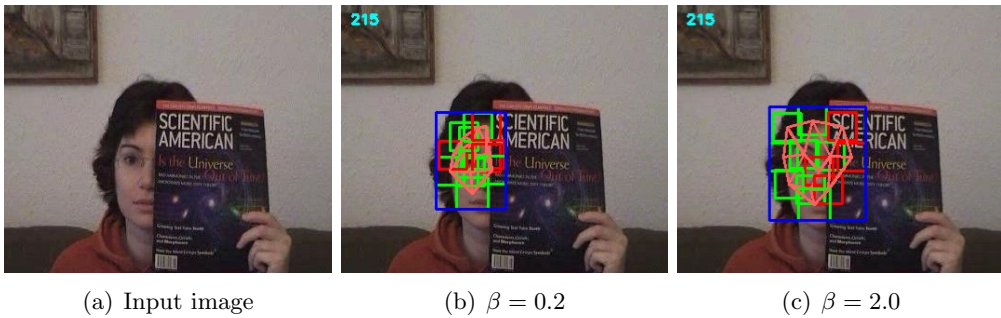


Figure 5.6: Example showing the effect of β parameter on tracking objects with partial occlusions. Tracking results for the **Face** sequence, frame #215, with $\beta = 0.2$ (b) and $\beta = 2.0$ (c).

good enough.

5.2.2 Performance Evaluation

Evaluation of the proposed method was performed through twelve image sequences. Each image sequence consists of different types of situations (occlusion, outer-plane motion, non-rigid deformation, etc.) Throughout the experiments, all parameters except β were fixed. The pool size M was set to 100, and the number of particles was set to 1000. For the linear SVM, we used default parameters provided in (Fan et al., 2008). For the sampling parameters, $\sigma_G = 8$ and $\sigma_L = 4$. Parameter λ controlling the concentration of samples, was set to 10. The implementation was done in C++. All experiments were held on a 3.0GHz desktop and ran comfortably about 20-50 frames per second depending on the number of patches (except for the **Dudek** sequence which was of size 720×480 , and ran 12 frames per second).

The test sequences are composed of some well-known sequences and some of our own. The **Dudek** sequence and the **Sylvester** sequence are from the work of Lim et al. (Ross et al., 2008), and the **Face** sequence and the **Woman** sequence are from the (Adam et al., 2006). The **Caviar** sequence is from the CAVIAR¹ dataset. These five sequences are some of the well-known sequences for evaluating tracking performances. The **High Jump** sequence is from Kwon et al.'s work (Kwon & Lee, 2009), and the **Motocross 1** and the **Mtn. Bike** sequences are from recent work by Godec et al. (Godec et al., 2013). The **Robot** sequence and the **Pedestrian** sequence are from (Yi et al., 2010), and the **Dove** sequence is from (Yin et al., 2011). Finally, the **Nemo** sequence is of our own.

For quantitative analysis, we compared the mean error of the four corner

¹EC Funded CAVIAR project/IST 2001 37540, found at URL: <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>

points of the tracking bounding box. The ground truth data was annotated by human hand, so that the target object fitted in the bounding box. In (Godec et al., 2013), Godec et al. used the Agarwal-criterion (Agarwal et al., 2004), which was defined as $score = \frac{R_T \cap R_{GT}}{R_T}$, where R_T is the tracking rectangle and R_{GT} the ground truth, but this measure is not suitable for describing how accurate tracking is. A single dot in the ground truth region would return maximum measure. However, by using the mean of the errors of the four corner points of the tracking bounding box we can measure the accuracy of a tracker without such problem. Also, this measure is easily applicable to many trackers since most trackers are based on giving a bounding box of the target object as a result.

The proposed method has been compared against seven other trackers. Beyond Semi-supervised Tracking (BEYOND) in (Stalder et al., 2009) and Multiple Instance Learning (MIL) in (Babenko et al., 2011) are methods based on the concept “tracking by detection”, Frag-Track (FRAG) in (Adam et al., 2006) and l_1 minimization (L1) in (Mei & Ling, 2009) are some representative methods for solving occlusion problems, Basin Hoppin Monte Carlo Tracking (BHMC) in (Kwon & Lee, 2009) and Hough-based Tracking (HOUGH) in (Godec et al., 2013) are state-of-the-art methods for solving non-rigid object tracking, and TLD Tracking (TLD) (Kalal et al., 2010) is a method which uses both detectors and trackers. For the experiments, we used the implementation provided by the authors of each paper. Also, we implemented our method in three different ways. The first is with manual initialization (LPT), the second is with initialization by dividing the target bounding box into equal grids (LPT_GRID), and the last one is applying simple temporal low-pass filtering to LPT (LPT_SMOOTH). For LPT_GRID, we assumed the patches were connected to its direct neighbors (patches which share boundaries) and the number of grids was set to 3×3 . For LPT_SMOOTH, the temporal smoothing was performed by weighted averaging

the new estimated result and the old estimate (the tracking result from previous frame). When averaging, we applied different weights for the smoothing of the center position and the smoothing of the width and height. For the position, 0.3 weight was applied to the new estimate and 0.7 for the previous estimate. For the width and height, we applied stronger smoothing than we did for the position since width and height do not change much between consecutive frames; 0.1 weight to the new estimate and 0.9 to the previous estimate. All other parameters were identical for all three implementations.

Figure 5.7 shows the mean error value of each tracker for each image sequence. β was set as in the sub-captions. The mean error value was calculated only for the frames the tracker returned results. This is because BEYOND and TLD returned results only when they are confident, and if they are not, returned results indicating tracking failures. The number on top of each marker denotes the percentage of frames that gave meaningful results, which mean that, if we denote the mean error of the four corner points as e_{mean} , width of the ground truth as w_{GT} , and height of the ground truth as h_{GT} , then $e_{mean} < \min(w_{GT}, h_{GT})$. The tracker with lowest mean error and with over 90% of the tracking results meaningful is marked with red bold text. The red dotted horizontal line denotes the mean value of $\min(w_{GT}, h_{GT})$ throughout each sequence. Tracker with mean error above the dotted red line means that most of the tracking results from that tracker were meaningless for that sequence. In general, all three implementations of our method consistently outperform or show comparable results against other compared methods. Note that the compared methods sometimes give better results than ours depending on the image sequence, but our method consistently gives good results, regardless of the image sequence used.

