Visual tracking of articulated and flexible objects

Daniel Wesierski

▶ **To cite this version:**

Daniel Wesierski. Visual tracking of articulated and flexible objects. Other [cs.OH]. Institut National des Télécommunications, 2013. English. NNT : 2013TELE0007 . tel-00939073

HAL Id: tel-00939073
https://theses.hal.science/tel-00939073

Submitted on 30 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

DOCTORANT EN CO-ACCREDITATION

TELECOM SUDPARIS ET L'UNIVERSITE EVRY VAL D'ESSONNE

Spécialité: Informatique

École doctorale : Sciences et Ingénierie

Présentée par

Daniel Wesierski

Pour obtenir le grade de

DOCTEUR DE TELECOM SUDPARIS

Visual tracking of articulated and flexible objects

Soutenue le 25 mars 2013

devant le jury composé de:

Frédéric Jurie	Professeur à l'Université de Caen	Rapporteur
Rémi Ronfard	Chargé de Recherche de 1ère classe HDR à l'INRIA et au LJK	Rapporteur
Catherine Achard	Maître de Conférences HDR à l'UPMC	Examinatrice
Malik Mallem	Professeur à l'Université d'Évry-Val-d'Essonne	Examinateur
Bernadette Dorizzi	Professeur à Télécom SudParis	Directrice de thèse
Patrick Horain	Ingénieur d'Étude à Télécom SudParis, Docteur ingénieur	Encadrant

Thèse 2013TELE0007

Contents

Abstract vii

Acknowledgments ix

1 Introduction 1

1.1 Problem formulation . . . 2

1.2 Challenges . . . 3

1.3 Contributions . . . 7

1.3.1 Tracking objects of various shapes and motions . . . 7

1.3.2 Efficient convolution routine for Haar-like features . . . 8

1.4 Organization of the thesis . . . 9

1.5 Publications . . . 10

2 State of the art 11

2.1 Image Fundamentals . . . 11

2.1.1 Feature primitives . . . 12

2.1.2 Feature transforms . . . 13

2.2 Tracking in low dimensions . . . 16

2.2.1 Basic idea behind tracking . . . 16

2.2.2 Gradient-based tracking methods . . . 18

2.2.3 Fast motion . . . 20

2.2.4 Appearance change . . . 22

2.2.4.1 Stability vs plasticity dilemma . . . 23

2.2.4.2 On-line learning . . . 24

2.2.5 Summary . . . 25
2.3 Tracking in higher dimensions . . . 27
2.3.1 Models of objects . . . 27
2.3.1.1 Shape and appearance . . . 28
2.3.1.2 Complex Motion . . . 30
2.3.2 Model matching algorithms . . . 31
2.3.2.1 Local methods . . . 32
2.3.2.2 Semi-global methods . . . 33
2.3.2.3 Global methods . . . 34
2.3.3 Summary . . . 35

3 Tracking objects with global spatio-temporal constraints 37
3.1 Background . . . 39
3.2 Chain-graph models . . . 41
3.2.1 Appearance term . . . 42
3.2.2 Spatial term . . . 42
3.2.3 Temporal term . . . 44
3.3 Model-based object categories . . . 46
3.4 Tracking algorithm . . . 48
3.4.1 Initialization . . . 48
3.4.2 Matching . . . 48
3.4.3 Update . . . 50
3.5 Experiments . . . 53
3.5.1 Implementation details . . . 53
3.5.2 Qualitative Analysis . . . 54
3.5.3 Quantitative Analysis . . . 55
3.6 Conclusions and Discussion . . . 60

4 Model configurations for tracking various objects 63
4.1 Introduction . . . 64
4.2 Tracking straight objects . . . 65
4.3 Tracking planar objects . . . 74

4.4 Tracking heavily deformable objects 78
4.5 Conclusions 86

5 Efficient Convolution of Haar-like Features 87
5.1 Motivation 88
5.2 Parsing Haar-like features 90
5.2.1 Background 91
5.2.2 Problem formulation 93
5.2.2.1 Model 93
5.2.2.2 Example 96
5.2.3 Proposed parsing algorithm 97
5.2.3.1 Decomposing features into smaller kernels 99
5.2.3.2 Ensembles of trees 100
5.2.3.3 Choosing the best ensemble 102
5.2.3.4 Implementation: B-channel buffer 103
5.3 Experiments 104
5.4 Conclusions and Discussion 107

6 Conclusions 109
6.1 Contributions of the thesis 109
6.2 Potential of the tracking approach 110
6.3 Limitations of the approach 111
6.4 Future work 112

List of figures 115

List of tables 119

Bibliography 121

Abstract

Humans can visually track objects mostly effortlessly. However, it is hard for a computer to track a fast moving object under varying illumination and occlusions, in clutter, and with varying appearance in camera projective space due to its relaxed rigidity or change in viewpoint. Since a generic, precise, robust, and fast tracker could trigger many applications, object tracking has been a fundamental problem of practical importance since the beginnings of computer vision.

The first contribution of the thesis is a computationally efficient approach to tracking objects of various shapes and motions. It describes a unifying tracking system that can be configured to track the pose of a deformable object in a low or high-dimensional state-space. The object is decomposed into a chained assembly of segments of multiple parts that are arranged under a hierarchy of tailored spatio-temporal constraints. The robustness and generality of the approach is widely demonstrated on tracking various flexible and articulated objects.

Haar-like features are widely used in tracking. The second contribution of the thesis is a parser of ensembles of Haar-like features to compute them efficiently. The features are decomposed into simpler kernels, possibly shared by subsets of features, thus forming multi-pass convolutions. Discovering and aligning these kernels within and between passes allows forming recursive trees of kernels that require fewer memory operations than the classic computation, thereby producing the same result but more efficiently. The approach is validated experimentally on popular examples of Haar-like features.

Acknowledgments

Foremost, I would like to thank my supervisor, Patrick Horain, who has been guiding me from the very first day of the thesis. He has always had time for me and my thinking benefited much from discussions with him on computer vision. Our interaction have progressed my skills considerably.

My sincere thanks go to my thesis director, Berndatte Dorizzi, and to Jérôme Boudy coordinating my work along the ambitious roadmap of the EU-funded CompanionAble project. Their cheerfulness and readiness to help in all matters have been very impressing to me. Moreover, it was a big fun to interact with the CompanionAble "crew" during our project meetings across whole Europe. In particular, I would like to thank the Technical University of Ilmenau for inviting me for a short stay at their robotics lab - it was a great, interesting experience.

Nice atmosphere at the department have also embraced me thanks to Badr-Eddine Benkelfat and his kind leadership as well as to Patricia Fixot and her smile, openness, and patience of the *Super Souris* power, who always helped me with administrative work. My friends at the lab, Maher, David, Manoj, Daria, Sesh, Jérôme, Paulo, Toufik, and Mohammed, merit my special thanks, too. I have shared interesting work and great time and jokes with them.

At all times, my fiancée, Aneczka, has been beside me with love, support, and encouragement. Her unmatchable positive attitude to life has been brightening my path throughout this intensive period of the thesis.

Finally, I wish to thank my parents, for everything, and dedicate this thesis to them.

To my parents, Anna and Grzegorz

- Chapter 1 -

Introduction

Object tracking has been a challenging task of practical importance since the beginnings of computer vision. Precise, robust, and fast trackers could stir robotics, surveillance, gaming, augmented reality, and human-machine interaction (Fig. 1.1).

This dissertation focuses on robust and fast tracking of objects that can vary in shape and motion dramatically. In particular, the goal is to track rigid and deformable objects of various shapes in the spatio-temporal domain, which can shift, rotate, change scale, and deform by flexing, articulating, and vibrating.

Inspired by the human visual system, our ultimate goal is to build a *generic tracker* of objects of various shapes and motions. Then, the tracker could aid in solving computer vision tasks such as object and action recognition, so enabling computers to answer basic and important questions: *What objects are there*? *What are they doing*? and more generally: *What is happening in the scene*?

Figure 1.1: *Visual object tracking plays a central role in various, important applications, with examples (in clockwise order) including (i) a surgeon with access to smart visual feedback, who could perform complex remote surgeries on humans and animals like sewing and knot tying aided by suture tracking; (ii) objects pasted into a 3D scene through rotoscoping, an interactive tracking technique that requires fast and generic tracking of object mattes; (iii) recognizing and tracking plastic bags by an autonomous vehicle that is able to discern them from children in the street (due to* Sebastian Thrun@CVPR'12*); (iv) 3D printing that requires precise and fast 3D model acquisition, e.g. from a series of images; (v) smart homes with static and mobile devices that could monitor and assist elderly to help them live safely, independently, and in dignity (e.g., EU-funded* CompanionAble project*); (vi) a gesture control interface, one of the earliest potential applications of computer vision, to facilitate control over static and mobile devices.*

§ 1.1 Problem formulation

Object tracking can be defined as estimating object *trajectory* in some state-space over time. The state-space of the object can be a hidden space, which may not be measured *directly* by means of a single camera. Examples of state-spaces typically are object location, scale, orientation, or deformations, which constitute object pose.

The evolution of object trajectory can be viewed as a statistical process, which is generated by some underlying dynamic system defined in a state-space. Statistical approaches to the problem of visual object tracking first specify the random variables of the system, which are divided into these that the system can observe and those that the system cannot observe. Then, the system has to infer these hidden variables based on the observed ones.

This is known as an *inverse problem*. It was discovered by Viktor Ambartsumian in 1929, who studied the Sturm-Liouville problem for describing the motion of a vibrating string[1]. Its main property is that it cannot be solved unless there are some premises, or else assumptions, about it. Such prior knowledge reduces the variability of the problem by regularizing it with constraints. One would then be inclined to say that the behavior of the system is more regular and, therefore, more predictable and easier to infer.

§ 1.2 Challenges

Object tracking is an easy-to-state problem and is easy to solve for the human visual system. Based on image observations, typically a tracking system seeks to recover location and scale of the object, but sometimes also its orientation or deformation. However, it is amazingly hard for a computer to track an object that can:

- move fast and abruptly, leading to relatively large, hard-to-predict displacements and blur in the image (Fig. 1.3)
- be partially or fully occluded by other objects (Fig. 1.4)
- deform in constrained (Fig. 1.5), semi-constrained (Fig. 1.6), or unconstrained fashion (Fig. 1.2)
- change its illumination and its pose against camera viewpoint
- look similar to its moving surrounding, causing ambiguities in appearance

These phenomena influence acutely the appearance of the object. Likewise, ambiguities arising from the cluttered surroundings of the object effectively compromise the discriminability of the learned appearance. Therefore, the hardness of the problem grows enormously when these phenomena cumulate and happen simultaneously. For instance, will the tracker make good, robust decisions when its target object is next to an alike object and has changed its appearance suddenly while its neighbor has not?

[1] Wikipedia: http://en.wikipedia.org

Figure 1.2: *Sequence (from movie* American Beauty*) showing unconstrained deformation of a plastic bag. A plastic bag has no rigid parts and can take endless forms.*

Figure 1.3: *Sequence* [Santner et al., 2010] *showing fast motion. As the object moves fast between consecutive frames, blur artifacts become apparent.*

Figure 1.4: *Sequence* [Santner et al., 2010] *showing a bottle which is partially and fully occluded by other bottles. During partial occlusion, the appearance of the target bottle changes.*

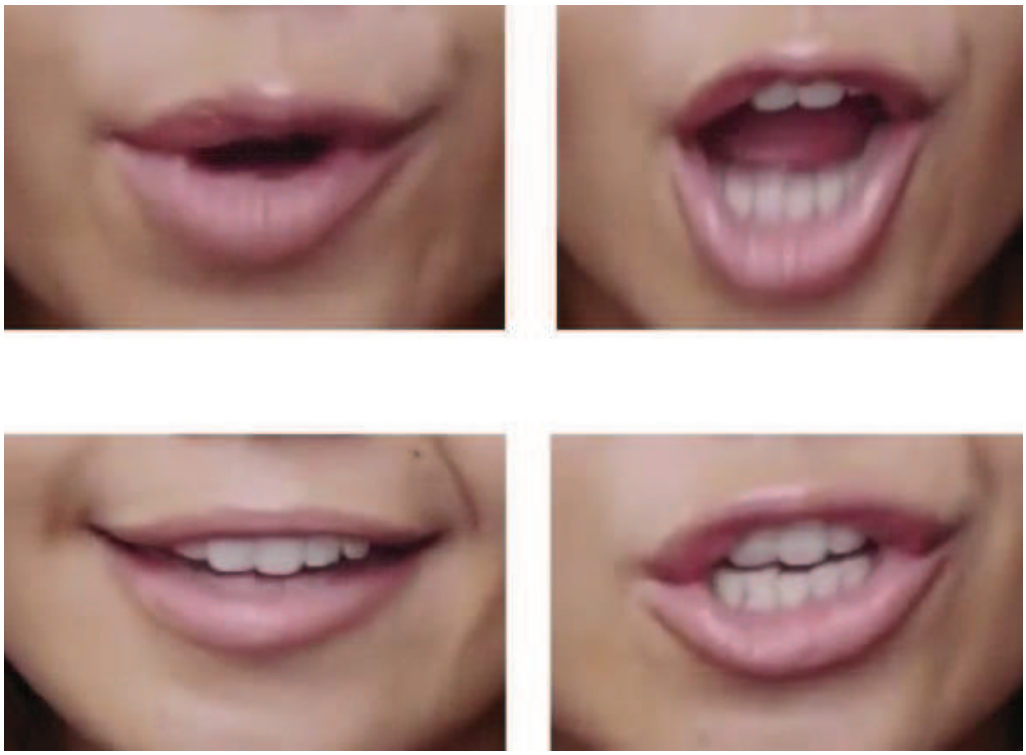


Figure 1.5: *Sequence (from* YouTube*) showing constrained deformation of human lips. The lips can take many forms of shape during a regular conversation.*

Figure 1.6: *Sequence* [Godec et al., 2011] *showing semi-constrained deformation. Human body deforms heavily but the deformations are constrained by its skeleton.*

The computational efficiency has to be addressed often in the design process as well. This is critical especially when another system accompanies a tracker, such as an object and action recognizer, in a time-bound application. Hence, in practice, a developer has to meet a reasonable trade-off between the robustness, precision, and processing speed.

Ideally, tracking systems should be engineered to solve "everything at once". Such a unified design requires a deep understanding of the synergy of the three axes of:

- image appearance,
- geometrical configuration,
- and their joint evolution over time.

This suggests that elaborating object representations should better encompass all these factors.

§ 1.3 Contributions

This thesis describes two contributions to the field of computer vision.

1.3.1 Tracking objects of various shapes and motions

The first contribution is a versatile representation for tracking a variety of rigid and deformable objects in various motions. Rigid and deformable objects are represented as chained segments of parts.

We wish to track objects by estimating their location, scale, in-plane orientation, articulations and confined deformations in the image, while being computationally efficient. Hence, we structure our model in the following way. We first split a segment into parts that are described simply with color histograms, so achieving invariance to orientation and scale change of the segment. Then, we link the parts into an attributed chain graph whose edges are labeled with distances between parts and their angular displacements w.r.t. the global orientation of the segment. Distances

control the stretching and shrinking of the segment, while the angular displacements control its bending. In this way, this graphical model captures the spatial coherency of the segment.

By linking segment parts in a chain graph, we enable efficient inference using dynamic programming. The model can represent objects of various geometrical shapes and track them stably and efficiently. In this way, further contributions upon the previous works are (i) to part-based object tracking, (ii) by exploiting temporal information effectively, (iii) thus demonstrating that our single model can be easily reconfigured from one object to another.

Finally, the proposed system can be implemented efficiently. The tracking algorithm scales linearly with the number of parts of the object and its video processing rate can exceed 100 `fps`. Tracking in the image plane does not require camera calibration. It also does not require background subtraction and can cope with a changing background.

1.3.2 Efficient convolution routine for Haar-like features

The second contribution is a formalism for parsing ensembles of Haar-like features (EHLFs), which are ubiquitous in computer vision. Individual HLFs are classically computed in one pass over integral image by reading the values at feature corners. Due to the overall simplicity of this approach, little work has been done on efficiently computing ensembles of Haar-like features.

Our primary observation is that EHLFs can be decomposed into simpler kernels, possibly shared by subsets of the features, thus forming multipass convolutions. Discovering and aligning these kernels within and between passes allows forming recursive trees of kernels that reuse the responses computed in the previous passes.