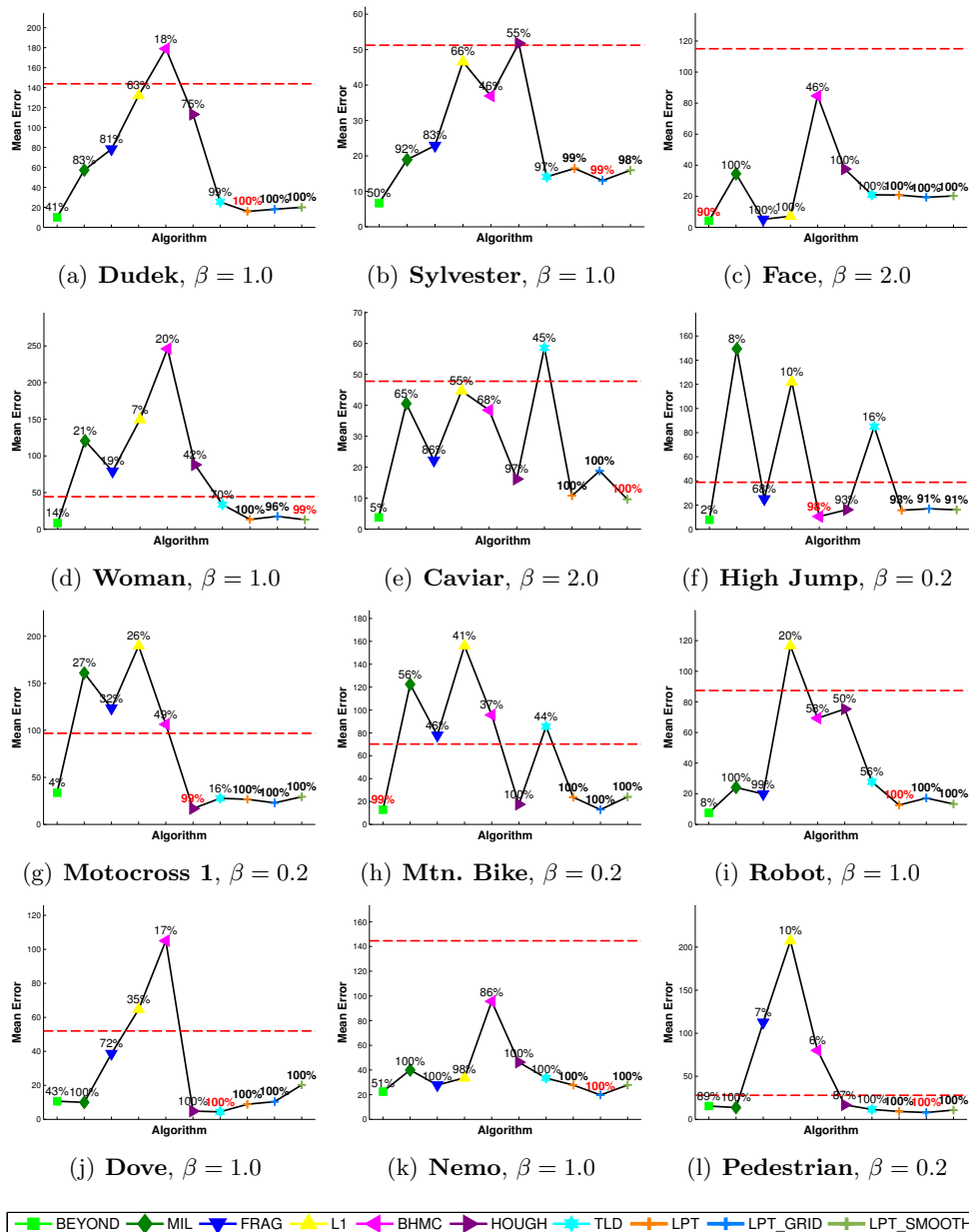


Figure 5.7: Mean errors for each sequence. Best tracker denoted by red bold text.

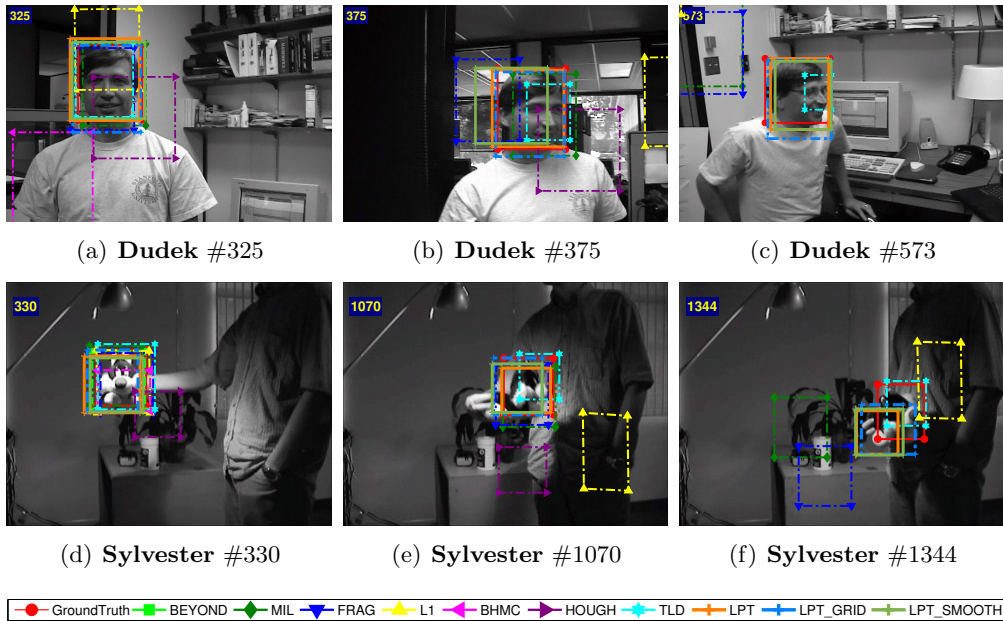


Figure 5.8: Tracking results for the **Dudek** sequence and the **Sylvester** sequence. *Best viewed in color.*

5.2.3 Discussion on Translation, Rotation, Illumination Changes

The **Dudek** sequence and the **Sylvester** sequence are well known datasets for testing robustness on translation, rotation, illumination changes. In both image sequences, BEYOND gave the most accurate result (Figure 5.7 (a) and (b)). However, in both sequences, BEYOND was able to track less than half of the whole sequence, whereas our method (LPT) was able to track most of the sequence (100% for **Dudek** and 99% for **Sylvester**) with promising results. For the **Sylvester** sequence, LPT_GRID and LPT_SMOOTH show similar results. Critical frames for the two sequences are shown in Figure 5.8. In Figure 5.8 (a), as the target person stands up, L1, BHMC, and HOUGH loses track. Also, as the person turns around in (b), FRAG fails whereas our method successfully tracks the whole sequence. In (b), LPT_SMOOTH also drifts a bit due to the abrupt

change in motion, but still keeps track of the target object and recovers after a few frames. In (d), we can see that HOUGH starts to drift off. The authors of (Godec et al., 2013) reported that they were able to track 99% of the image sequence, but even with same initial conditions the authors provided, we were not able to reproduce their result (slightly degraded result). We suspect that this is because HOUGH uses fully randomized trees, meaning that the accuracy of their algorithm may vary according to the random seed. For all image sequences, even with the same initial settings, HOUGH returned various results, making the tracking accuracy unstable. In (e) and (f), our method (LPT) successfully keeps track of the target object but with relatively inaccurate results, due to the large outer-plane motion of the target object.

5.2.4 Discussion on Partial Occlusions

The sequences **Face**, **Woman**, and **Caviar** are sequences which contain partial occlusions. For the **Face** sequence, all three implementations of our method gave comparable results against other compared methods (Figure 5.7 (c), (d), and (e)). Moreover, our method was also capable of tracking objects showing non-rigid deformations, whereas methods showing good performances on this sequence (BEYOND, FRAG, and L1) did not show good performances in other situations (**High Jump**, **Motocross 1**, and **Mtn. Bike**). Also, note that the methods designed for non-rigid object tracking (BHMC and HOUGH) do not show good results for **Face** and **Woman**. For the **Face** and **Woman** sequences, the target object is occluded gradually and severely, where some parts of the sequence have over half of the target occluded. Critical frames for these sequences are shown in Figure 5.9. In Figure 5.9 (b), as occlusion occurs, we can see other methods failing, whereas our method, LPT, LPT_GRID, and LPT_SMOOTH all three, successfully tracks. As in (c), TLD re-detects the target object when the occlusion

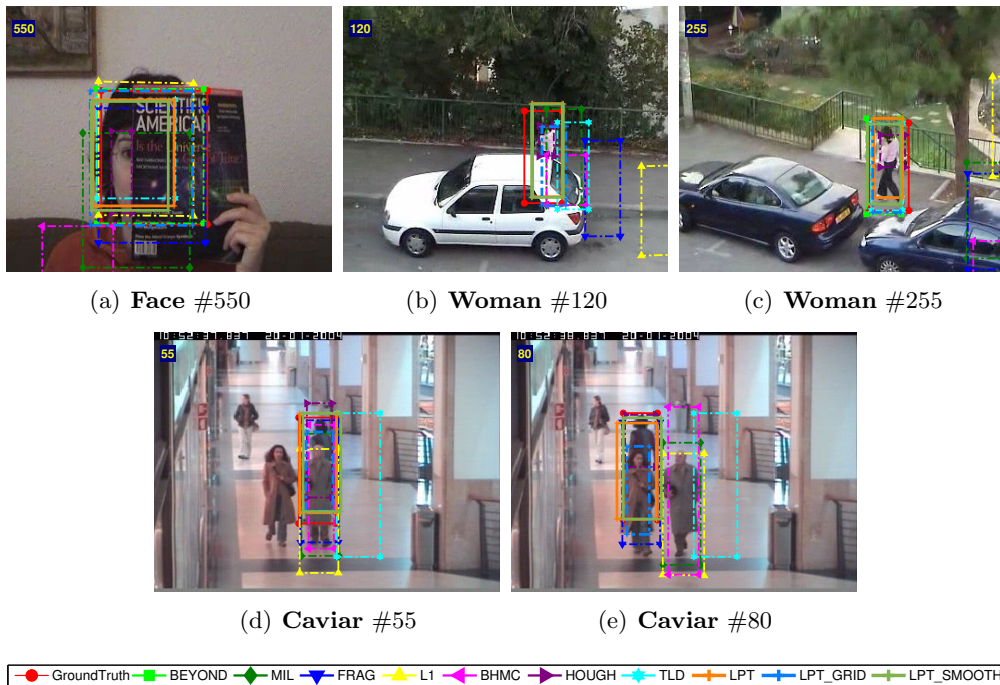


Figure 5.9: Tracking results for the **Face** sequence, the **Woman** sequence, and the **Caviar** sequence. *Best viewed in color.*

is gone, but as the target object gets occluded again, the tracker fails. This sequence was used in (Adam et al., 2006), and FRAG was able to track the target object throughout the whole sequence in their paper. However, in (Adam et al., 2006), only a portion of the whole sequence was used. When started from the first frame, FRAG loses track as well.

5.2.5 Discussion on Non-Rigid Deformations

The sequences **High Jump**, **Motocross 1**, and **Mtn. Bike** are sequences with non-rigid deformations. For all sequences, all three implementation of our method gave promising results (Figure 5.7 (f), (g), and (h)). BHMC and HOUGH show good results for objects with non-rigid deformations, but do not show good results in general (especially for sequences with occlusions) and run only a few frames per second, whereas our method runs 20 to 50 frames per second. Note that the trackers showing good performance against partial occlusions (FRAG and L1) tend to show unsatisfactory results in these cases, whereas our method consistently gives good results. Critical frames for these sequences are shown in Figure 5.10.

5.2.6 Discussion on Additional Cases

The **Robot** sequence contains both non-rigid deformations and partial occlusions. The **Dove** and the **Pedestrian** sequences show fast and abrupt movements. Finally, the **Nemo** sequence has scale changes and in-plane rotations. Some critical frames for these sequences are shown in Figure 5.11. For the **Robot** sequence, as in Figure 5.7 (i), the proposed method outperformed all other trackers (BEYOND showed lower mean error, but was able to track only 7.7% of the sequence). Also, for the sequences **Dove**, **Pedestrian**, and **Nemo**, both LPT and LPT_GRID

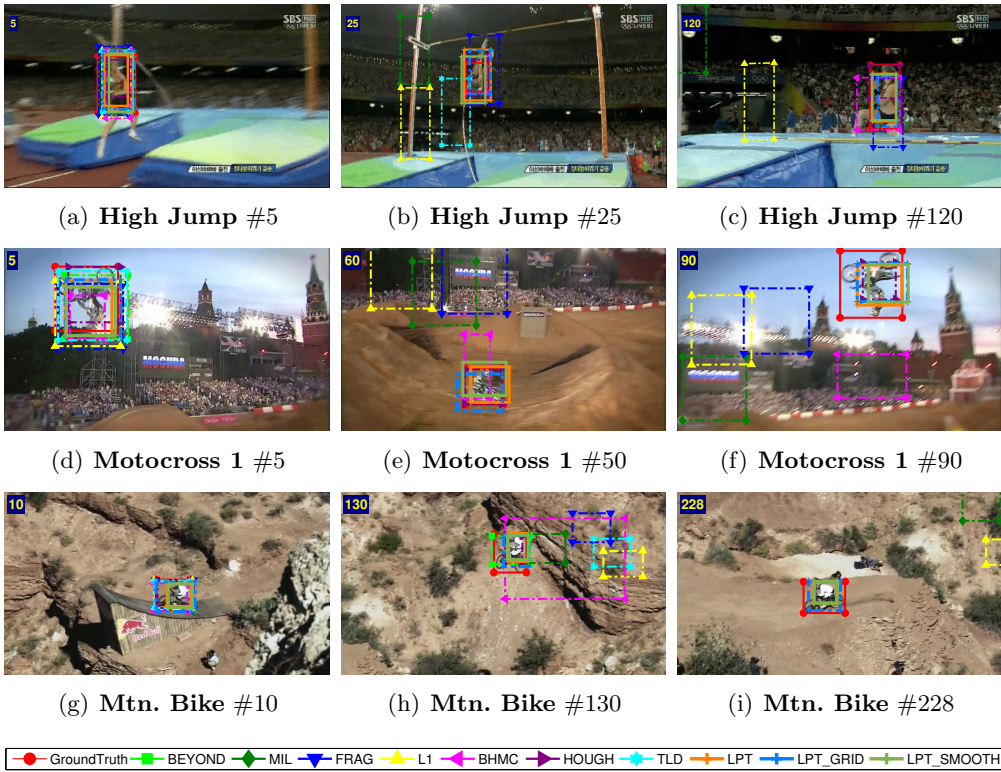


Figure 5.10: Tracking results for the **High Jump** sequence, the **Motocross 1** sequence, and the **Mtn. Bike** sequence. *Best viewed in color.*

Table 5.1: The mean error values and the percentage of meaningful tracking results with all frames in all image sequences for each algorithm. Red underlined text indicates best result and the plain blue text indicates second best.

	Mean Error	% Meaningful
BEYOND	<u>10.10</u>	48.43
MIL	48.57	82.11
FRAG	43.51	73.04
L1	88.00	54.11
BHMC	97.80	42.93
HOUGH	53.53	74.57
TLD	26.52	83.86
LPT	17.03	<u>99.48</u>
LPT_GRID	<u>16.03</u>	99.14
LPT_SMOOTH	17.85	<u>99.33</u>

Table 5.2: The mean error and the percentage of meaningful tracking results for each algorithms with all sequences. Red underlined text indicates best result and the plain blue text indicates second best.

outperformed or showed comparable results against other trackers. In case of LPT_SMOOTH, the accuracy degraded for the **Dove** sequence due to the fast movement of the target object (example shown in Figure 5.11 (g)), but is still comparable to other trackers. Note that in (a), (c), and (f), the toy shows complex movements (spreading legs apart, bending, and sitting down), which other trackers fail to describe. Since the neighborhood connections in the grid initialization are not accurate, LPT_GRID also fails to describe the movement of the target object in this case. In (d) and (e), LPT shows robust performance against severe partial occlusions.

5.2.7 Summary of Tracking Results

The evaluation results are summarized in Table 5.2. As in Table 5.2, our method (LPT) outperforms other compared trackers with respect to the percentage of