Next, in order to speed-up a data-bound computation such as convolution with HLFs, where arithmetic operations have a negligible cost, it is mandatory to reduce the data transfer between the processor and the memory. This requires a cache-friendly implementation for which we propose a circular N-channel buffer.

Combining these concepts together, the parsing procedure is designed to reduce the memory read-write count of the convolution with the trees of kernels. The trees

require less memory operations to produce the same result which leads to a more efficient convolution routine.

The approach allows predicting with high accuracy the potential theoretic speed-up over the classic computation, which is validated experimentally on two popular examples of ensembles of Haar-like features.

§ 1.4 Organization of the thesis

Chapter 2 is a review of the previous works on object tracking which are broadly divided into low- and high-dimensional tracking.

Chapter 3 introduces a deformable parts model by specifying its appearance, spatial, and temporal attributes. It is demonstrated how the model can represent objects with various shapes and motions by splitting them into a chained constellation of parts. In particular, objects are assigned to three categories using specific model configuration: (i) straight objects, where locations of their parts are approximated by a straight line, (ii) planar objects, where the locations admit a planar layout, and (iii) articulated objects, which are compositions of the objects from the first two categories. Next, the chapter describes an efficient tracking framework which employs the model. Finally, the tracker is evaluated on a popular benchmark for tracking rigid objects.

Then, chapter 4 presents several instances from the above three object categories together with their particular model configurations and applications.

In chapter 5 we describe the procedure for parsing Haar-like features into recursive convolution trees of simpler kernels. First, the steps of the parser, decomposition, permutation, and alignment, are explained. Their costs defined by the number of inputs and outputs are given. The described parsing procedure combines these steps to reduce the number of memory accesses. Finally, a technique is developed to efficiently implement the trees using a circular N-channel buffer. Performance gains are presented on two examples of ensembles of Haar-like features.

To conclude, the thesis is summarized in chapter 6, followed by a discussion on potential future work.

§ 1.5 Publications

Part of the work on object tracking from chapter 3 and chapter 4 was presented at ICIP'12 [Wesierski et al., 2012a]. The work on the Haar-like features parser was published in CVPR'12 [Wesierski et al., 2012b]. The work, which was published in EMMCVPR'11 [Jezierska et al., 2011], described an image denoising algorithm based on a two-move graph cuts framework and is out of scope of the thesis.

- Chapter 2 -

State of the art

Research on visual object tracking spans several last decades, so the volume of published results is large. In this chapter we synthesize influential concepts in object tracking. The gradual development of the domain and the motivations behind individual approaches are presented. Hence, the chapter begins by studying the basic cases and continues by enriching them with more complex challenges.

The first section reviews the image fundamentals together with representations of signals extracted from the images. Then, tracking methods are categorized by the challenges they address and are arranged chronologically. It is shown how the tracking schemes have evolved and progressed over the years.

Several important tracking domains are not described, such as multiple-object tracking, and the recently emerging fields of context-based tracking and tracking for re-identification, which are out of scope of the thesis.

§ 2.1 Image Fundamentals

An image of an object is more than a plain matrix of pixels. One seeks to recover its salient characteristics that can help recognize objects over time. These relevant characteristics may include color, texture, and optical flow, and refer to object features such as keypoints, contours, surfaces, and their motion. Hence, image patches can be represented by these features and tracked.

2.1.1 Feature primitives

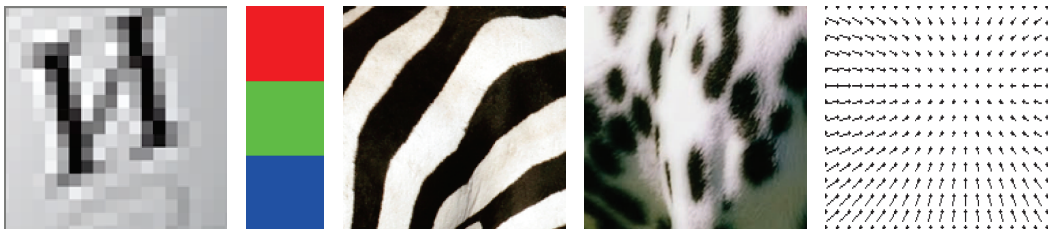
Image patches can be described with raw pixels, color and texture representations, and optical flow that results from object motion between consecutive frames, as depicted in Fig. 2.1.

Raw pixels: An image patch is a contiguous, usually rectangular region in a gray or color image, which contains raw pixels, representing light intensity. This representation can be used for global template tracking.

Color: Many color representations have been used for tracking, among which `RGB`, `HSV`, `Luv`, and `Lab` are the popular ones. Although one representation has its merits over the other, they all exhibit sensitivity to change of illumination, as reported in [Yilmaz et al., 2006].

Texture: Broadly, an image patch contains a regular or an irregular spatial pattern of pixels, which can be described as a texture. In this regard, blobs, corners, and edges, which are very characteristic to a human observer, are examples of textures exploited by tracking methods. Unlike color representations, texture gradients are little sensitive to illumination changes.

Optical flow: Optical flow is a two-dimensional motion field which captures the displacements of object edges and surfaces between two successive frames. Its computation is expensive, so it is estimated either sparsely [Lucas and Kanade, 1981] or densely on Graphics Processing Units (GPUs) [Werlberger et al., 2009].



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

Figure 2.1: *Image primitives.* (a) *Raw pixels.* (b) *Color: here RGB space.* (c) *Spatially regular texture.* (d) *Spatially irregular texture.* (e) *Optical flow.*

2.1.2 Feature transforms

Tracking methods often use feature transforms, which map raw pixels into other feature representations in order to build-in invariance to changes of e.g. scale, orientation, illumination, blur, or local deformations of the object. In this section, we briefly review several common feature transforms.

Color histograms: A simple and popular feature transform of an image patch is a color histogram, which represents the patch by a *bag-of-pixels* model. From a statistical point of view, it approximates the color probability density. Their limited estimation capabilities [Scott, 1992] notwithstanding, this thesis shows that color histogram strikes a good balance between efficacy and fast computation.

Apart from its remarkable simplicity, color histogram has a series of other advantages. It can be normalized to achieve invariance to scale changes, if needed. Next, it is invariant to permutations of pixels. This means the pixels can evolve freely within the patch during tracking, implicating not only invariance to rotation but also certain robustness to local deformations, which happen especially for non-rigid objects. Also, color histogram varies only slowly when the object/camera pose changes [Shapiro and Stockman, 2001]. On the other hand though, this means that it loses the position of pixels in the patch and so ignores the properties of patch's texture. Therefore, the patches shown in Fig. 2.2 share the same color histogram, but this representation then allows to track objects such as the one shown in Fig. 2.3.

Figure 2.2: *The left color patch of an object deforms gradually into the other patches. The middle patch could belong to the same moving object as it is only a slight deformation of the first patch. The right patch is an extreme deformation of the previous patches. It is unlikely to belong to the object as it has a very different texture pattern. Since all three patches yield the same color histogram, it is histogram's advantage to support slight deformations, while it is usually its disadvantage to support extreme deformations.*

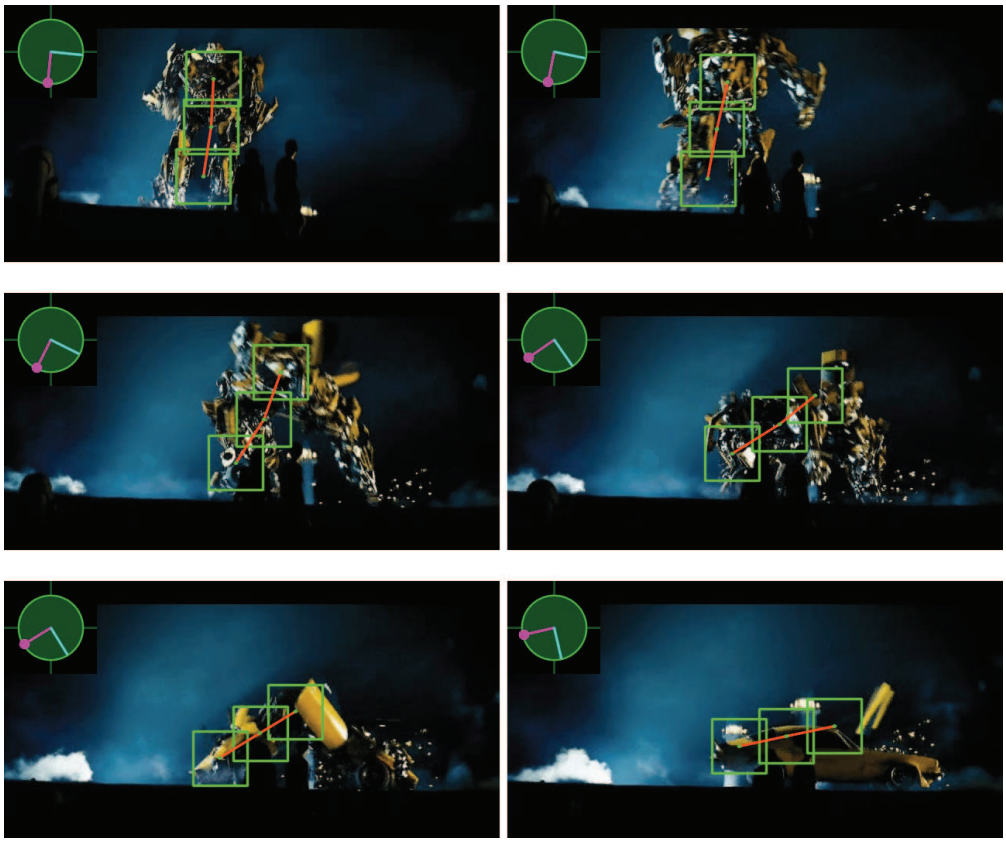


Figure 2.3: Transformers *sequence* [Kwon and Lee, 2009], *in which our tracker follows the location and orientation of the robot transforming into a car. The tracker uses color histograms, which can be thought of as bags-of-pixels. Since these representations lose the spatial layout of their components, the approach can track even such objects which undergo extreme shape deformations.*

A straightforward computation of color histograms is relatively inexpensive. However, even more efficient strategies [Porikli, 2005], [Berger et al., 2012] can be applied in specific scenarios, as exposed in [Berger et al., 2012].

Gradient features: Texture of object surface can be represented with Gabor [Daugmann, 1985] or steerable [Freeman and Adelson, 1991] filters. Since the spatial appearance of the surface is likely to be deformed during motion, it is common to track local features such as blobs and corners which are less sensitive to object motion. The features can be *described* with SIFT [Lowe, 1999] or its faster approximation SURF [Bay et al., 2006]. Both are scale invariant, although SIFT may suffer from blur artifacts [Gu et al., 2010]. Moreover, a rotation invariant generalization of SIFT is RIFT descriptor, proposed in [Lazebnik et al., 2005].

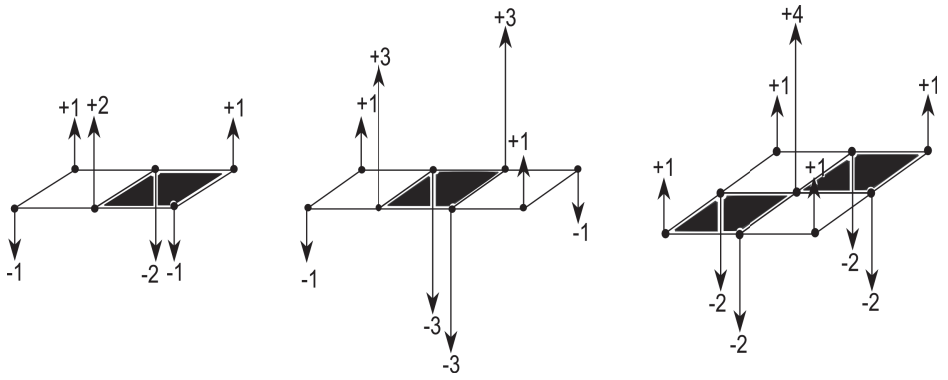


Figure 2.4: *Examples of double-differentiated Haar-like features, which approximate image edges and bars. As their kernels are sparse matrices, they can be convolved efficiently with integral image to produce the same result but faster than their non-differentiated counterparts convolved with the original image. Moreover, most kernels of Haar-like features are separable. Hence, this property is exploited in the thesis to create even more efficient, recursive convolutions with integral image.*

Edges are an important cue for tracking, as well. The Canny edge detector [Canny, 1986] finds binary edges under an optimal search criterion. More recently, Histograms of Oriented Gradients (HOG) [Dalal and Triggs, 2005] at edges have become popular for object recognition. HOG can be regarded as a feature transform leading to a laxer representation of the edges than the one obtained with the Canny edge detector. The transform is not orientation invariant, as opposed to color histogram or RIFT. However, it can capture effectively the object shape because it describes object contours and is tolerant to their local deformations.

Haar-like features: HLFs have become very popular in computer vision during the last decade. They are reminiscent of Haar wavelets and can be thought of as simple, coarse image templates. When combined, they can capture effectively sparse, local image structure, *e.g.* for face detection [Viola and Jones, 2001]. They can be smoothing filters and approximations of the first- and second-order image derivatives [Burt, 1981], [Heckbert, 1986], [Bay et al., 2006], denoting edges and bars, respectively. HLFs are used frequently in vision, *e.g.* for pedestrian detection [Viola et al., 2003] in spatio-temporal domain, for image descriptors [Dalal and Triggs, 2005], [Bay et al., 2006], for human limb tracking [Ramanan et al., 2007], for side face detection [Li et al., 2002] capturing spatial relations between patches, or for matching templates in Haar wavelet basis [Tang et al., 2007], [Ouyang et al., 2010].

Modern trackers make a great use of Haar-like features. Their double differentiated forms, with examples shown in Fig. 2.4, can be computed very rapidly [Viola and Jones, 2001] using integral image [Crow, 1984]. This thesis proposes an even more efficient recursive convolution scheme for these image representations.

§ 2.2 Tracking in low dimensions

This section describes approaches to tracking objects in low-dimensional state spaces, which estimate 2D location and occasionally the scale and orientation of the tracked object. We explain here their main concepts and overview the challenges they face. Potential applications span, among others, image stabilization, autonomous navigation, and surveillance.

2.2.1 Basic idea behind tracking

In a simple form, the problem of tracking is as follows: *given the initial appearance of an object within its bounding box, find this object in a video sequence.*

Further assuming that the background is stationary allows invoking background subtraction techniques to segment out the moving object. In real cases, this technique is *always* challenged by varying illumination, foreground occluding similar background, or moving nuance objects. Hence, adaptive background subtraction techniques have been proposed [Stauffer and Grimson, 1999], [Zivkovic, 2004], [Barnich and Van Droogenbroeck, 2011], though they still struggle with subtracting the background in the general case.

Alternatively, if the object is much different from a moving background (*e.g.* one of the balls well contrasted on a billiard table), one can resort to template matching techniques, which track the shifting object by detecting it in each frame. This *tracking-by-detection* scheme works in four steps by: (i) scanning a window template across the whole image, (ii) extracting the candidate template of the same size at each pixel location, (iii) using an objective function, like sum-of-squared difference (SSD) or correlation, to score the match between the target and candidate templates, and finally (iv) picking the new object's location yielding the best matching score.

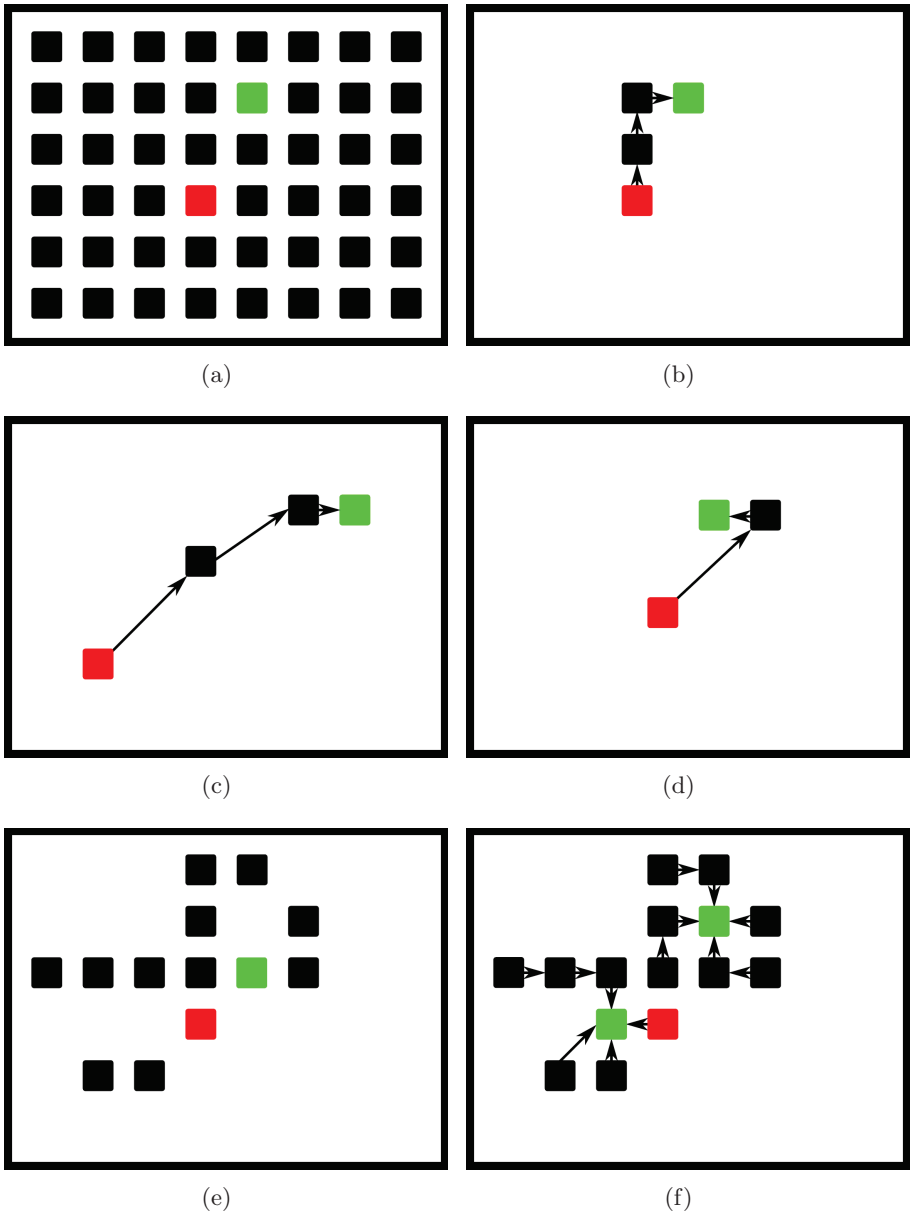


Figure 2.5: *Tracking approaches with low-dimensional state space, which is defined as object location in the image plane. The locations, estimated in the previous frame and in the current frame, are red and green squares, respectively. They denote shift motion of the object.* (a) *Template matching searches for the object over the entire image plane irrespectively of its previous location but is relatively expensive in computation.* (b) *If the object moves slowly, gradient-based methods initialize their search at the previous location and converge rapidly to current, true location only in few steps. The steps of convergence are marked with arrows and black squares.* (c) *A hierarchical search allows for tracking objects that move fast by running the gradient-based search procedure from lowest to highest level of image pyramid.* (d) *Alternatively, Kalman filter first predicts the current location. It uses a motion model to make a smart guess, which is then corrected by image observation.* (e) *Single prediction is often risky in visual tracking because of hard-to-define trustworthy motion models and background clutter. At increased computational cost, particle filter makes multiple predictions of locations, i.e. particles, which are sampled from the motion model.* (f) *Since the number of particles is limited, the approximate locations can be refined with gradient-based search, initialized at each particle, to converge to local modes of the probabilistic distribution of locations. Here, one of the two modes can be chosen which has higher likelihood. This procedure can increase the precision of tracking and reduce the number of particles.*

Template matching trackers feel quite comfortable in such scenarios. However, there is no free lunch. Since they scan the entire image to search for the object, as shown in Fig. 2.5(a), such a brute-force approach largely compromises the computational resources of the computer. In addition, it is hardly applicable to objects that undergo transformations more complex than shift [Jurie and Dhome, 2002].

If the inter-frame displacement of the object is small, other tracking schemes can be considered. By matching the template implicitly, iterative, gradient-based methods converge to the true object location within reduced number of steps.

2.2.2 Gradient-based tracking methods

The tracking methods, which optimize a continuous objective function by computing its gradient, are called gradient-based methods. They converge to the true object location in a few steps, as shown in Fig. 2.5(b). In order to design a gradient-based tracking algorithm, several issues have to be considered. Firstly, a representation of the appearance of the object has to be chosen, which will implicate a feature vector in a feature space. Secondly, a *differentiable* objective function has to be specified, which will measure the similarity between the known object feature and the feature extracted at a chosen location in the image plane. Finally, a gradient-based search technique has to be proposed, which will optimize the given objective function by searching iteratively for the best object location. In the following, we discuss two pervasive methods and their modifications, which address these design choices.

Lucas-Kanade trackers: Gradient-based tracking dates back to the seminal work of [Lucas and Kanade, 1981]. Given a template patch of the target object, one seeks to recover the parameters of the warping function of the image patch, which model the shift motion of the object. Starting from its location in the previous frame, the template is iteratively aligned with subsequent candidate patches. Hence, instead of matching the template to a patch at every location, the patches are extracted iteratively at subsequent locations that are determined by the updated warp parameters.

The parameters were found by the gradient-descent procedure minimizing the SSD function of the template and the warped image patch. The intensity landscape

of the warped patch is generally nonlinear so it was linearized with the first-order Taylor expansion. The convergence was guaranteed by computing the magnitude of change of the parameters between two consecutive iterations.

Originally proposed to estimate optical flow, the tracker has had many successors. We refer the reader to [Baker and Matthews, 2004] for an overview with rich technical details, but briefly, the warping parameters were extended to recover affine object motion [Bergen et al., 1992], [Tomasi and Shi, 1994]. Since the original method required Jacobian and Hessian matrix that are relatively costly to obtain, [Shum and Szeliski, 2000] updated the warping function instead of its parameters such that the Jacobian matrix could be precomputed to gain speed. Second-order approximation of the objective function [Benhimane and Malis, 2004] did not require computing the Hessian, thus increasing the numerical stability of the algorithm.

Mean-shift trackers: Another seminal gradient-based tracking method was proposed in [Comaniciu et al., 2000]. Unlike [Lucas and Kanade, 1981], who represented the object simply as a patch of pixels, here the object patch was transformed into a color histogram to better handle non-rigid objects. The histogram was weighted by an Epanechnikov kernel[1], laid over the patch, to robustify this representation. The kernel preserved spatial appearance to some extent and gave more importance to pixels closer to the object's center, as the peripheral pixels might belong to background.

The target template was matched with a candidate template under the Bhattacharyya distance function. The function was approximated by the first-order Taylor expansion around the sought after template and was optimized by shifting the kernel, and thus the object template, in the direction of the ascending gradient over weighted pixels. The algorithm required only several iterations to converge to the true location instead of evaluating a candidate template at each pixel, which speeded-up tracking. Additionally, the scale of the target was found by picking the best score of one of three kernels. They were obtained by rescaling the bandwidth of the current kernel with $\times 0.9$, $\times 1.0$, $\times 1.1$ and mean-shifted independently.

[1]Note that the single kernel did not estimate but weighted the density, in this case. This is in contrast to kernel density estimation methods, where the probabilistic density function is represented nonparametrically by a set of accumulated kernels anchored at data points.

By observing in [Hager et al., 2004] that Bhattacharyya and SSD measures are related, the SSD objective function was minimized by a gradient-descent algorithm. The authors hinted at the singularity issue of the kernel modulated histogram, which is insensitive to some particular motions of the object. To alleviate this problem, object representation was augmented with a stack of kernels. It was argued in [Collins, 2003] that the originally proposed approach to update the scale of the target had a tendency to shrink the kernel, lacking *de facto* the expansion power. Therefore, the Epanechnikov kernel was replaced by a Difference-of-Gaussians kernel to correct the scale update mechanism. In [Zivkovic and Kröse, 2004], the authors developed an EM-like algorithm to update scale as well as the orientation of the kernel by estimating the covariance matrix of the weighted pixel locations. Second-order Taylor expansion was proposed in [Xiao and Li, 2008], which improved the precision of the object localization at slightly increased computational cost. And recently in [Leichter, 2012], the histogram was smoothed with a uniform convolution kernel. This improved the inexact estimation of the density function resulting from histogram discretization and color drifts between nearby histogram bins, effectively leading to a more robust tracker.

2.2.3 Fast motion

Gradient-based tracking is a well justified approach given the object shifts rather slowly. Occasionally though, the object might shift between two consecutive frames rapidly. If the magnitude of the displacement of its location between the frames is greater than the local image area over which the gradient is computed, the object may escape from the tracker's potential reach. This is a well-known shortcoming of gradient-based trackers. Thus, we discuss here several categories of trackers that can cope with fast motion.

Hierarchical gradient-based tracking: Without departing far from the gradient-based tracking, the principle ideas behind the image pyramid framework stand attractive to alleviate the problem of fast motions [Anandan, 1989], [Bergen et al., 1992]. Namely, instead of starting the convergence solely from the previous location and scale of the original image, a hierarchical algorithm starts at the lowest

level of the pyramid of rescaled images. The found location at the lowest level of the pyramid initializes the convergence procedure at the next level. The procedure is repeated recursively up to the finest level to localize the object, as shown in Fig. 2.5(c). This hierarchical algorithm was integrated into Lucas-Kanade [Bouguet, 2000] and mean-shift [Li et al., 2010] trackers.

Prediction-based tracking: Alternatively, the rapidly changing object location can be first predicted and then corrected. A one-shot prediction with subsequent correction can be accomplished by a Kalman filter [McLauchlan et al., 1997], [Boykov and Huttenlocher, 2000]. It is a Bayesian filter, which maximizes the posterior distribution over the object state by recursively inferring on the first-order Hidden Markov Model (HMM). The filter can predict and correct for rapidly moving objects, as shown in Fig. 2.5(d), using a linear motion model of e.g. high speed and acceleration and a linear appearance model, respectively. If the object moves nonlinearly, its motion can be described with a nonlinear model, which is then linearized under Extended Kalman Filter [Rosales and Sclaroff, 1999] with the first-order Taylor expansion. However, verification of only a single hypothesis of object's location narrows the utility of the Kalman filter. In visual object tracking, objects may move not only fast and nonlinearly but also may look similar to the background, an effect known as clutter.

Particle filter extends the Kalman filter within the HMM setting. Instead of keeping only a single, most likely hypothesis, it updates multiple, more and less likely hypotheses, as shown for hypothesized locations in Fig. 2.5(e). Should the object be in clutter, the particle filter accounts for thus arising multimodal distribution simply by predicting multiple locations of objects represented by e.g. contours [Isard and Blake, 1998] or colors [Nummiaro et al., 2002], [Pérez et al., 2002]. This luxury of multiple hypotheses comes at the cost of increased computational cost, as the candidate appearances are evaluated at each hypothesized location. Provided that the number of particles is rather small (e.g. several hundred), this is not grave considering the computational power of modern computers. Nevertheless, while Kalman filters estimate the best state exactly under a given model, such a coarse partitioning of the underlying distribution function of the particle filter yields only

approximative state estimates.

Hybrid tracking: A way to increase the precision of a particle filter and to reduce the number of particles is to *refine* the approximate estimation of the filter with a gradient-ascent procedure, like mean-shift. The procedure climbs up the dense particle clouds seeking for the modes of the posterior distribution that correspond to more precise hypotheses, as shown for locations in Fig. 2.5(f). Namely, each particle is an initialization for the gradient-based procedure. The particle is pushed in the direction of the ascending gradient of the posterior and locked on one of the modes after convergence [Chang and Ansari, 2003], [Maggio and Cavallaro, 2005].

Next section reviews tracking systems in cases where objects not only move fast but also change their appearance during motion. Hence, the reviewed methods attempt to learn the new appearance while tracking the target object.

2.2.4 Appearance change

An appearance model of an object determines the representation of the object in the image. The object can be described by surface texture, contours, and color within a bounding box. However, the appearance model is challenged inevitably by occlusions, changes of illumination and of camera/object pose, object's spatial deformations, as well as blur caused by fast motions. Therefore, important aspects to be addressed are that of (i) what should be observed in the image and (ii) how this observation should be updated while preferably (iii) not compromising the speed of the tracker much.

When camera acquires an image of a moving object and its surrounding scene, the above artifacts are likely to be present simultaneously. Therefore, it is likely they will influence to a large extent the appearance of the object's template, specified by an external user in the first frame. This calls for an automatic update mechanism of the appearance model over time. Because of the above noise effects though, it is a complex decision problem.

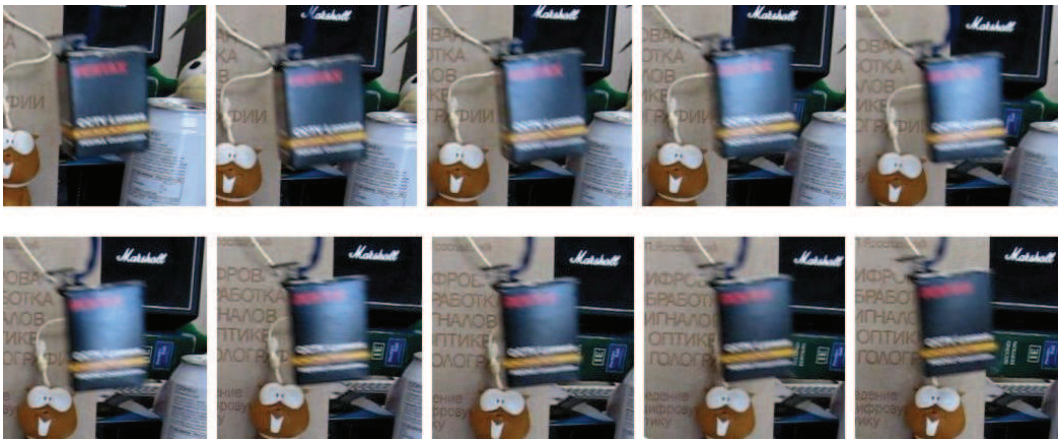


Figure 2.6: *A sequence of consecutive frames showing an external light source flashing over the surface of the black box. The light changes object appearance in a fraction of a second. Due to* [Santner et al., 2010].

2.2.4.1 Stability vs plasticity dilemma

A straightforward appearance update mechanism would replace the old template with the best one in each new frame. If an extra light source flashes over the object surface though (Fig. 2.6), the tracker learns this noisy observation instead of the true, new appearance of the object. Thus, such a blind, local updating degrades the appearance model by accumulating the erroneous observations. The tracker is likely to drift and may be unable to recover.

On the other hand, if the template is not updated at all, the tracker will not learn the incident noise. Provided that the object keeps its original appearance with only occasional changes, the tracker will be globally more stable, and it may lose track only sporadically. This marriage to the global template might lead to a happy-end, unless at one point the object appearance changes enough so that the template better matches the background clutter than the object.

To update or not to update leads to a well known stability-plasticity dilemma [Grossberg, 1987], making the design of a robust appearance model challenging. Researchers have developed numerous clever and complex decision systems that address this problem. The systems can decide *when* and *how* to learn the appearance of the object on-line in order to track objects stably despite noisy observations and flexibly in the presence of varying object appearance.

2.2.4.2 On-line learning

Humans can track objects which they have never seen before. Along these lines, semi-supervised learning has become the mainstream approach for on-line appearance learning. The problem is to design a feature selection mechanism which, initialized with a single bounding-box in the first frame, can bootstrap *good* features over time. The algorithm has to choose and classify these features of the object, which are most characteristic or which best discriminate it from the background. Trackers use boosting procedures of AdaBoost [Freund and Schapire, 1995] and Multiple Instance Learning (MIL) [Dietterich et al., 1997]. Recently, random forests [Breiman, 2001] have become popular as well, as they are less sensitive than boosting algorithms to noisy training examples, can be learned faster during on-line tracking, and can be easily parallelized on GPUs [Santner et al., 2010].