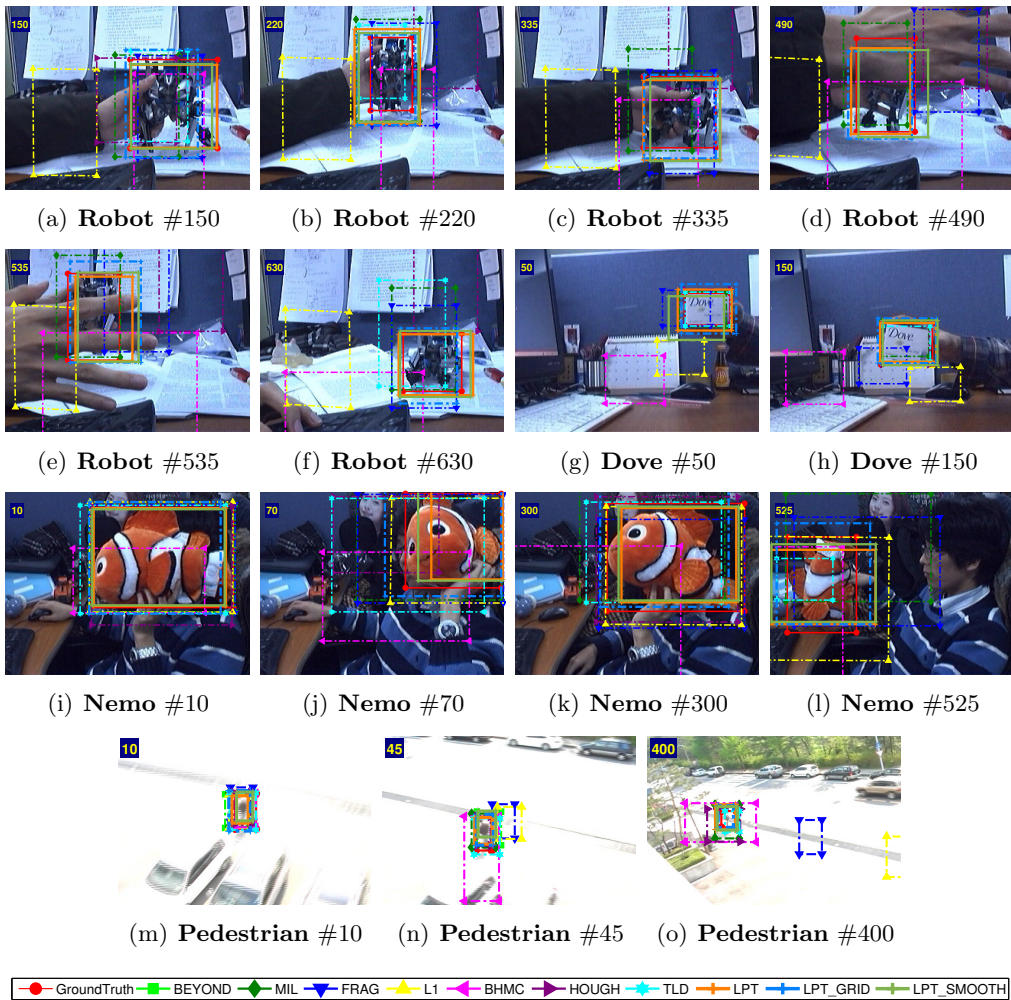


Figure 5.11: Tracking results for the **Robot** sequence, the **Dove** sequence, the **Pedestrian** sequence, and the **Nemo** sequence. *Best viewed in color.*

meaningful tracking results. Except for BEYOND, our method (LPT_GRID) also outperforms all other compared method in terms of mean error as well. BEYOND showed the lowest mean error, but showed the worst result when considering the percentage of meaningful tracking results. Note that as shown in Figure 5.7, our method may perform slightly worse than the compared methods depending on the image sequence, but when all frames in all sequences are considered, our method outperforms all compared methods. This means that our method gives consistent results among all sequences. Note that the grid initialization version of our method, LPT_GRID, shows similar results as LPT. This is because in many tracking situations, the local structure of the target object is preserved, and the grid configuration is accurate enough for the tracker to work. Also, LPT_SMOOTH shows slightly degraded performance, but gives more stable results than LPT (see Section. 5.2.9 for details on the stability of the estimated tracking result).

5.2.8 Effectiveness of Hierarchical Diffusion

To demonstrate the effectiveness of hierarchical diffusion, we have compared the mean error of the tracking results obtained with hierarchical diffusion and with simple Gaussian diffusion (the random-walk motion model). During the experiment, to compare only the effect of different diffusion strategies, all parameters including the number of particles were fixed except for the diffusion parameters. For hierarchical diffusion, $\sigma_G = 8$ and $\sigma_L = 4$, whereas for simple Gaussian diffusion, we varied σ from $\sigma = 0.5$ to $\sigma = 10$ having 0.5 as step size. Comparison results are shown in Figure 5.12. In Figure 5.12, results of hierarchical diffusion are denoted by the solid black lines, the mean results of simple Gaussian diffusion are denoted by the grey dashed lines, and the range in which results of simple Gaussian diffusion lie in are denoted with the grey area. As shown in Figure 5.12, the proposed hierarchical diffusion generally gives better tracking results with the

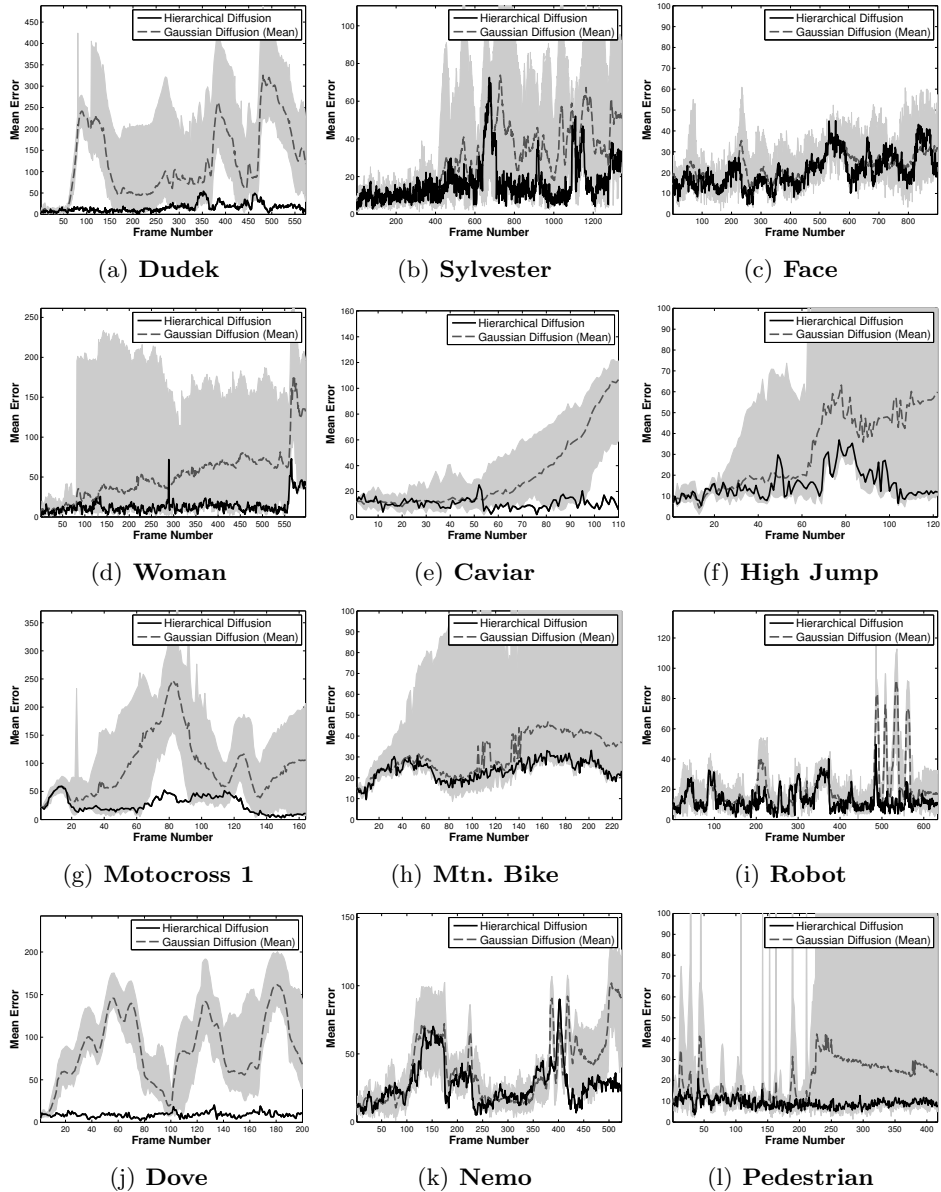


Figure 5.12: Mean error obtained using hierarchical diffusion and simple Gaussian diffusion. See text for details.

same number of particles. Note that there are some frames which simple Gaussian diffusion provides better results, such as in (i) and (k). These are special cases when the global motion of the target object is not large. However this is not a common case and when all sequences are considered, hierarchical diffusion achieves lower mean error. Also, note that the same diffusion parameters were used for all sequences.

5.2.9 Limitations

Though our method outperforms other methods in terms of mean error and the percent of meaningful frames, there are some limitations to the proposed method. The first limitation is the accuracy of individual local patches. The accuracy of individual patches may not be as good when large deformations exist or when the local patch is not very distinctive from its surroundings. However, when all local patches are considered together, the overall tracking result for the whole object is quite accurate owing to the proposed elastic structure. Example tracking results and their local patch structures are shown in Figure 5.13. As shown in Figure 5.13 (a) - (c), the position of each individual patch is quite accurate when the patches are discriminant from its surroundings. In (d) - (f), the patch near the head of the person drifts and the position of the patches is not as accurate as in (a) - (c) due to large deformation and fast motion. An extreme case for the individual patch accuracy limitation is shown in Figure 5.13 (g) - (l). In this hand tracking sequence, the target object is highly deformative and the local patches look similar to their surroundings. As a result, patches in the lower part of the hand drift. However, note that the majority of the hand is still tracked when considering the entire configuration (the blue rectangle), which is one of the advantages of the elastic structure.

The second limitation is related to the robustness of the proposed method

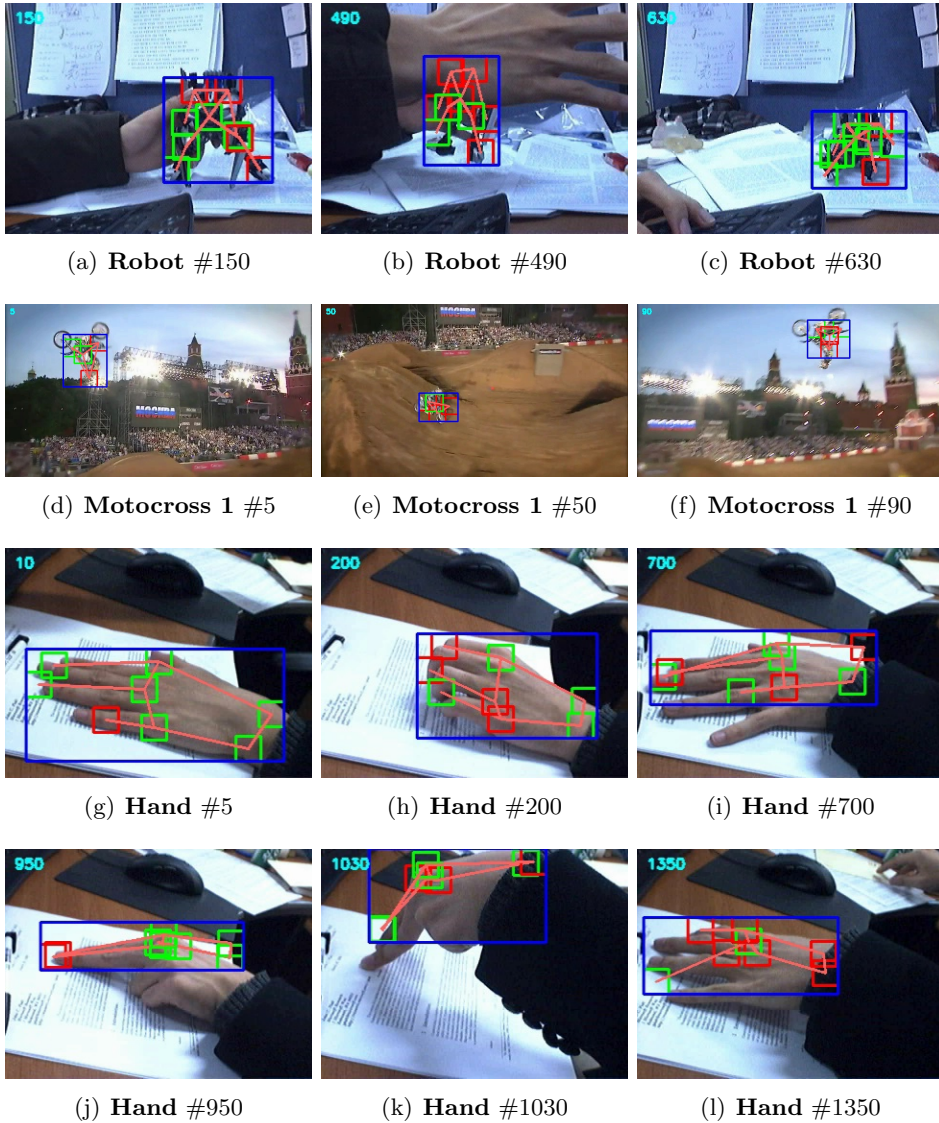


Figure 5.13: Example tracking results and their local patch structures. Blue rectangle is the final tracking result, green rectangles are local patches with high confidence (SVM score above 0), red rectangles are local patches with low confidence, and the orange lines denote the neighborhood relationships. *Best viewed in color.*

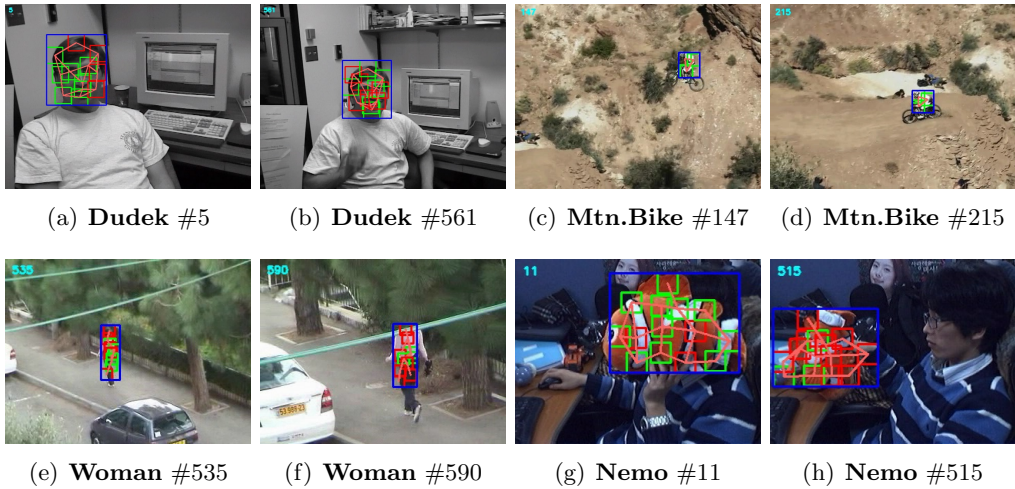


Figure 5.14: Example tracking results with scale and orientation changes. Blue rectangle is the final tracking result, green rectangles are local patches with high confidence (SVM score above 0), red rectangles are local patches with low confidence, and the orange lines denote the neighborhood relationships. *Best viewed in color.*

against scale and orientation changes. The proposed method models the movement of individual patches considering translational movements only. In general, this does not cause major problems since minor scale and orientation changes can be described as the change in elastic structure. Example of the proposed method adapting to minor scale change is shown in Figure 5.14 (a) and (b), and example of the proposed method tracking a target object with minor orientation change is shown in Figure 5.14 (c) and (d). As shown in the examples, small scale and orientation changes are explained as change in the elastic structure. Also, even if the structure fails to describe the change, a few local patches with good matching scores are enough to track the target object. Figure 5.14 (e) - (h) are examples of such failures. In Figure 5.14 (e) and (f), the target object undergoes a drastic scale change as the camera zooms in. Our method fails in adapting to the fast scale change, but still does not lose track of the target object. In Figure 5.14 (g) and

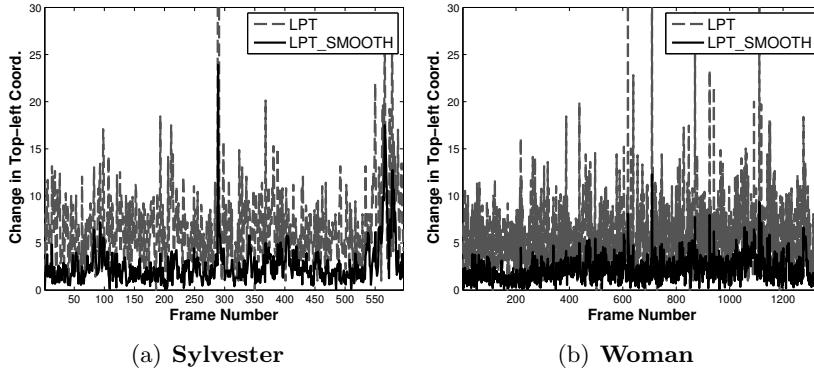


Figure 5.15: Change in the coordinates of the top-left corner point of the estimated bounding box for each frame. Note that the smoothed version (LPT_SMOOTH, solid black line) shows much less change than the original (LPT, gray dashed line) in each frame.

(h), the target object undergoes large in-plane orientation change. Our method also fails to accurately describe the target object motion in this case. However, note that in both cases, though the accuracy of the estimate may decrease, our method does not lose track of the target object, owing to a few patches with strong matches directing the entire structure to the correct position.

The last limitation is the noisy nature of the estimated tracking result. Since our problem space is high-dimensional, even with an efficient diffusion scheme, the number of particles is relatively scarce when trying to obtain a real-time solution. As a result, we end up with noisy estimates, since particles are relatively positioned sparsely. As shown in Figure 5.15 with the dashed gray lines, this leads to abrupt changes in the estimated position. However, as shown in Table 5.2, when all frames are considered, the results are good with small mean error on average. Thus, we can simply apply a temporal low-pass filtering to reduce the noise in the estimate, which is LPT_SMOOTH. As shown in Figure 5.15 with the black solid lines and overall performance in Table 5.2, this simple low-pass filtering reduces the abruptness in the tracking result greatly without much harm in the

performance.