An early procedure for on-line learning was proposed in [Oza, 2001]. A tracker detected a new example of an object in each new frame with a set of predefined weak classifiers. It used the new example to retrain its classifiers with AdaBoost. Then, the learning task was transformed in [Grabner and Bischof, 2006] from updating the weights of classifiers to updating the weights of selectors, which chose the best positive features from their feature pools. This allowed to build reliably on-line appearance templates of the tracked object. Additionally, the pool was augmented with negative examples, chosen from the background of the object, to select these good features that discriminated the foreground from the background the most. As an alternative, in [Grabner et al., 2008] a pool of classifiers was chosen only in the first frame to increase the stability of tracking. AdaBoost was integrated into a particle filter tracker of object location and scale in [Klein and Cremers, 2011], where the bounding-box of the object was partitioned rigidly into four adjacent regions. Each region had its own pool of Haar-like features to better preserve spatial layout of object's appearance. Multiple Instance Learning [Babenko et al., 2009] selected the Haar-like features from the pool more flexibly by grouping them into positive and negative bags, instead of treating them as separate entities.

In object recognition, random forests are usually trained in off-line mode, with

training examples specified by an external user. In [Saffari et al., 2009], they were trained on-line yielding the on-line random forest (ORF) tracker that reported improvement in stability and resistance to occlusions over the on-line boosting of [Grabner and Bischof, 2006]. PROST [Santner et al., 2010] proposed to integrate simple template matching with the mean-shift clustered dense optical flow. The template tracker was responsible for stable tracking across the whole sequence, whereas the optic flow/mean-shift tracker reacted to local, most apparent shift. As both, the local and the global trackers had the same priority, the appearance of the main ORF tracker was updated when its output bounding-box overlapped with the one having higher score. A Tracking-Learning-Detection (TLD) approach [Kalal et al., 2012] estimated location and scale of objects over long video sequences. The tracker demonstrated robustness against object's fast motion and disappearance for longer periods. Locally, the TLD ran a Median-Flow tracker to estimate location and scale. It combined a pyramid Lukas-Kanade approach to estimate sparse optical flow, followed by a median filter to choose the most reliable flows. The tracker was then combined with a detector that was trained on-line using random forests. The forests were trained with positive and negative examples. The former were synthesized by applying affine warping to the current object appearance, and the latter were chosen around the object. In effect, the system could track rigid objects, while the performance of the tracker degraded for deformable objects whose templates warp in non-affine manner.

2.2.5 Summary

This section reviewed some popular trackers which output a low-dimensional object pose in the form of location, scale, and orientation. The gradient-based and prediction-based trackers seek to limit the search over the state space in order to gain on the processing speed. However, this might lead to problems when objects move fast and change their appearance.

The recent methods which learn the appearance on-line have advanced the field considerably. Building on the concepts of the previous approaches and using powerful machine learning techniques, they are more robust to fast motions and changing

appearance under real conditions. They often work in real-time, owing in part to the increasing computational power of modern computers and efficient designs of mechanisms which capture object appearance.

These contributions yield impressive results in tracking rigid objects. However, it remains open how these low-dimensional, on-line appearance learning approaches scale into higher dimensional state spaces. Such spaces can represent geometrical configurations of objects that e.g. articulate and flex. Possibly, these types of objects may require a part-based approach [Babenko et al., 2011].

Likewise, the rigid objects are observed only after being projected from the 3D scene onto a 2D image. Therefore, *are the rigid objects also rigid in the image*? Since they are represented by data drawn from some distribution underlying the image formation process, their rigidity is compromised by the projection and distortion effects arising from the object/camera interaction. It is therefore interesting to relax the assumption of inherent rigidity and treat this entity statistically.

In view of this, the model for object tracking developed in this thesis casts a rigid object as a collection of local parts and links them with soft geometrical constrains to exploit rigidity in a soft fashion. This deformable part model has the state space of increased dimensionality that results in the increased computational cost. Nevertheless, the complexity of the tracking algorithm scales linearly with the number of object parts and the tracker can yield real-time performance.

The appearance of each part is not updated which stands in contrast to the state-of-the-art trackers that update it on-line. It is motivated by a premise that if, say, a part changes its appearance, other parts might not. The ensemble of parts will likely attract this part to enforce spatial rigidity, imposed by spatial constraints. We show that this approach of representing the appearance locally with spatial coherence leads to a tracker with competitive qualitative and quantitative results.

Finally, such a soft, part-based model of rigid objects extends comfortably into representations for articulated and flexible objects. Since these representations are defined over high-dimensional state spaces, the state of the art in this domain is reviewed in the next section.

§ 2.3 Tracking in higher dimensions

Tracking an object in higher dimensional state spaces refers to the recovery of unobservable configuration of the pose of deformable objects. In Computer Vision, most notable examples are the recovery of pose of moving articulated objects like skeletons of mammals or human hands, and the recovery of flexing human facial muscles. Moreover, tracking of objects in higher dimensions usually assumes the state space of fixed dimensionality. Thus, the space dimension is not updated during tracking.

Apart from varying illumination, (self-)occlusions, and camera/object 3D pose change, deforming object pose affects object's appearance as well. The approaches, which track objects in low-dimensional state spaces, strive to be robust against deformations of a given object, and thus attempt to filter them out [Kwon and Lee, 2009], [Cehovin et al., 2011], [Godec et al., 2011]. In this regard, the trackers described in this section are principally different, as they seek to explain the appearance change of the object with its estimated deformation.

The approaches tracking objects in higher dimensional state spaces:

- *model* object shape, appearance, and motion;
- *learn* the parameters of their models;
- *reduce* the dimensionality of their models;
- *match* their models to track the object.

In the following, we discuss common modeling concepts. Then, we proceed with reviewing tracking methods, which differ in the manner of matching their particular object models to image observations.

2.3.1 Models of objects

Tracking articulated and flexible objects is governed by a model of object appearance and shape. Sometimes, motion is also modeled depending on a particular application, e.g. for motion animation and synthesis or for action recognition.

2.3.1.1 Shape and appearance

Shape of an object is represented locally by a meshed constellation of object "parts"[2]. One of the earliest and prominent models of object shape was the Active Contours (AC) [Kass et al., 1987], also known as Snakes. Having a physical interpretation, the model was formulated in terms of an energy function. It included the external force, which pulled the contour to the image observations, and the internal force, which aimed at preserving the smoothness of the contour. The features were gradient-based and locally constrained. Hence, since cluttered backgrounds could deform Snakes into shapes that mismatched the true object contours [Sclaroff and Isidoro, 2003], they were extended into models with region-based support [Ronfard, 1994], [Zhu and Yuille, 1996].

Active Shape Models (ASM) [Cootes and Taylor, 1992] could estimate object pose, scale, and flexibility. They were trained on a set of images containing annotated landmark locations on the object edges. The landmarks formed a point cloud in a high-dimensional state space of object shape, captured conveniently with a covariance matrix. These 'Smart Snakes' thus gained greater control than original Snakes over flexing shape of the object, though they could only be applied to a trained object category. The dimensionality of this space was reduced with Principal Component Analysis (PCA). The principal components denoted the canonical set of a flexing shape. In order to track the pose of flexible objects, the set was augmented by global, rigid transformation of the object and formalized as the Point Distibution Model (PDM) [Cootes et al., 1991], [Cootes et al., 1992].

Since the appearance model of ASM was matched only to local landmark locations, Active Appearance Model (AAM) [Cootes et al., 2001] represented object appearance with global textures. It improved upon the ASM as it converged faster and more robustly to the true solution [Romdhani, 2005]. The direct relation between the sampled global textures and corresponding flexing shape was learned in a regression manner. Building on AAMs, an object was represented with a planar, flexible mesh of parts in [Masson et al., 2005]. The mesh deformation parameters

[2]In addition to the term of a part, one can also use other, equivalent terms such as a landmark, a keypoint, a control point.

were obtained by PCA on a series of images taken at different viewpoints of the object. The appearance of the object was split into local patches. Each patch had an assigned tracker, regressed on the front-view image. A similar approach to AAM was proposed in [Sclaroff and Isidoro, 2003], where Active Blobs of a given elastic object were trained with a one-shot annotation. Namely, object's texture was acquired only from the initial frame of the video sequence. The boundary of the mesh was specified by segmentation and the potential flexing of the mesh was simulated with Finite Elements method.

Another well established, deformable part model representing an object as a constellation of parts, is the Pictorial Structures (PS) [Fischler and Elschlager, 1973], [Felzenszwalb and Huttenlocher, 2005]. Unlike PDM, which represents the object shape fully with a dense covariance matrix, the probabilistic graphical model of the PS is usually represented by a tree-graph structure. It yields a sparse covariance matrix but enables efficient matching. In particular, a star-graph was used for human pose estimation in still images [Ronfard et al., 2002], [Ramanan, 2006], [Eichner and Ferrari, 2009], [Andriluka et al., 2009], [Sapp et al., 2010] and in video sequences [Ramanan et al., 2007], [Andriluka et al., 2008], or for object detection [Felzenszwalb et al., 2010]. It has been extended to other tree graph topologies, for example to capture proxemics in photographs [Yang et al., 2012], to detect faces from different viewpoints [Zhu and Ramanan, 2012], and to capture full human pose [Yang and Ramanan, 2011]. Complete graph was used to detect general articulated structures in [Kumar et al., 2004b].

As the model proposed in the thesis uses the PS formulation, it is here briefly reviewed. A PS comprises an assembly of K parts arranged in a deformable geometrical configuration. It can be viewed as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, whose nodes $\mathcal{V}$ indicate parts, and edges $\mathcal{E}$ indicate connections between them. The edges thus define a particular shape configuration of a given object that models the dependence between the parts. A window containing a part (i.e. a bounding box) may be for instance a rectangle (e.g., for articulated models of human body parts [Felzenszwalb and Huttenlocher, 2005]) or a square (e.g., for iconic models of facial or body landmarks [Yang and Ramanan, 2011], [Zhu and Ramanan, 2012], [Yang et al., 2012]).

Typically, the state space of object parts $O = \{p_i\}_{i=1}^{K}$ is defined by location (x_i, y_i) of the window center in the image I, and also its local orientation θ_i. It can also be extended by scale s, which is a single global variable, shared by all windows. Given the image observation I, the posterior defines the probability map over the configuration of parts O as:

$$P(O|I) \propto \prod_{i=1}^{K} P(I|p_i) \prod_{(i,j)\in\mathcal{E}} P(p_i, p_j) \tag{2.1}$$

The appearance term $P(I|p_i)$ for part p_i models the likelihood of observing a chosen image feature (e.g., a color or gradient histogram, filter banks) over the state space of p_i. The prior term $P(p_i, p_j)$ corresponds to a pairwise kinematic potential between two given parts p_i and p_j and can be thought of as a spring between the parts capturing their allowable deformation in the image plane.

2.3.1.2 Complex Motion

Objects often move in complex, dynamic fashion, thereby their motion is generally difficult to model [Soatto and Yezzi, 2002] and estimate [Tomasi et al., 2003]. This is especially the case in high-dimensional state spaces that imply exponentially many degrees of freedom for object motion. For particular objects though, like humans, motion is a strong cue simplifying the problem of tracking in the presence of occlusions [Andriluka et al., 2008] and pose ambiguities [Urtasun et al., 2006] arising from the use of a single camera. Since the motion models are learned off-line from temporal data, they usually apply to specific motion styles [Urtasun et al., 2006], [Vondrak et al., 2008], [Brubaker et al., 2010] but can then be used in recognizing activities [Weinland et al., 2011].

Other approaches restrict their models to general motions simply by penalizing large displacements in location between two consecutive frames [Sidenbladh et al., 2000], [Ramanan et al., 2007], [Park and Ramanan, 2011]. In other words, fast motions are considered unlikely. In this thesis, a general motion model is also proposed to enable tracking of a handful of objects having various motion styles, like articulation and vibration. Therefore, the proposed approach is freed from an off-line

trained motion model that would be dedicated only to tracking a particular category of objects.

Specifically, the proposed motion model exploits the global orientation of each segment of the object from the previous frame of the sequence. As a result, the model penalizes fast rotation of the object, which is assumed to be rare, thereby favoring shift motion, which is assumed to be the dominant motion style. Hence, slow and fast shift motions are treated equally under the proposed model. The effect of this simple model on the fast rotary motion is analyzed in chapter 4.

2.3.2 Model matching algorithms

Tracking an object in higher dimensional state spaces is accomplished by matching[3] the model to each new image frame. However, matching a model is not straightforward because it is faced with the following challenge: *how to design an algorithm which can match the appearance model of the object to the image efficiently and optimally, such that the object shape preserves modeled spatial coherency of the meshed constellation of object parts over time.*

Theoretically, a straightforward and exact approach would be to enumerate all possible configurations in the state space. Even though this might be feasible when the dimension of the state space is small, it hardly scales to higher dimensions, in which the number of necessary enumerations grows exponentially. This curse-of-dimensionality effect renders such an approach infeasible in practice. Hence, many approximative matching techniques have been proposed to stay computationally attractive. They whether:

- limit their search space or
- sacrifice the accuracy of their models.

The algorithms search locally, semi-globally, and globally. They likewise relax the requirement for the full preservation of the spatial coherency of object parts, which compromises the optimality of the solution. Here, they are categorized by the *volume* of the state space they explore to find the best match.

[3] An equivalent term is model fitting. Similarly, if the model is statistical, such as in Bayes formulation, then one speaks of model inference.

2.3.2.1 Local methods

In Active Contours, a smooth template contour was matched to the nearby image contour by applying a gradient-based procedure. The geometrical constraints of the contours were hard coded, so in [Amini et al., 1990] the contour was represented by a chain graph which allowed for encoding soft constraints. Instead of the gradient-based search though, the best contour configuration in a given iteration was found with dynamic programming (DP), and the contour converged to the true configuration also in the iterative fashion. Even though the efficiency of DP decreases quadratically with the number of states (here the grid of pixel locations), the method was fast as the size of the local grid was small in each iteration.

Matching Active Shape Models allowed tracking flexing shape of e.g. a human face. Applied gradient-based procedure iteratively searched for parameters of the rigid transformation and of the best linear composition of the canonical shapes. Active Appearance Model was matched by searching over the shape and appearance parameters. Matching Constrained Local Model (CLM) [Cristinacce and Cootes, 2006] involved two-step iterative procedure, where a set of templates was matched under PDM as a regularization factor. In each iteration, the local templates of facial landmarks were matched independently within local neighborhoods of the current locations to generate response maps. The PDM was then fitted to these responses using simplex optimization to find the best match. In [Saragih et al., 2011], the authors argued that even though this simplex optimizer was more robust against measurement noise than classic gradient descent, it might be slow for high-dimensional state spaces. Therefore, they proposed to mean-shift [Fukunaga and Hostetler, 1975] the shape model to the maxima of the response map of each individual landmark, while regularizing each mean-shift step with the full shape PDM.

In [Fan et al., 2007], single kernel based mean-shift tracking [Comaniciu et al., 2000] was extended into the collaboration scheme of multiple kernels to track deformable objects, e.g. a magazine cover or articulating human arms. It mean-shifted kernel modulated color histograms of parts jointly to follow object deformations. The kernels were constrained chain-wise by distances to follow in-plane arm articu-

lation. Also, they were mean-shifted jointly to follow deformation of the magazine cover. In the latter case, the kernels were guided through the subspace over possible displacements, obtained by PCA on the batch of initial images, which corresponded to low-dimensional, affine transformation. The global affine motion was replaced by local affine motions in [Martinez and Binefa, 2008]. The ensemble of kernels was split into local triplets to better cope with object/camera pose changes. The method was applied to track rigid objects like cars and subregions of faces which undergo in- and out-of-plane rotation. The authors pointed out that generally small inter-frame displacements of the kernels were assumed, a well known shortcoming of local search methods.

2.3.2.2 Semi-global methods

Particle filtering (PF) is a prominent semi-global technique for model matching. Despite its approximative nature, it searches over a larger state space, which is its main advantage over the local search methods. Certainly, this comes at an increased computational cost[4]. In the theoretical case of infinite number of samples, it would be a global matching method, though such a scenario is computationally infeasible on existing computer architectures.

The trackers integrating the PF usually sample from the *motion* or *spatial* model. CONDENSATION tracker [Isard and Blake, 1998] extended the single hypothesis snake tracking. By sampling multiple contours from the motion model, the tracker achieved higher robustness against clutter. The dimensionality of the contours was reduced by representing their motion as affine, leading to a facilitated sampling. Individual object parts were pushed with independent PFs through the model of dynamics in [Mauthner et al., 2008], [Chang et al., 2009]. They were later assembled using spatial constraints to maintain spatial coherency of the object.

Instead of sampling particles for each part from the motion model, other schemes sample the parts jointly from the full spatial model of the object [Sudderth et al., 2003], [Isard, 2003]. Apart from their complexity, they are relatively slow though [Hua and Wu, 2006]. Since PF can be trapped in local maxima due to insufficient num-

[4]However, it might be overcome because each particle can be matched to the image independently of other particles lending itself to GPU implementation.

ber of particles in the high-dimensional state space, one can propose clever ways of resampling. Methods include scaling the covariance to smooth the density [Sminchisescu and Triggs, 2001], using annealing to gradually react to tight peaks in density [Deutscher et al., 2000], or using Rao-Blackwellization to reduce the dimensionality of the state space [Schindler and Dellaert, 2005]. Alternatively, by taking advantage of the problem domain to track 3D poses of humans, [Sminchisescu and Triggs, 2003], [Gomez Jáuregui et al., 2010] use inverse kinematics to better guide the particles during sampling, thus efficiently coping with 3D ambiguities arising from monocular vision.

2.3.2.3 Global methods

Global search methods typically apply the framework of Pictorial Structures yielding state-of-the-art results in object detection. Apart from works on object detection mentioned earlier, the PS has been employed to track objects by detecting them over the whole state space from frame-to-frame [Gu and Tomasi, 2011] or in a batch of frames [Kumar et al., 2004a], [Ramanan et al., 2007], [Andriluka et al., 2008].

Provided the graphical structure is a tree, a PS was matched to the image using dynamic programming in quadratic time in [Fischler and Elschlager, 1973]. Provided further that the costs on the spatial links in the tree graph had a particular form, [Felzenszwalb and Huttenlocher, 2000] propose to use Generalized Distance Transform [Felzenszwalb and Huttenlocher, 2004.] to reduce the complexity of model matching from quadratic to linear time. Unlike PF, it renders an optimal solution for the whole search space instead of only its approximation. Also, its inference complexity scales linearly with the number of parts, instead of exponentially at the cost of weakening the full shape model to only a tree. However, [Zhu and Ramanan, 2012] showed this might not be a grave shortcoming when applied to the detection, deformation and pose estimation of such complex objects as faces.

Variants of matching the PS to the image include the best, maximum *a posteriori* configuration L_{MAP}, or efficient sampling [Felzenszwalb and Huttenlocher, 2005] and maximal N-best configurations [Park and Ramanan, 2011] for higher level processing. It is tractable when the graph's structure $\mathcal{E}$ is a tree as standard dy-

namic programming routines can then be invoked. If the graph has a star structure, a more efficient, branch-and-bound matching technique can be applied [Kokkinos, 2011]. Alternatively, one can use a coarse-to-fine matching strategy [Ferrari et al., 2008], [Pedersoli et al., 2011].

2.3.3 Summary

This section highlighted model-based approaches to tracking objects by searching over high-dimensional state spaces. The dimensionality of the state space remains fixed throughout the sequence as it directly corresponds to semantic object configuration. By representing an image entity of a given object as a meshed constellation of parts, the reviewed models explain deformation of object contours, surfaces, and poses.

The described model matching algorithms can be categorized as local, semi-global, and global. Local methods can match the model relatively fast while fully preserving the configuration constraints. However, they can get trapped in a local optimum of the state space as their search space is small.

Sampling schemes are usually integrated into the PF framework. They explore the state space with multiple state hypotheses that can also be generated from the full configuration model. To render the matching feasible, they are forced to keep the number of particles low, implying a semi-global search. Thus, they are only approximative and scale poorly with the increasing dimensionality of the state space, thereby risking to get trapped in local optima as well.

In this thesis, the proposed model is integrated into the PS framework to yield a stable and efficient tracking of simple as well as more complex objects that are split into a chained assembly of parts. For instance, a short rigid bottle is described in a lower dimensional state space with several parts while a long vibrating guitar string or a long, deformable surgical suture are described in a state space of much higher dimensionality as they are composed of tens of parts. The PF scales exponentially with the increasing dimensionality and thus would hardly generalize to follow closely the latter, elongated objects. On the contrary, the proposed PS-based method scales linearly and can find globally optimal object configuration under a given model.

This comes generally at the cost of the approximative model of object structure, derived here as a chain graph, as object parts can admit planar layout, however, it is particularly well justified for tracking the elongated objects. The PS-based method is described in the next chapter.

- Chapter 3 -

Tracking objects with global spatio-temporal constraints

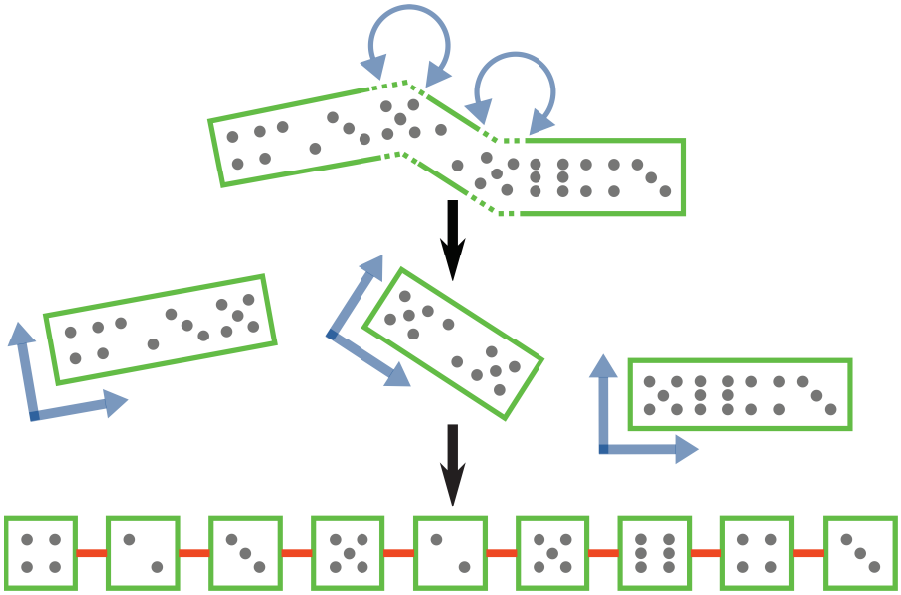


Figure 3.1: *Proposed model hierarchy for a deformable object (**top**) with hinged segments (**middle**), which are split into chained parts (**bottom**). The links between the parts are attributed with pairwise distances and angular displacements w.r.t. the coordinate systems to maintain spatial coherency of each segment over time. The segments can rotate freely around the hinge parts.*

Object tracking is an important open problem in computer vision, potentially with many rewarding applications. The state of the art in this domain has advanced greatly, though contributions continue to struggle with shape deformations, occlusions, changes in appearance, scale, and orientation, as well as illumination changes and background clutter.

This chapter describes a system for tracking objects in spatio-temporal domain, which have various shapes and motions. The objects can shift, rotate, change scale, and be rigid or deform by flexing, articulating, and vibrating. Therefore, we aim at a model-based object tracking system that:

- is modular, capturing spatial and dynamic properties of an object,
- is robust to appearance changes resulting from 3D pose change and occlusions,
- estimates 2D pose and scale of the object,
- yields a simple implementation with low computational cost.

We approach our objective by building on the Pictorial Structures (PS) [Fischler and Elschlager, 1973], [Felzenszwalb and Huttenlocher, 2005] framework. PS-based approaches usually model objects by tree graph structures to allow tractable inference. We cast the PS simply in the form of a chain graph, a particular tree graph, and thus represent an object as a chain of smaller parts. To model the local rigidity of the object, we structure it hierarchically with a layer of segments, which are composed of local parts, as shown in Fig. 3.1. Two segments share a part, which is anchored at their hinge, thus forming a heavily deformable object. We use pairwise kinematic potentials to keep consecutive parts in the chain at some preferred relative distance. In this way, we gain control over stretching and shrinking of each segment of parts. We then reference each segment with a global coordinate system whose orientation determines the inclination of the whole segment. The coordinate systems are used, in turn, to control the bend of each segment. Consequently, the model maintains the spatial coherence of the whole object over time.

Enforcing a global orientation constraint over a segment controls its rigidity without the need for higher order cliques in the graph, which is the key to fast inference. The angular displacements of pairs of parts in the segment, referenced with respect to segment's global orientation, effectively maintain the geometrical structure of the chain and thus its spatial coherence. The main assumption is that the orientation of the segment varies little between frames as we update it using the correspondences of the inferred locations of parts. However, borrowing terminology

from automatic control, we consider the orientation to be a slow-changing variable. Hence, when it is updated with one-frame lag, the effectiveness of the approach remains satisfactory.

To match the parts-based model to the image, our approach is first to create a discrete solution space in the current frame around the positions of parts from the previous frame. The model-based tracker then promotes these positions of parts at which the observed and modeled appearances of parts match well. It also favors these pairwise positions of parts, which follow the modeled spatial configuration by conforming to their preferred relative angular deviations and distances over time. As a result, the system can represent objects having a large variety of shapes and motions and yields stable and efficient tracking.

§ 3.1 Background

One family of tracking algorithms is initialized with a single bounding-box in the first frame. The objects can be rigid [Santner et al., 2010], [Kalal et al., 2012], deform less [Babenko et al., 2009], [Saffari et al., 2009], [Klein and Cremers, 2011], [Wnuk and Soatto, 2011], or much [Kwon and Lee, 2009], [Cehovin et al., 2011], [Godec et al., 2011] and the task is to update their appearance over time. Apart from illumination change and occlusions, these approaches also strive to be robust *against* deformations of a given object, and thus attempt to filter them out. In this regard, our approach is principally different and more general, as it not only applies to track rigid objects, but also applies to *track* the deformations of topologically constrained objects. Consequently, our model can be configured to track, for instance, trams or facial expressions.

We achieve stable tracking with a part-based model, which:

- splits the whole object into parts,
- links the parts with soft geometrical constraints,
- fixes the appearance and topological layout of the parts.

Part-based models with a fixed number of parts represent objects locally. Thus they overcome the problem of learning background pixels that puzzles single bounding-box trackers [Godec et al., 2011]. However, these models differ in the mechanisms for assembling the parts in the spatio-temporal domain [Cehovin et al., 2011]. In [Shahed et al., 2010], several parts of articulated objects were tracked jointly by greedily aggregating them for maximal coverage of the foreground binary mask, obtained by graph-cuts. [Fan et al., 2007] jointly mean-shifted kernel modulated histograms of parts to follow object deformations but required precomputing the subspace over their possible displacements on initial series of images to guide their joint convergence. Particle filtering (PF) was used in [Pérez et al., 2002], where the object was split into several parts modeled by color histograms. The tracker reported drastic improvement in stability owing to finer representation by parts but the parts were linked rigidly. On the other hand, large flexibility of parts was granted in [Mauthner et al., 2008]. However, the fundamental shortcoming of PF is that it scales exponentially with the dimensionality of the state space and thus with the number of parts. It is challenging to strike a balance between a reasonable amount of particles for algorithmic efficiency and satisfactory approximation of the underlying distribution of potential part locations for stability and precision.

A prominent framework for representing an object as a constellation of parts is the Pictorial Structures (PS) [Felzenszwalb and Huttenlocher, 2005] formalism, where structural relations between parts are treated statistically. Its merits and applications were already mentioned in Chapter 2, but briefly, object detection and tracking approaches generally apply PS to specific object categories [Ronfard et al., 2002], [Ramanan, 2006], [Eichner and Ferrari, 2009], [Andriluka et al., 2008] and to particular scenarios like gesturing and street scenes. Namely, in [Sapp et al., 2010] human upper body was approximated by a dense graph, decomposed into a mixture of trees to deal with heavy forearm foreshortenings. In [Andriluka et al., 2009] a tree-based model of whole human body was matched on a dense pixel grid to handle full and partial occlusions of pedestrians, resulting though in minutes-long inference per image.

We adopt the PS framework to track a large variety of objects by representing

them with a chain graph. At the first glance, this may seem restrictive. However, we show that our chain-based model can perform well when parameterized with global orientations over object segments allowing to maintain spatial structure of object appearance. Our tracking framework combines local shape and appearance descriptions through joint collaboration between parts in the spatio-temporal domain, requiring neither segmentation to assemble the parts during tracking, nor off-line learning on multiple examples of object structure and motion. The proposed method configures object structure only with one-shot annotation. It is able to deal with occlusions while estimating fast motions, in-plane orientation and scale changes of rigid, flexible, and articulated objects that can vary in the number of parts by several orders of magnitude. We also contribute by demonstrating that even though pictorial structures are usually considered slow [Jiang et al., 2012], we integrate them into a hierarchical model that can register object pose up to speeds far exceeding real-time. The following section introduces our model of an object composed of chained segments of parts.

§ 3.2 Chain-graph models

Depending on a particular object shape and its degree of local rigidity, we partition the object O into K segments $O = \{O_i\}_{i=1}^{K}$, such that each segment is not heavily deformable. Then, segment i is partitioned further into k_i parts $O_i = \{p_{i,j}\}_{j=1}^{k_i}$ specified as a rectangular or square window. Notably, this hierarchical and modular representation of the object needs not to be semantic. Formally, we represent the structure of the object as a chain graph $\mathcal{G}_{ch} = (\mathcal{V}, \mathcal{E})$, where nodes $\mathcal{V}$ are associated with the parts and edges $\mathcal{E}$ are associated with the links between the consecutive parts in the chain, as depicted in Fig. 3.1.

Each part is associated with an observed appearance feature $f_{i,j}$, hidden center location $l_{i,j}^t = [x_{i,j}^t \ , \ y_{i,j}^t]^{\mathrm{T}}$, and hidden scale $s_{i,j}^t$, forming random variable $p_{i,j}^t = [l_{i,j}^t \ , \ s_{i,j}^t]^{\mathrm{T}}$. The random variables are indexed with time $t = 1, \ldots$ and the initial frame is indexed with $t = 0$. The appearance model is kept constant to redetect objects after e.g. occlusions. The posterior over our PS of object O^t at the image

frame I^t then yields:

$$P(O^t|I^t, O^{t-1}) \propto \prod_{i=1}^{K}\prod_{j=1}^{k_i} \underbrace{P(I^t|p_{i,j}^t)}_{\text{Appearance term}} \prod_{j=1}^{k_i-1} \underbrace{P(p_{i,j}^t, p_{i,j+1}^t)}_{\text{Spatial term}} \underbrace{P(p_{i,j}^t, p_{i,j+1}^t|O_i^{t-1})}_{\text{Temporal term}} \quad (3.1)$$

where we set $p_{i,k_i}^t = p_{i+1,1}^t$, i.e. the last part of each segment is the first part of the next segment in the chain, so denoting a hinge. Thus, our graph has $|\mathcal{V}| = \sum_{i=1}^{K} k_i - (K-1)$ nodes.

3.2.1 Appearance term

The appearance of each variable $p_{i,j}$ is simply captured with a normalized color histogram $f_{i,j} = h_{i,j}$ to ensure invariance to scale and orientation changes of each part. The appearance term takes the following form:

$$P(I^t|p_{i,j}^t) = \exp(-\frac{1}{\nu_{i,j}}\chi^2(h_{i,j}, h_{i,j,c}^t)\ ) \quad (3.2)$$

where $\chi^2(h_{i,j}, h_{i,j,c}^t)$ is the chi-square distance between the model histogram $h_{i,j}$, precomputed in the initial frame I^0, and the histogram $h_{i,j,c}^t$ at a candidate location and scale for part $p_{i,j}$ in the current frame I^t. The distance is weighted with $\nu_{i,j}$ to allow for some variability in the color of the part. The set of appearance parameters for the parts of segment i is then $\Lambda_i = \{\nu_{i,j}, h_{i,j}\}_{j=1}^{k_i}$.

We note the tracker is not limited to orientation invariant features, such as color histograms. As we update the orientation of segments during tracking, orientation variant features, which use e.g. gradients, could be updated accordingly, as in [Maggio et al., 2005].