5.3 Remarks and Discussion

A new tracking method based on sequential Bayesian inference has been proposed. The proposed method tackled both the problem of partial occlusions and non-rigid deformation when tracking objects, by modeling the target object with an elastic structure of local patches, and by performing hierarchical diffusion in the solution space. By modeling the target object with an elastic structure of local patches, the proposed method was able to track objects with partial occlusions and non-rigid deformations. Also, through hierarchical diffusion, the tracking procedure was performed in real-time on a desktop PC. The method was evaluated against state-of-the-art trackers through twelve image sequences with large occlusions and non-rigid deformations. The experimental results showed that the proposed method outperformed other compared methods, and demonstrated the robustness of the proposed method against various situations including partial occlusion, non-rigid motion, abrupt motion, translation, rotation, and illumination change.

As shown in the experiments, the proposed method showed good performance even with simple initialization strategy. However, with better initialization, the performance of our method would be enhanced. Therefore, providing sophisticated initializations would be one of the most beneficial directions for future research. Recently, detecting and recognizing objects with part-based models have drawn much interest (Mikołajczyk et al., 2004; Felzenszwalb et al., 2010). As a result, importance of part-based tracking methods is increasing. Incorporating part-based detection and recognizing methods for the initialization of our method would be a promising way to enhance initialization.

Chapter 6

Concluding Remarks and Future Works

In this thesis, we have proposed bio-mimetic schemes for motion detection and visual tracking to overcome the limitations of existing methods in actual environments. The methods were inspired from the theory that four different forms of visual memory exists for human visual representations of a scene; visible persistence, informational persistence, visual short-term memory (VSTM), and visual long-term memory (VLTM). We tackled the vision inference problem as modeling and representing the observed scene with the temporary short-term models (TSTM) and the conservative long-term models (CLTM). We have studied on building efficient and effective models for TSTM and CLTM, and utilized them together to obtain robust detection and tracking results under occlusions, clumsy initializations, background clutters, drifting, and non-rigid deformations encountered in actual environments. We have also combined the key ideas of our proposed methods to form a new method for detection and tracking giving accurate pixel-wise results.

First, to reduce the computation load required for motion detection on non-stationary cameras, we proposed an efficient representation of TSTM. To achieve real-time capability with satisfying performance, the proposed method modeled the background through dual-mode kernel model and compensated the motion of the camera by mixing neighboring models. Modeling through dual-mode kernel model prevented the background model from being contaminated by foreground pixels, while still allowing the model to be able to adapt to changes of the background. Mixing neighboring models reduced the errors arising from motion compensation and their influences were further reduced by keeping the age of the model. Also, to decrease computation load, the proposed method applied the proposed modeling grid-wise rather than pixel-wise without performance degradation.

Second, for tracking, to solve the problems of occlusions, background clutters, and drifting simultaneously with the new tri-model using both TSTM and CLTM. The proposed tri-model was composed of three models, where each model learned the target object, the background, and other non-target moving objects online. The proposed method performed tracking by finding the best explanation of the scene with the three learned models. By utilizing the information in the background and the foreground models as well as the target object model, our method obtained robust results under occlusions and background clutters. Also, the target object model was updated in a conservative way to prevent drifting. Furthermore, our method was not restricted to bounding-boxes when representing the target object, and was able to give pixel-wise tracking results.

Third, a feature-point based tracking method using both TSTM and CLTM to track objects in case of uncertain initializations and severe occlusions was proposed. To track objects accurately in such situations, the proposed method used “motion saliency” and “descriptor saliency” of local features and performed track-

ing based on generalized Hough transform (GHT). The proposed motion saliency of a local feature utilized instantaneous velocity of features to form TSTM and emphasized features having distinctive motions. The descriptor saliency modeled local features as CLTM and emphasized features which are likely to be of the object in terms of its feature descriptors. Through these saliencies, the proposed method tried to “learn and find” the target object rather than looking for what was given at initialization, thus being insensitive to initialization problems.

Fourth and last, we focused on solving both the problem of tracking under partial occlusions and the problem of non-rigid object tracking in real-time. We have built robust CLTM with local patches and their neighboring structures based on sequential Bayesian inference. The proposed method was mainly composed of two parts. The modeling part which the target object was modeled using elastic structure of local patches, and the solver part which employed efficient hierarchical diffusion method to perform the tracking process in real-time. The elastic structure of local patches allowed the proposed method to handle partial occlusions and non-rigid deformations and the proposed hierarchical diffusion method reduced the required computational load.

Though the four schemes have their own pros and cons, they share the same philosophy of being inspired from the human visual memory structure. Thus, combining them into an overall scheme which has all the benefits of each individual scheme is our future research direction. The first two schemes and chapter two and three are based on pixels and are directly related in that chapter two can be used for modeling the background model for the tri-model in chapter three. The scheme presented in chapter four can be understood as a feature-point-wise implementation of the same idea. The last scheme provides a more abstract and higher-level representation for the target object. We believe by appropriately using their results in parallel, we would be able to benefit from each scheme.

Bibliography

- Adam, A., Rivlin, E., & Shimshoni, I. (2006). Robust fragments-based tracking using the integral histogram. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, vol. 1, (pp. 798 – 805).
- Agarwal, S., Awan, A., & Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, 26, 1475–1490.
- Alvarez, G. A., & Cavanagh, P. (2004). The capacity of visual short-term memory is set both by visual information load and by number of objects. *Psychological Science*, 15(2), 106–111.
- Asadi, M., Dore, A., Beoldo, A., & Regazzoni, C. (2007). Tracking by using dynamic shape model learning in the presence of occlusion. In *Proceedings of Advanced Video and Signal Based Surveillance, IEEE Conference on*, (pp. 230 –235).
- Avidan, S. (2007). Ensemble tracking. *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, 29(2), 261 –271.
- Babenko, B., Yang, M.-H., & Belongie, S. (2011). Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, 33(8), 1619 –1632.

- Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, *13*(2), 111–122.
- Barnich, O., & Van Droogenbroeck, M. (2011). ViBe: A universal background subtraction algorithm for video sequences. *Image Processing, IEEE Transactions on*, *20*(6), 1709–1724.
- Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, *110*(3), 346–359.
- Besner, D., Davies, J., & Daniels, S. (1981). Reading for meaning: The effects of concurrent articulation. *The Quarterly Journal of Experimental Psychology*, *33*(4), 415–437.
- Brady, T. F., Konkle, T., Alvarez, G. A., & Oliva, A. (2008). Visual long-term memory has a massive storage capacity for object details. *Proceedings of the National Academy of Sciences*, *105*(38), 14325–14329.
- Brainerd, C. J., & Reyna, V. F. (2005). *The science of false memory*. Oxford University Press New York.
- Castel, A. D., Pratt, J., & Craik, F. I. (2003). The role of spatial working memory in inhibition of return: Evidence from divided attention tasks. *Perception & Psychophysics*, *65*(6), 970–981.
- Cohen, J. D., Perlstein, W. M., & Smith, E. E. (1997). Temporal dynamics of brain activation during a working memory task. *Nature*, *386*, 604.
- Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *25*(5), 564–577.

- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.
- Currie, C. B., McConkie, G. W., Carlson-Radvansky, L. A., & Irwin, D. E. (2000). The role of the saccade target object in the perception of a visually stable world. *Perception & Psychophysics*, 62(4), 673–683.
- Dick, A. (1974). Iconic memory and its relation to perceptual processing and other memory mechanisms. *Perception & Psychophysics*, 16(3), 575–596.
- Dinh, T. B., Vo, N., & Medioni, G. (2011). Context tracker: Exploring supporters and distracters in unconstrained environments. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*.
- Elgammal, A., Duraiswami, R., Harwood, D., Davis, L. S., Duraiswami, R., & Harwood, D. (2002). Background and foreground modeling using nonparametric kernel density for visual surveillance. In *Proceedings of the IEEE*, (pp. 1151–1163).
- Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9, 1871–1874.
- Felzenszwalb, P., Girshick, R., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9), 1627–1645.
- Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381–395.