3.2.2 Spatial term

Neighboring parts in the i-th segment, $p_{i,j}^t$ and $p_{i,j+1}^t$, are constrained to lie within some predefined euclidean distance $d_{i,j}^t$ from each other, where:

$$d_{i,j}^t = \left\| l_{i,j}^t - l_{i,j+1}^t \right\|_2 \quad (3.3)$$

However, the varying scale of the object changes these relative distances, as we have:

$$P(p^t_{i,j}, p^t_{i,j+1}) = P(l^t_{i,j}, l^t_{i,j+1}|s^t_{i,j}, s^t_{i,j+1})P(s^t_{i,j}, s^t_{i,j+1}) \tag{3.4}$$

For simplicity, we model the joint scale prior $P(s^t_{i,j}, s^t_{i,j+1})$ for each pair of parts in the chain as a uniform distribution. Hence, we omit it and reduce the spatial term only to $P(l^t_{i,j}, l^t_{i,j+1}|s^t_{i,j}, s^t_{i,j+1})$ as:

$$\begin{aligned} P(p^t_{i,j}, p^t_{i,j+1}) &\propto P(l^t_{i,j}, l^t_{i,j+1}|s^t_{i,j}, s^t_{i,j+1}) \\ &\propto \mathcal{N}(\ d^t_{i,j}\ ; \rho^t_{i,j}\mu^{t-1}_{i,j;i,j+1}, (\rho^t_{i,j}\sigma^{t-1}_{i,j;i,j+1})^2\ ) \end{aligned} \tag{3.5}$$

The parameters $\mu^{t-1}_{i,j;i,j+1}$, $\sigma^{t-1}_{i,j;i,j+1}$ in (3.5), computed in the previous frame, denote mean distance between locations $l^t_{i,j}$ and $l^t_{i,j+1}$ of two neighbor parts and its standard deviation, respectively. They are rescaled with $\rho^t_{i,j}$ to denote their dependence on the local scales of the parts. We simply set the rescaling factor as an arithmetic mean of the scales of parts:

$$\rho^t_{i,j} = \frac{1}{2}\ (s^t_{i,j} + s^t_{i,j+1}) \tag{3.6}$$

The set of spatial parameters of the i-th segment with k_i parts is then $\Psi^t_i = \left\{\mu^{t-1}_{i,j;i,j+1}\ ,\ \sigma^{t-1}_{i,j;i,j+1}\right\}^{k_i-1}_{j=1}$.

We are interested in tracking rigid, flexible, and articulated objects. By taking advantage of the rigidity of segments, we would like to *globally* constrain each segment. A straightforward approach would augment the chain graph into a complete graph or, at least, a graph with higher order cliques, modeling jointly the distances between more than two parts in the segment. This would, however, hamper the computational efficiency of the tracker during inference on (3.1). As the spatial term controls only shrinking and stretching deformations of the neighboring parts in the chain, we propose to control a planar bend of each segment in the angular domain. We achieve this with our temporal term, described below.

3.2.3 Temporal term

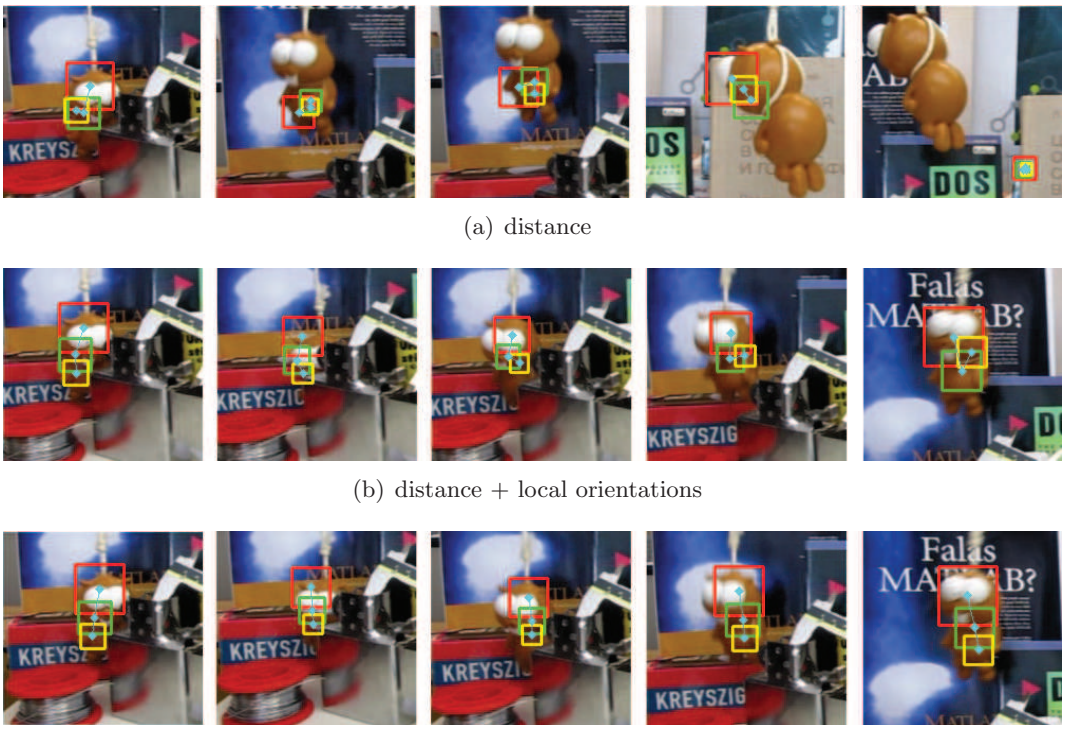
In a typical PS, part-based object structure can be parameterized by orientation in addition to location and scale. This extra parameter allows defining local coordinate systems for each part, which, in turn, allows for an angular kinematic constraint between a pair of parts. Since the parts in our model are parameterized by their center location and scale, the kinematic constraint appears only in the form of undirected distance. This is not sufficient to cope with appearance changes caused by occlusions or pose variations, as shown in Fig. 3.2(a). One might construct local coordinate systems between pairs of parts in the chain in order to constrain their angular motion locally. For instance, this could prevent the parts from flipping over between successive frames. However, forming local coordinate systems between pairs of parts is ineffective as this configuration does not restrict the parts *even of a rigid object* from rolling into a *spaghetti* over time, as depicted in Fig. 3.2(b).

We propose to constrain the parts of the segments of the object *globally* in order to take advantage of the local rigidity of each segment. The kinematic prior is extended to temporal domain to take advantage of temporal context. Namely, before tracking, each segment i is referenced with a 2D coordinate system having orientation Θ_i^0 w.r.t. the coordinate system of the image frame. This configuration of a segment allows determining local angular relations of the neighbor parts in the chain. These relations are defined by the angular displacements $\theta_{i,j;i,j+1}$, obtained (in the clockwise order) as `arccos` between the vector $[1, 0]$ defined in the coordinate system of the i-th segment and the normalized vector of $l_{i,j+1}^0 - l_{i,j}^0$. They are computed once and are not updated during tracking. Together with the spatial term, they capture *completely* the spatial layout of the parts and, thus, the rigidity of the whole segment. Hence, the angular bending of all the parts of the segment is tracked with the following temporal term:

$$P(p_{i,j}^t, p_{i,j+1}^t | O_i^{t-1}) = \mathcal{M}(\theta_{i,j;i,j+1}^t; \theta_{i,j;i,j+1} + \Theta_i^{t-1}, \kappa_i) \tag{3.7}$$

where $\mathcal{M}$ denotes the von Mises distribution and κ_i denotes the angular stiffness of the object.

During tracking, the stiffness penalizes angular deviations from $\theta_{i,j;i,j+1}$ (with offset orientation Θ_i^{t-1}) caused by deformation and rotation of the segment. Therefore, our model favors such arrangements of parts, which maintain predefined geometrical configuration of their segment assuming its orientation Θ_i^{t-1} does not change much between successive frames. This leads to an efficient and effective tracking procedure, as shown in Fig. 3.2(c), which does not suffer from the drawbacks of the two previous configurations. The set of global, angular parameters of the i-th segment is $\Omega_i^t = \left\{\Theta_i^{t-1}\ ,\ \kappa_i\right\}$.



(a) distance

(b) distance + local orientations

(c) distance + global orientation

Figure 3.2: *Tracking results for a PS model with three variants of spatial configuration. One segment represents a rigid toy object, whose three parts are linked by* (a) *distance only,* (b) *distance and local orientations of pairs of parts,* (c) *distance and single, global orientation over the whole segment. In* (a) *the rigidity of the object is often not maintained due to week local constraints, eventually leading to the lost track. In* (b) *the track is not lost, though the rigidity is also often not maintained leading to wrong estimates of orientation. Here the lowest part pivots counterclockwise around the the center of the middle part. In* (c) *the locations of the parts always approximate a straight line, thus maintaining object's rigidity correctly across the whole sequence. Sequences* (b) *and* (c) *represent the same frames for comparison between the local orientations and the single global orientation. The parameters were set to the same values for all three configurations of the model. The sequence comes from* PROST *dataset* [Santner et al., 2010].

The stiffness parameters κ_i can be adjusted in order to account for the anticipated change in angular speed of Θ_i^{t-1} between frames. For instance, tracking an acrobatic diver composed of several segments may require adjusting the parameters κ_i such that the model can explain his fast in-plane spins, whereas tracking a pedestrian, who walks slowly, suggests adjusting them such that the model considers only small rotary changes.

To summarize our model, the parts collaborate locally w.r.t. the distance constraints of Ψ_i^t, and globally w.r.t. the angular constraints of Ω_i^t for the i-th segment. By proper adjustment of the parameters K and $\theta_{i,j;i,j+1}$, the model can capture complex shapes of objects and control their elasticity and articulation in the image plane. This is further explained in the next section.

§ 3.3 Model-based object categories

This section explains how to configure our model (3.1) to represent a given object. The model is configured with K segments, each having k_i parts $p_{i,j}$. The parts of the i-th segment have predefined relative distances $\mu_{i,j;i,j+1}$ and angular displacements $\theta_{i,j;i,j+1}$ w.r.t. the orientation Θ_i of the 2D coordinate system assigned to the segment.

In this way, the mean values of the model, $\mu_{i,j;i,j+1}$ and $\theta_{i,j;i,j+1}$, enforce the rigidity of the object. This implies that, in the theoretical case of no noise, *any* graph $\mathcal{G}$ (e.g. any tree) would suffice to represent a given segment with the model. We use indeed a chain graph $\mathcal{G}_{ch}$ which is able to capture complex shapes of objects and control their elasticity and articulation. However, in practice we relax the assumption of object's rigidity with increased $\sigma_{i,j;i,j+1}$ and decreased κ_i. This enables the model to explain well the potential flexibility (i.e. confined deformation) and to be robust against 3D rotation of the segment.

The configuration parameters K and $\theta_{i,j;i,j+1}$ yield three categories of the tracked objects, which are shown in Fig. 3.3 and discussed below:

- **Rigid and flexible straight objects:** We define straight objects to be represented by a constellation of parts, such that their centers lie on a straight

line. Consequently, such an object is rigid or flexible. It has only one segment $K = 1$ with pairs of parts having no angular displacements from the oriented line $\theta_{1,j;1,j+1} = 0$.

- **Rigid and flexible planar objects:** The temporal term can be used to model the rigidity of planar objects, such that the centers of the assembly of their parts are located in the 2D image plane. In this case, a given object is defined by one planar segment $K = 1$ with possible angular displacements in the range $0 \leq \theta_{1,j;1,j+1} < 2\pi$.

- **Heavily deformable objects:** Any deformable object (e.g. articulated) that is amenable to a feasible split into a chained group of segments lends itself to a coherent model specified by our temporal term. All segments, where $K > 1$, are composed of parts having whether straight layout $\theta_{i,j;i,j+1} = 0$, or planar layout $0 \leq \theta_{i,j;i,j+1} < 2\pi$.

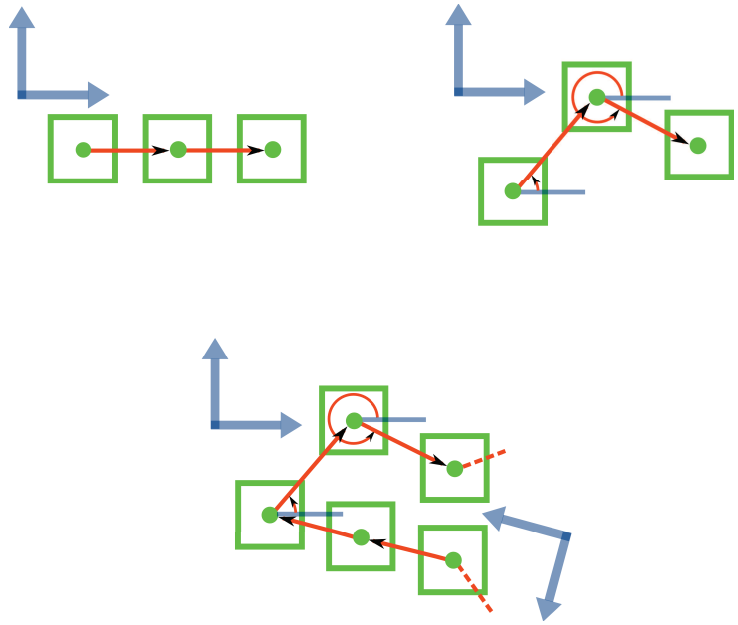


Figure 3.3: *Three object categories (in clockwise order): straight object, planar object, and deformable object. Their parts are assembled with a chain graph which determines pairwise connections attributed with distance and angular constraints. The global orientation of the coordinate systems is the reference for pairwise angular displacements of the parts.*

§ 3.4 Tracking algorithm

This section describes the proposed model-based tracking method, which is summarized in Alg. 1. We begin by first describing a single-shot configuration procedure of the model. Once initialized, the tracking consists in matching the model to each new frame to find the best configuration of object parts. When the best configuration is found in the current frame, the parameters of the model are updated, which is explained last. The below presentation is accompanied by a tracking example, depicted in Fig. 3.4.

3.4.1 Initialization

The object tracking method admits the following manual initialization. The number of segments K and their orientations Θ_i^0 are defined in the first frame of a video sequence I^0. Then, the location and size of each part $p_{i,j}$ in a given segment is specified. Lastly, a chain graph $\mathcal{G}_{ch} = (\mathcal{V}, \mathcal{E})$ is proposed, which determines the manner in which the parts are linked.

Based on the locations and connections of the parts, the model features $h_{i,j,m}$ as well as pairwise distances $\mu^0_{i,j;i,j+1}$ and angular offsets $\theta_{i,j;i,j+1}$ from Θ_i^0 are computed automatically. The remaining parameters of the model are set manually.

3.4.2 Matching

We follow a tracking-by-detection approach to track an object. The model (3.1) is matched to each new frame I^t based on the model parameters and object state estimated in the previous frame, as shown in Fig. 3.4(a).

1. Construction of the graph: We first build a Markov Random Field (MRF) in the form of a chain graph $\mathcal{G}_{ch}$ with $|\mathcal{V}|$ nodes, which represent the parts $p^t_{i,j}$. Each node of $\mathcal{V}$ in the graph has $C_{i,j}$ candidate states, which denote hypothesized locations and sizes of a part. As shown in Fig. 3.4(b), the states are generated simply by forming a regular grid. The sparsity of the grid is $R_{i,j}$, which determines the spacing between the states in the image plane. The grid is anchored at the best state of the corresponding part from the previous frame. The size of the grid, and

thus the size of the state space, is proportional to the size of the part by the factor $Q_{i,j}$. In effect, the grid approximates a uniform motion prior over $p^t_{i,j}$ to facilitate tracking of unexpected and fast motions and enables recovery from the lost track caused by e.g. occlusions.

2. Evaluation of the appearance term: For each hypothesized state, we compute a candidate appearance of the part, determined by $h^t_{i,j,c}$, and score it with (3.2). The score indicates how well the candidate appearance matches the modeled appearance at that hypothesized location and size of the part.

3. Evaluation of the spatial term: The spatial score between the hypothesized locations $l^t_{i,j}$ and $l^t_{i,j+1}$ of a pair of parts is obtained by (3.5). The score relates to the amount of shrinking or stretching of the pair of parts. As noted earlier, if some parts change their scale between successive frames $t-1$ and t, the mean and variance parameters in Ψ^t_i change accordingly. In order to account for it, the parameters are rescaled with the $\rho^t_{i,j}$ factor (3.6), which is obtained from the hypothesized scales of the pair of parts. The example in Fig. 3.4(c) shows the parts with the same locations but different scales, resulting in different scores. We refer the reader to the caption for further details.