- Georgiadis, G., Ayvaci, A., & Soatto, S. (2012). Actionable saliency detection: Independent motion detection without independent motion estimation. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, (pp. 646–653). IEEE.
- Godec, M., Roth, P., & Bischof, H. (2013). Hough-based tracking of non-rigid objects. *Computer Vision and Image Understanding*, 117(10), 1245 – 1256.
- Grabner, H., Grabner, M., & Bischof, H. (2006). Real-time tracking via on-line boosting. In *Proceedings of the British Machine Vision Conference*, vol. 1, (pp. 47–56).
- Grabner, H., Leistner, C., & Bischof, H. (2008). Semi-supervised on-line boosting for robust tracking. In *Proceedings of the European Conference on Computer Vision*, (pp. 234–247).
- Grabner, H., Matas, J., Van Gool, L., & Cattin, P. (2010). Tracking the invisible: Learning where the object might be. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, (pp. 1285 –1292).
- Henson, R. N. (1998). Short-term memory for serial order: The start-end model. *Cognitive psychology*, 36(2), 73–137.
- Hollingworth, A. (2005). The relationship between online visual representation of a scene and long-term scene memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31(3), 396.
- Hollingworth, A., & Henderson, J. M. (2002). Accurate visual memory for previously attended objects in natural scenes. *Journal of Experimental Psychology: Human Perception and Performance*, 28(1), 113.

- Hollingworth, A., et al. (2004). Constructing visual representations of natural scenes: The roles of short-and long-term visual memory. *Journal of Experimental Psychology-Human Perception and Performance*, 30(3), 519–537.
- Irwin, D. (1992). Visual memory within and across fixations. In K. Rayner (Ed.) *Eye Movements and Visual Cognition*, Springer Series in Neuropsychology, (pp. 146–165). Springer New York.
- Irwin, D. E. (1991). Information integration across saccadic eye movements. *Cognitive psychology*, 23(3), 420–456.
- Irwin, D. E., & Zelinsky, G. J. (2002). Eye movements and scene perception: Memory for things observed. *Perception & Psychophysics*, 64(6), 882–895.
- Isard, M., & Blake, A. (1998). Condensation–conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29, 5–28.
- KaewTrakulPong, P., & Bowden, R. (2003). A real time adaptive visual surveillance system for tracking low-resolution colour targets in dynamically changing scenes. *Image and Vision Computing*, 21(10), 913 – 929.
- Kalal, Z., Matas, J., & Mikolajczyk, K. (2010). P-n learning: Bootstrapping binary classifiers by structural constraints. In *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, (pp. 49 –56).
- Kim, I., Choi, H., Yi, K., Choi, J., & Kong, S. (2010). Intelligent visual surveillance - a survey. *International Journal of Control, Automation and Systems*, 8(5), 926–939.
- Kim, J., Wang, X., Wang, H., Zhu, C., & Kim, D. (2012). Fast moving object detection with non-stationary background. *Multimedia Tools and Applications*, (pp. 1–25).

- Kim, S. W., Yun, K., Yi, K. M., Kim, S. J., & Choi, J. Y. (2013). Detection of moving objects with a moving camera using non-panoramic background model. *Machine Vision and Applications*, *24*(5), 1015–1028.
- Klein, R. M. (2000). Inhibition of return. *Trends in cognitive sciences*, *4*(4), 138–147.
- Ko, T., Soatto, S., & Estrin, D. (2010). Warping background subtraction. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, (pp. 1331–1338).
- Krausz, B., & Bauckhage, C. (2011). Automatic detection of dangerous motion behavior in human crowds. In *Proceedings of Advanced Video and Signal Based Surveillance, IEEE International Conference on*, (pp. 224–229).
- Kwak, S., Lim, T., Nam, W., Han, B., & Han, J. H. (2011). Generalized background subtraction based on hybrid inference by belief propagation and bayesian filtering. In *Proceedings of Computer Vision, IEEE International Conference on*, (pp. 2174–2181).
- Kwon, J., & Lee, K. M. (2009). Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, (pp. 1208 –1215).
- Kwon, J., Lee, K. M., & Park, F. (2009). Visual tracking via geometric particle filtering on the affine group with optimal importance functions. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, (pp. 991 –998).
- Levy, B. A. (1971). Role of articulation in auditory and visual short-term memory. *Journal of Verbal Learning and Verbal Behavior*, *10*(2), 123–132.

- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91–110.
- Luck, S. J. (2007). Visual short term memory. *Scholarpedia*, 2(6), 3328.
- Luck, S. J., & Vogel, E. K. (1997). The capacity of visual working memory for features and conjunctions. *Nature*, 390(6657), 279–281.
- Magnussen, S., Greenlee, M. W., Thomas, J. P., et al. (1996). Parallel processing in visual short-term memory. *Journal of experimental psychology. Human perception and performance*, 22(1), 202.
- Mahadevan, V., & Vasconcelos, N. (2009). Saliency-based discriminant tracking. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, (pp. 1007–1013).
- Mei, X., & Ling, H. (2009). Robust visual tracking using l_1 minimization. In *Proceedings of Computer Vision, IEEE International Conference on*, (pp. 1436–1443).
- Mikolajczyk, K., Schmid, C., & Zisserman, A. (2004). Human detection based on a probabilistic assembly of robust part detectors. In *Proceedings of the European Conference on Computer Vision*, (pp. 69–82).
- Mittal, A., & Huttenlocher, D. (2000). Scene modeling for wide area surveillance and image synthesis. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, (pp. 160–167).
- Murray, D. J. (1967). The role of speech responses in short-term memory. *Canadian Journal of Psychology*, 21(3), 263.
- Palmer, S. E. (1999). *Vision science: Photons to phenomenology*, vol. 1. MIT press Cambridge, MA.

- Pashler, H. (1988). Familiarity and visual change detection. *Perception & Psychophysics*, 44(4), 369–378.
- Pérez, P., Hue, C., Vermaak, J., & Gangnet, M. (2002). Color-based probabilistic tracking. In *Proceedings of the European Conference on Computer Vision*, (pp. 661–675).
- Peterson, M. S., Kramer, A. F., Wang, R. F., Irwin, D. E., & McCarley, J. S. (2001). Visual search has memory. *Psychological Science*, 12(4), 287–292.
- Platt, J., et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3), 61–74.
- Posner, M. I., & Cohen, Y. (1984). Components of visual orienting. *Attention and performance X: Control of language processes*, 32, 531–556.
- Raffone, A., & Wolters, G. (2001). A cortical mechanism for binding in visual working memory. *Journal of Cognitive Neuroscience*, 13(6), 766–785.
- Rao, N. I., Di, H., & Xu, G. (2007). Panoramic background model under free moving camera. In *Proceedings of Fuzzy Systems and Knowledge Discovery, IEEE International Conference on*, (pp. 639–643).
- Ross, D. A., Lim, J., Lin, R.-S., & Yang, M.-H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77, 125–141.
- Rother, C., Kolmogorov, V., & Blake, A. (2004). "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transaction on Graphics*, 23, 309–314.
- Sheikh, Y., Javed, O., & Kanade, T. (2009). Background subtraction for freely

- moving cameras. In *Proceedings of Computer Vision, IEEE International Conference on*, (pp. 1219–1225).
- Sheikh, Y., & Shah, M. (2005). Bayesian modeling of dynamic scenes for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, *27*, 1778–1792.
- Snyder, J. J., & Kingstone, A. (2001). Inhibition of return at multiple locations in visual search: When you see it and when you don't. *The Quarterly Journal of Experimental Psychology: Section A*, *54*(4), 1221–1237.
- Sperling, G. (1960). The information available in brief visual presentations. *Psychological monographs: General and applied*, *74*(11), 1–29.
- Stalder, S., Grabner, H., & van Gool, L. (2009). Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *Computer Vision Workshops (ICCV Workshops), IEEE International Conference on*, (pp. 1409–1416).
- Standing, L. (1973). Learning 10000 pictures. *The Quarterly journal of experimental psychology*, *25*(2), 207–222.
- Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Proceedings of Computer Vision and Pattern Recognition, IEEE Conference on*, vol. 2, (pp. 246–252).
- Tomasi, C., & Kanade, T. (1991). Detection and tracking of point features. Tech. rep., Carnegie Mellon University.
- Vogel, E. K., & Machizawa, M. G. (2004). Neural activity predicts individual differences in visual working memory capacity. *Nature*, *428*(6984), 748–751.

- Vogel, E. K., Woodman, G. F., Luck, S. J., et al. (2001). Storage of features, conjunctions, and objects in visual working memory. *Journal of experimental psychology Human perception and performance*, 27(1), 92–114.
- Wilken, P., & Ma, W. J. (2004). A detection theory account of change detection. *Journal of Vision*, 4(12).
- Yi, K. M., Jeong, H., Kim, S. W., & Choi, J. Y. (2012). Visual tracking with dual modeling. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand, IVCNZ '12*, (pp. 25–30). New York, NY, USA: ACM.
- Yi, K. M., Kim, S. W., Jeong, H., Oh, S., & Choi, J. Y. (2010). Non-rigid object tracking with elastic structure of local patches and hierarchical sampling. In *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*, (pp. 1–8).
- Yi, K. M., Yun, K., Kim, S. W., Chang, H. J., Jeong, H., & Choi, J. Y. (2013). Detection of moving objects with non-stationary cameras in 5.8ms: Bringing motion detection to your mobile device. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, (pp. 27–34).
- Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys*, 38(4).
- Yin, S., Na, J. H., Choi, J. Y., & Oh, S. (2011). Hierarchical kalman-particle filter with adaptation to motion changes for object tracking. *Computer Vision and Image Understanding*, 115(6), 885 – 900.

국문 초록

본 논문에서는 실제 환경에서 발생하는 기존 움직이는 물체 검출 및 추적 방법들의 한계를 극복하기 위한 생체 모방(bio-mimetic) 모델들을 제안한다. 제안하는 모델들은 인간의 시각 인지(visual perception) 과정에 있어서 네 가지 서로 다른 형태의 시각 기억(visual memory)이 작용한다는 이론을 기반으로 한다. 인간의 네 가지 시각 기억은 가시 지속성(visible persistence), 정보 지속성(informational persistence), 시각 단기 기억(visual short-term memory, VSTM), 및 시각 장기 기억(visual long-term memory)으로 분류된다. 이를 바탕으로 본 논문에서는 움직이는 물체 검출 및 추적 문제를 관측된 장면(scene)을 임시적 단기 모델(temporary short-term model, TSTM) 및 보수적 장기 모델(conservative long-term model, CLTM)로 모델링하고 나타내는 문제로 이해한다. 즉, TSTM 및 CLTM을 효율적이고 효과적인 방법으로 구성하고, 둘을 함께 사용함으로써 가려짐(occlusion), 부정확한 초기화, 배경 간섭(background clutter), 드리프팅(drifting), 및 비-강체(non-rigid) 변화등의 문제가 일어나는 실제 환경에서 강인한 물체 검출 및 추적 성능을 얻을 수 있음을 보인다.

첫 번째로 본 논문에서는 비-고정(non-stationary) 카메라에서 움직이는 물체를 검출하기 위한 효율적인 TSTM 방법을 제안한다. 제안하는 방법은 개인 컴퓨터(personal computer, PC) 상에서 프레임 당 5.8밀리초(milliseconds, ms) 내에 동작하며 또한 모바일 기기에서도 실시간으로 동작한다. 실시간 성능 및 강인한 성능을 달성하기 위하여 제안하는 방법은 이중모드 커널 모델(dual-mode kernel model, DMKM)을 사용하여 배경(background)을 모델링 하며, 인접한 모델들을 혼합하여 카메라 움직임을 보정한다. DMKM을 통한 배경 모델링은 전경(foreground) 정보의 배경 모델 간섭을 방지하면서도 배경의 다양한 변화에 모델이 적응 가능하도록 한다. 제안하는 인접 모델 혼합 방법은 움직임 보정(motion compensation) 시 발생하는 오차를 감소시키며, 이 오차의 영향은 모델의 나이(age)를 저장함으로써 더욱 감소된다. 또한 연산량을 줄이기 위하여 제안하는 방법은 하나의 DMKM을 다중 픽

셀에 적용한다. 다양한 비교 실험을 통하여 제안하는 방법의 적은 연산량 및 강인한 성능을 확인하며, 제안하는 방법이 모바일 기기에서도 실시간으로 동작함을 보인다.

두 번째로 TSTM과 CLTM의 개념(concept)을 모두 활용하는 새로운 물체 추적 방법인 삼중 모델링(tri-model) 방법을 제안한다. 제안하는 방법은 가려짐, 배경 간섭, 및 드리프팅 문제를 새로운 삼중 모델링 방법을 통하여 동시에 해결하는 것을 목표로 한다. 제안하는 삼중 모델링 방법은 세개의 모델로 구성되며 각 모델은 추적 대상, 배경, 및 추적 대상이 아닌 움직이는 물체를 온라인(online)으로 학습한다. 제안하는 방법은 학습된 세 모델을 활용하여 현재 장면을 가장 잘 설명하는 구성을 찾는 방식으로 물체 추적을 수행한다. 이에 따라 추적 대상 모델 외에도 배경 및 전경 모델에 학습된 정보를 활용함으로써 가려짐 및 배경 간섭에 강인한 결과를 획득하는 것이 가능하다. 또한, 추적 대상은 이들 모델을 활용하여 보수적인 방식으로 갱신(update)되어 드리프팅이 방지된다. 나아가 제안하는 방법은 물체를 나타냄에 있어서 사각형 표시(bounding-box)에 국한되지 않고 픽셀 단위 결과를 추출하는 것이 가능하다.

세 번째로 본 논문에서는 픽셀 단위 모델링 방법을 넘어서 TSTM 및 CLTM을 모두 활용하는 로컬 특징점(local feature) 기반 물체 추적 방법을 제안한다. 제안하는 방법은 부정확한 초기화 및 극심한 가려짐에 강인한 방법으로 로컬 특징점의 “움직임 특출성(motion saliency)” 및 “모양 특출성(descriptor saliency)”을 새롭게 제안하여 활용하며, 일반화된 허프 변환(generalized Hough transform, GHT)으로 최종 추적 결과를 획득한다. 제안하는 움직임 특출성은 각 로컬 특징점의 순간적인 속도를 TSTM으로 구성하며 추적 대상 외의 물체들의 움직임들을 기반으로 두드러지는 움직임들을 강조하도록 동작한다. 모양 특출성의 경우 로컬 특징점들을 CLTM으로 모델링하며 특징점 모양을 기반으로 추적 대상의 특징점일 가능성을 학습한다. 제안하는 방법은 이들 특출성을 바탕으로 처음에 주어진 그대로를 찾는 것이 아니라 추적 대상이 어떠한 특징을 지니는지 지속적으로 학습하고 찾으며, 이에 따라 초기화에 덜 민감한 성능을 보인다. 또한 제안하는 방법은 추적 대상 외에도 주변의 모든 특징점들로부터의 추정 결과를 합하여 물체의 위치를 구하며, 이에 따라 가려짐에

도 강인한 성능을 보인다. 아홉 개의 영상에 대해 100개의 랜덤(random) 초기화를 사용한 비교 실험으로, 제안하는 방법이 다른 여덟 개의 비교된 알고리즘에 비하여 우수함을 보인다.

마지막으로 본 논문에서는 로컬 패치(local patch) 및 인접한 패치들 간의 구조를 활용하는 강인한 CLTM 방법을 제안한다. 제안하는 방법은 순차 베이지안 추론(sequential Bayesian inference)에 기반한 방법으로 부분적인 가려짐 및 물체의 비-강체 변화가 있는 환경 하에서 실시간으로 물체를 추적하는 것을 주안점으로 한다. 제안하는 방법은 크게 (1) 강인한 성능을 위한 로컬 패치들의 탄성적 구조로 이루어진 물체 모델링법 및 (2) 실시간 성능을 위한 효율적인 계층적 확산(diffusion) 방법의 두 부분으로 구성된다. 로컬 패치의 탄성적 구조는 인접한 패치들 간의 관계를 통하여 부분적인 가려짐 및 물체의 비-강체 변화 문제를 해결한다. 계층적 확산 방법은 연산 시간을 줄이기 위하여 사후분포(posterior distribution)가 집중된 부분에 샘플(sample)들을 생성한다. 또한, 본 논문에서는 제안된 방법의 유효성을 검증하기 위하여 가려짐 및 비-강체 변화가 존재하는 다수의 어려운 영상에서 실험을 수행하며, 제안된 방법을 통하여 다양한 상황 하에서 실시간으로 강인한 결과를 얻을 수 있음을 보인다.

주요어: 생체 모방 모델, 물체 추적, 움직임 검출, 임시적 단기 모델, 보수적 장기 모델

학번: 2007-23039