4. Evaluation of the temporal term: The angular deviations of pairs of parts are computed w.r.t. the orientation Θ^{t-1}_i, which is estimated for their segment i in the previous frame. The temporal scores are then obtained by (3.7). Specifically, in Fig. 3.4(d) a connection between hypothesized locations of a pair of parts p_1 and p_2 has a high score as its angular deviation α from Θ^{t-1}_1 is small. On the other hand, Fig. 3.4(e) depicts a situation where a connection between the parts p_1 and p_2 has a very low score as the angular deviation β is very large. In this way, the model gains control over the amount of bend of each segment.

5. Inference on the graph: After computing all the numeric tables of the appearance, spatial, and temporal scores, we infer on the negative log-posterior $\texttt{-log}(P(O^t|I^t, O^{t-1}))$ of (3.1) using dynamic programming routine to obtain the Maximum-A-Posteriori (MAP) configuration of the object O^t_{MAP}.

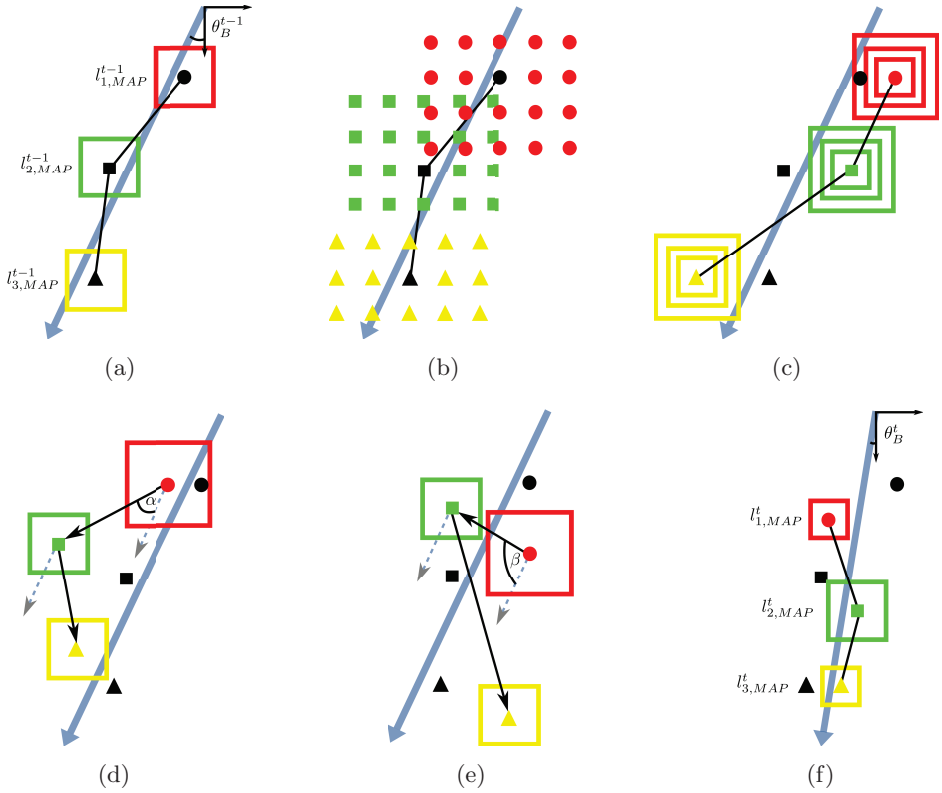


Figure 3.4: *Tracking procedure of a rigid object with one segment* ($K = 1$) *on a synthetic example.* (a) *Estimated states* O_{MAP}^{t-1} *and orientation* Θ_1^{t-1} *of* $|V| = k_1 = 3$ *parts in frame* $t-1$. (b) *Uniform sampling around parts' locations from previous frame realized by regular grids. The points on each grid represent hypothesized locations and sizes of each part in the current frame* t. (c) *For each hypothesized location of each part, the number of scale states* $|s|$ *of each part (here:* $|s| = 3$*) results in* $|s|^{|V|} = 3^3$ *part configurations. The local distances and orientations of consecutive pairs of hypothesized locations are the same for these* 3^3 *part configurations. However, the* 27 *configurations have different scores as the distances are rescaled, according to particular hypothesized scale of each part.* (d) *Part configuration with high score (small* α*).* (e) *Part configuration with low score (large* β*).* (f) O_{MAP}^t *of the parts inferred from the MRF in frame* t. *New orientation* Θ_1^t *of the object is computed from the correspondences between previous and current locations of the parts using Kabsch algorithm.*

3.4.3 Update

We update the parameters of our model based on the inferred MAP configuration of the parts in the current image frame I^t.

1. Location: The tracker is shifted to the MAP locations $L_{i,MAP}^t$ of the parts of each segment i, where $L_{i,MAP}^t = \left\{ l_{i,j,MAP}^t \right\}_{j=1}^{k_i}$. When the central location of the segment $L_{i,m}^t$ is required, we compute it as the average over the parts:

$$L_{i,m}^t = \frac{1}{k_i} \sum_{j=1}^{k_i} l_{i,j,MAP}^t \tag{3.8}$$

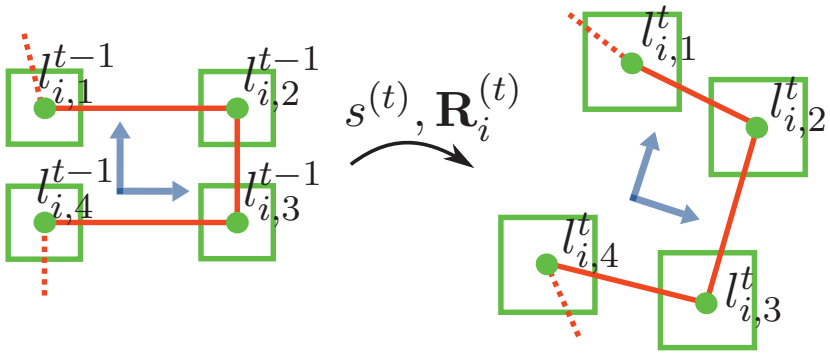


Figure 3.5: *Synthetic example of segment i of articulated object, whose scale s^t increases. The segment, consisting of $k_i = 4$ parts, deforms and rotates clockwise by $\mathbf{R}_i^t$ between two successive frames. The locations of the parts, translated back to the origin of the 2D coordinate system, and their correspondences between both frames allow to recover the rotation $\mathbf{R}_i^t$ of the segment in the least-squares sense despite its incident deformation. The dotted links connect it to neighbor segments, which rotate independently of the i-th segment.*

2. Scale: The global scale s_O^t of the whole object O is computed as the average over the scales of all windows of parts as:

$$s_O^t = \frac{1}{|V|} \sum_{j=1}^{|V|} s_{i,j,MAP}^t \tag{3.9}$$

and passed through the `IIR` filter as

$$s_O^t = (1 - r)\ s^{t-1} + r\ s_O^t, \tag{3.10}$$

with the forgetting factor r. Alternatively, the scales could be updated individually for each particular part or segment, depending on application. The parameters of (3.5) are updated with the filtered scale as:

$$\mu_{i,j;i,j+1}^t = s_O^t\ \mu_{i,j;i,j+1}^{t-1} \tag{3.11}$$

and

$$\sigma_{i,j;i,j+1}^t = s_O^t\ \sigma_{i,j;i,j+1}^{t-1}. \tag{3.12}$$

3. Orientation: The coordinate system (CS) of each segment $i \in 1, \ldots, K$ with updated in-plane orientation Θ_i^{t-1} is the reference for the spatial layout of parts $p_{i,j}^t$

in the next frame I^{t+1}. The fixed number of parts allows establishing correspondences between the MAP locations $L^{t-1}_{i,MAP}$ and $L^{t}_{i,MAP}$ of parts of segment i in the preceding and current frame, respectively, as shown in the example in Fig. 3.5. Hence, we propose to compute the orientation Θ^t_i using Kabsch algorithm [Kabsch, 1978], which estimates the rotation $\mathbf{R}^t_i$ between the point correspondences[1] in the least-squares sense by solving:

$$\underset{\mathbf{R}^t_i}{\operatorname{argmin}} \sum_{j=1}^{k_i} \left\| \hat{l}^t_{i,j} - s^t \mathbf{R}^t_i \hat{l}^{t-1}_{i,j} \right\|_2^2 \tag{3.13}$$

Note that $\hat{l}^t_{i,j}$ and $\hat{l}^{t-1}_{i,j}$ are translated to the origins of the respective CSs and $\hat{l}^{t-1}_{i,j}$ are rescaled. Effectively, the algorithm tracks in-plane orientations and is robust to considerable out-of-plane orientations (e.g. turning face) and large deformations (e.g. mouth opening wide). The stiffness parameters κ_i remain constant, as they are assumed invariant to any object deformations and changes in viewpoint.

Algorithm 1 Tracking procedure.

Initialization:

1. Define K segments of the target object O
2. For each segment i, define its orientation Θ^0_i and k_i parts
3. Define chain structure $\mathcal{G}_{ch} = (\mathcal{V}, \mathcal{E})$ of O to link the parts
4. Find $O^0_i = \left\{ p^0_{i,j} \right\}_{j=1}^{k_i}$ and compute $\left\{ \Lambda_i, \Psi^0_i, \Omega^0_i \right\}$ based on O^0_i and $\mathcal{G}_{ch}$
5. Define state space with $\{R_{i,j}, Q_{i,j}, |s|\}$

Iterations:

For frames I^t, $t = 1 \ldots$ do
- For segments $i = 1 \ldots K$ do
  - For parts $j = 1 \ldots k_i$ do
    - 1.1. Generate hypotheses $\Gamma^t_{i,j} = \left\{ p^t_{i,j,c} \right\}_{c=1}^{C_{i,j}}$ with $\{R_{i,j}, Q_{i,j}, |s|\}$ around $p^{t-1}_{i,j,MAP}$ (§ 3.4.2)
    - 1.2. Evaluate Appearance term with $\left\{ \Lambda_{i,j}, \Gamma^t_{i,j} \right\}$ (§ 3.2.1)
- For segments $i = 1 \ldots K$ do
  - For parts $j = 1 \ldots k_i - 1$ do
    - 2.1. Evaluate Spatial term with $\left\{ \Psi^{t-1}_{i,j}, \Gamma^t_{i,j}, \Gamma^t_{i,j+1} \right\}$ (§ 3.2.2)
    - 2.2. Evaluate Temporal term with $\left\{ \Omega^{t-1}_{i,j}, \Gamma^t_{i,j}, \Gamma^t_{i,j+1} \right\}$ (§ 3.2.3)
- 3. Infer best configuration $O^t_{MAP} = \left\{ O^t_{i,MAP} \right\}_{i=1}^{K}$ (§ 3.4.2)
- 4. Update $\left\{ s^{t-1}, \Psi^{t-1}_i, \Omega^{t-1}_i \right\}$ to $\left\{ s^t, \Psi^t_i, \Omega^t_i \right\}$ (§ 3.4.3)

[1] For ease of readability, we drop the index $t-1$ in the notation of scale change and rotation from frame $t-1$ to t.

§ 3.5 Experiments

This section evaluates qualitatively and quantitatively our tracker on the popular PROST[2] dataset. We compare the tracker with existing state-of-the-art methods that track *rigid* objects. The purpose of such a comparison is to demonstrate that our spatio-temporal model is rich enough to deliver comparable performance with the best of these algorithms (on this benchmark), while having a remarkably simple appearance model (3.2). The appearance is fixed, as in [Adam et al., 2006], and thus not learned on-the-fly, as opposed to [Saffari et al., 2009], [Babenko et al., 2009], [Gu et al., 2010], [Santner et al., 2010], [Klein and Cremers, 2011], [Wnuk and Soatto, 2011], [Kalal et al., 2012] that learn their appearance models on-line. As the occlusion event is not modeled explicitly, we enforce constant appearance so that the tracker is robust against occlusions and thus can recover from them by re-detecting the object.

3.5.1 Implementation details

All experiments use the following *fixed* settings. In our model (3.1), the appearance term (3.2) of each part $p_{i,j}$ uses remarkably simple image features in the form of 512-dimensional `RGB` color histograms $h_{i,j}$ (8 bins for each color channel), evaluated with chi-square distance and weighted with $\nu_{i,j} = 2.0$. The mean distance of the spatial term (3.5) between pairs of parts, $p_{i,j}$ and $p_{i,j+1}$, is computed automatically from the initial locations of the parts as $\mu^0_{i,j;i,j+1} = \left\| l^0_{i,j} - l^0_{i,j+1} \right\|_2$, while the standard deviation $\sigma^0_{i,j;i,j+1}$ as an average of these window radii of the parts which are close to the global orientation. The angular stiffness parameters κ_i for each segment i of the temporal term (3.7) are chosen such that they correspond to the angular standard deviation of 60°.

Then, the tracking procedure, described in section 3.4, renders the MAP configuration of the parts from (3.1), which is defined on the collection of 3D regular grids assigned to the parts. The resolutions of the grids $R_{i,j}$ are sparser than the image grid and their sizes are set to twice ($Q_{i,j} = 2$) the sizes of their respective parts. The

[2] http://gpu4vision.icg.tugraz.at/index.php?content=subsites/prost/prost.php

scale is partitioned as $\times 0.9$, $\times 1.0$, $\times 1.1$ and `IIR`-filtered with the coefficient $r = 0.1$.

The tracking algorithm was implemented in `C++` and the experiments were run on a plain PC equipped with Intel Xeon @ 2.4 GHz, 4 MB last level cache, and 3.5 GB RAM, with single-threaded implementation. The frame processing speed scales linearly with the number of the parts, though also depends on their window sizes.

3.5.2 Qualitative Analysis

This section demonstrates the qualitative results of our method in Fig. 3.6–3.9 on PROST dataset. Several challenging cases of the dataset, which are encountered during tracking, are exposed for each video sequence in Table 3.1. We configure our model with K=1 segment for each object and specify the initial orientation of its coordinate system. The segment is then partitioned *evenly* into $k_1 = 3$ parts, i.e. such that the parts span the object with no (or very small) overlap. This initialization is shown in the first frame of the figures below.

Using a remarkably simple appearance model in the form of color histograms, our method tracks objects reliably. Despite the fixed appearance, the method can handle blur, heavy scale changes, and 3D motion in the **Board** and **Lemming** sequences, as shown in Fig. 3.6 and 3.8.

Contrary to the gradient-based and prediction-based methods, it detects objects simply by scanning evenly a larger portion of the image. Hence, it can cope with fast motions, which are present e.g. in the **Box** sequence. However, the algorithm lost the track in this sequence because of high illumination change. The color histograms are not robust enough to cope with such severe changes of appearance, though the tracker later re-detected the box. We also tried `HSV` color space, as in [Pérez et al., 2002], but increased robustness against such illumination changes was not observed.

On the other hand, an important advantage of fixing the appearance is increased robustness against multiple occlusions. This is demonstrated very well in the **Liquor** sequence in Fig.3.9, where the target bottle is occluded by other bottles heavily and multiple times.

In all sequences, the tracker estimated the in-plane orientation and scale of the object. The below figures show that the tracker generally makes reliable updates

of the orientation and scale while the processing speed also remains at a high rate. Seldom, the orientation is not updated correctly but the tracker is able to recover the right orientation. This can be seen especially on the **Board** sequence, where the board changes its 3D pose drastically by making an out-of-plane rotation. Even though this motion is not built into our model and yet changes the appearance of the board much, the tracker can recover from these artifacts as shown in the final stage of the sequence.

Sequence	Frames	Main Challenges
Board	698	3D motion
Box	1161	fast 3D motion, occlusions, illumination
Lemming	1336	heavy scale changes, motion blur
Liquor	1741	motion blur, occlusions

Table 3.1: *The challenges of the experimental sequences of the* PROST *benchmark, which deteriorate the appearance of the four objects during tracking. The majority of the listed challenges is due to* [Santner et al., 2010].

3.5.3 Quantitative Analysis

We also evaluate quantitatively our approach on the PROST dataset. We use the following evaluation measures:

- Intersection-over-union from PASCAL VOC challenge, as in [Santner et al., 2010],
- Mean distance precision, as in [Babenko et al., 2009].

Specifically, the first criterion renders a detection as true positive when its bounding box overlaps with the ground truth bounding box by $> 50\%$. The recall performance is reported as the number of true positives over the sum of true positives and false negatives. The second criterion computes the ℓ_2-distance between the centers of the detected and the ground truth bounding boxes.

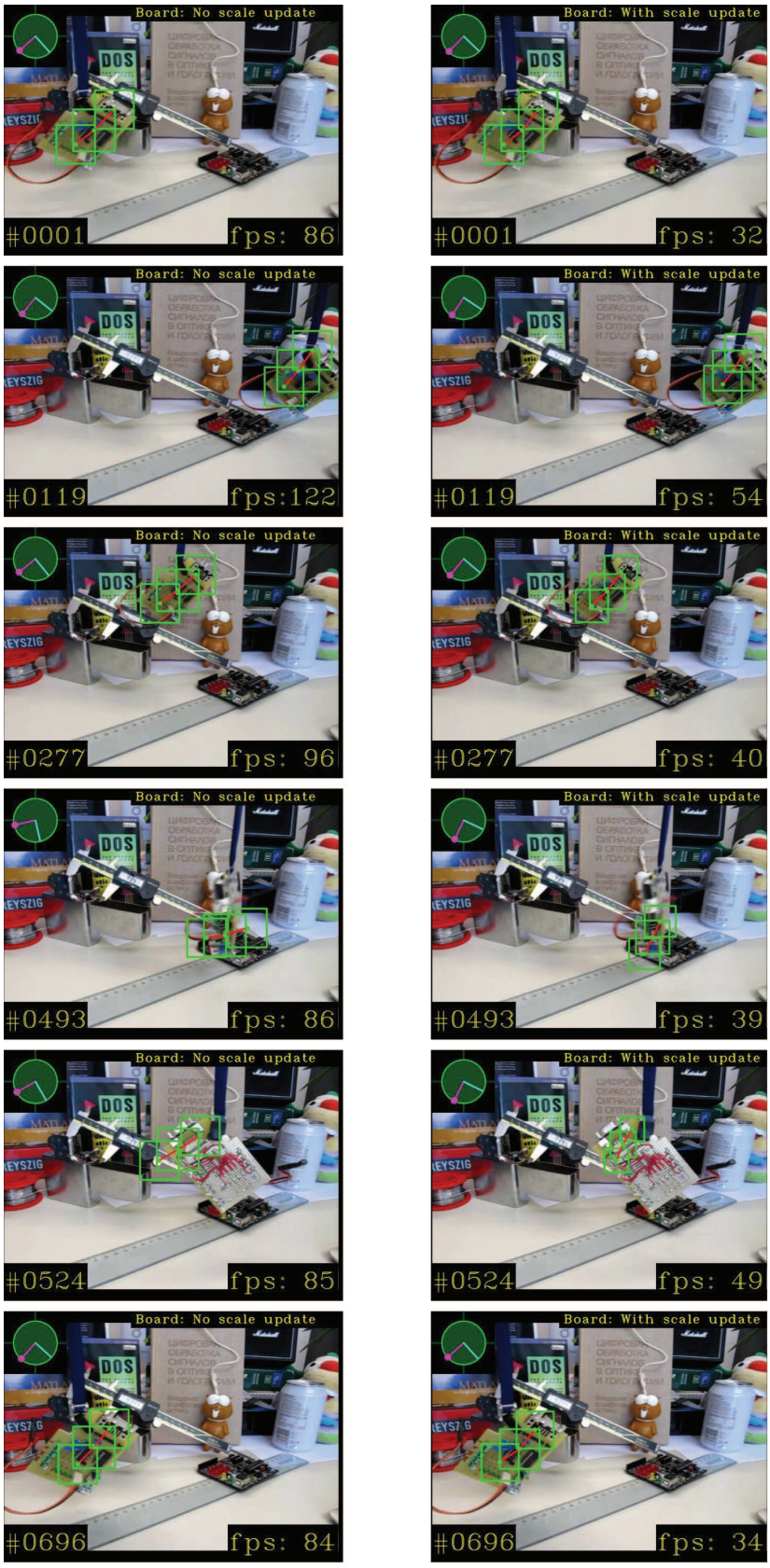


Figure 3.6: *Several frames of* **Board** *sequence from* PROST *database, which show tracking results of our method without (**left**) and with (**right**) scale update. The tracker remains stable despite abrupt appearance change in frames* #493 *and* #524 *caused by* 3D *pose change.*

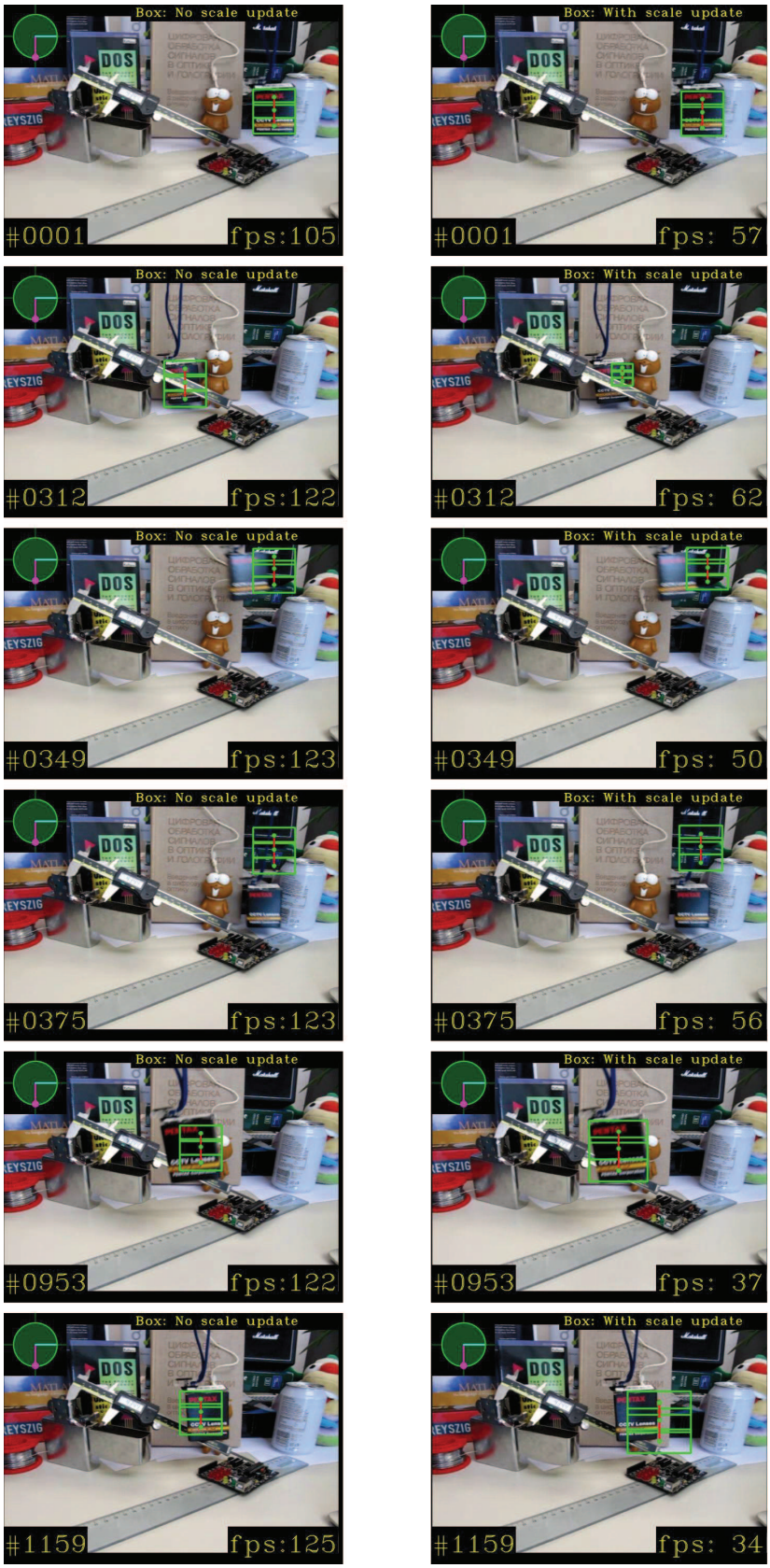


Figure 3.7: *Several frames of* **Box** *sequence from* PROST *database, which show tracking results of our method without (**left**) and with (**right**) scale update. The tracker behaved well throughout, handling partial occlusion in frame* #312*. But, it lost track of the object commencing from frame* #349 *in both cases because of the rapid illumination change of the box surface. It re-detected the box later in the sequence.*

Figure 3.8: *Several frames of* **Lemming** *sequence from* PROST *database, which show tracking results of our method without (**left**) and with (**right**) scale update. The tracker behaved stably despite partial occlusions in frames* #346 *and* #450*, and* 3*D pose change in frame* #1037*.*

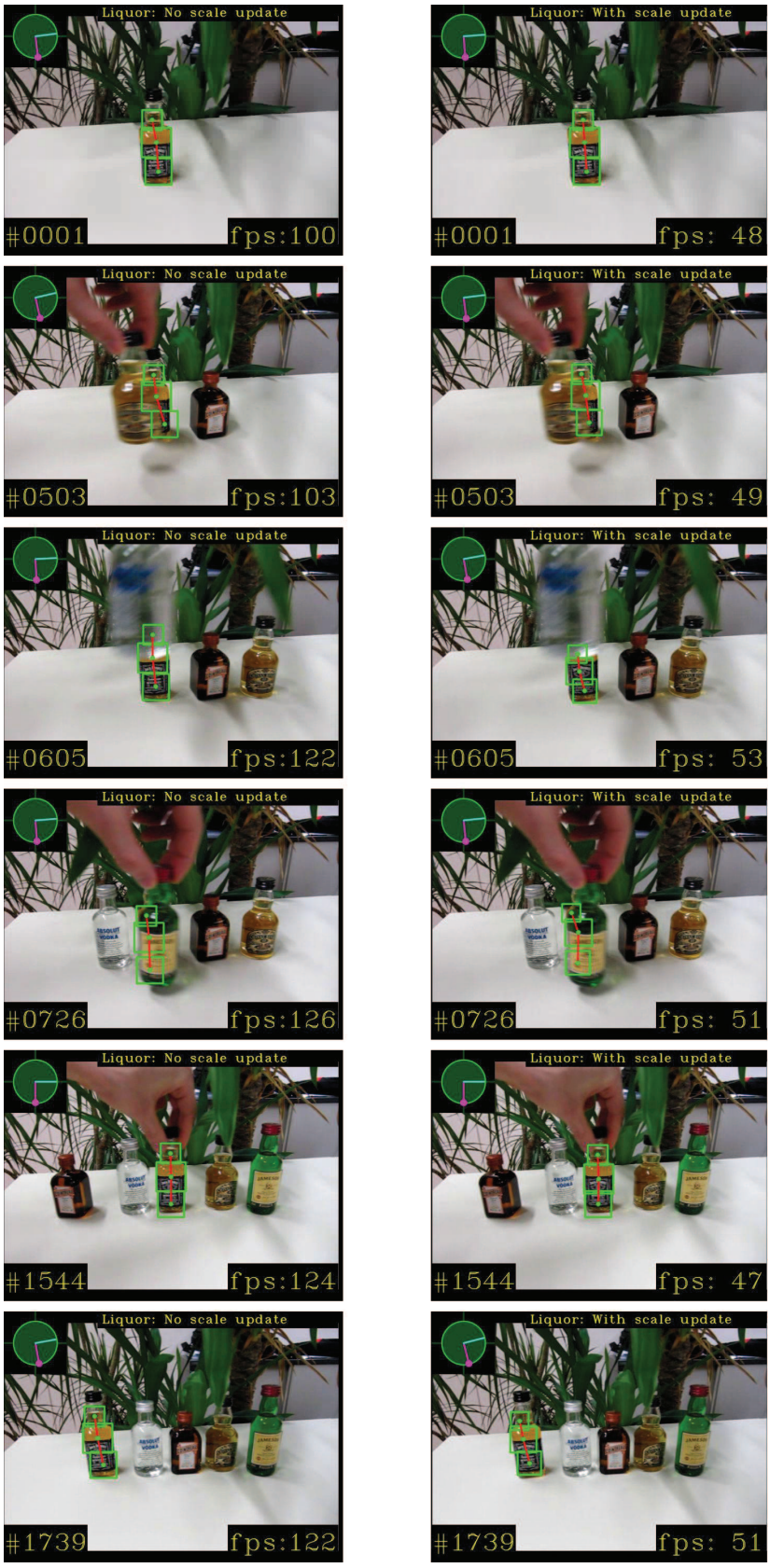


Figure 3.9: *Several frames of* **Liquor** *sequence from* PROST *database, which show tracking results of our method without (**left**) and with (**right**) scale update. In this sequence the target bottle is occluded partially and fully by other bottles many times. Nevertheless, it produces reliable tracks.*

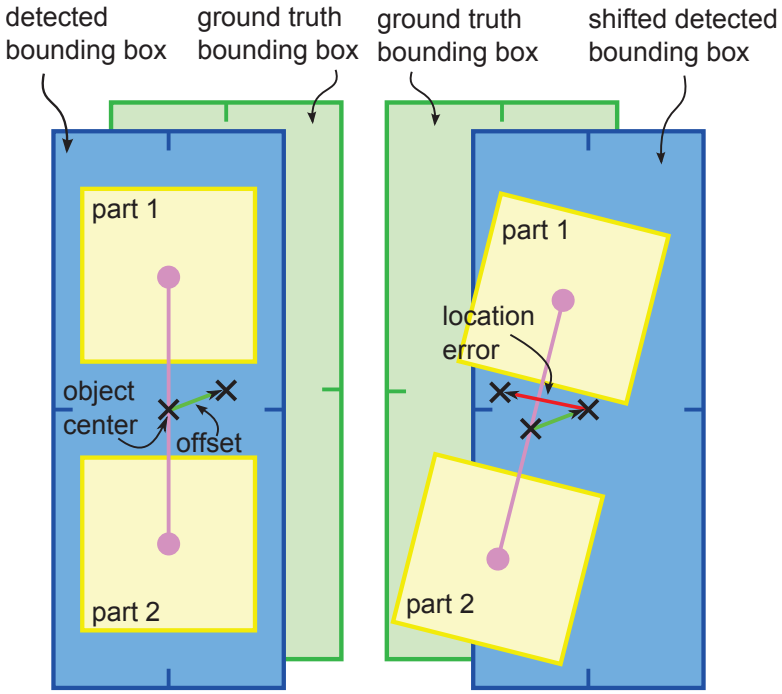
To make a fair comparison, we do not update the scale of our tracker and always output the same size of the ground truth bounding box. Note, however, that in the first frame of each sequence our tracker outputs a center location of the whole object which is slightly misaligned (by several pixels) from the center of the ground truth bounding-box because it averages the locations of all its parts. For this reason, we precompute this misalignment vector in the first frame and fix it for the whole duration of the sequence. Then, in subsequent frames, the tracker shifts our center by the above constant offset, as shown in Fig. 3.10.

The quantitative results are shown in Table 5.3. Our tracker, with constant appearance, approaches the performance of the best algorithms on this benchmark, the TLD [Kalal et al., 2012] and GD [Klein and Cremers, 2011] trackers, while processing videos at $\sim$ **100** `fps`. It performed worse on the **Box** sequence, in which the box changed drastically its appearance due to the heavy illumination change on its reflective surface. Our method performed the best w.r.t. both evaluation criteria on the **Liquor** sequence and on the **Lemming** sequence. For the sake of coherency of the experiments, we also ran our algorithm to detect scale change and observed comparable performance. In this case, the video processing rate was $\sim$ 45 `fps`.

§ 3.6 Conclusions and Discussion

We presented a statistical model for tracking rigid, flexible and articulated objects. In this chapter, tracking performance was evaluated on a standard, challenging dataset of rigid objects. A notable observation is that our tracker struggles with high illumination changes. Hence, our model should benefit from better features by augmenting its appearance term. Several, possibly complementing, improvements are (i) to design an on-line appearance update mechanism, (ii) to integrate such features which are robust to illumination change, or (iii) to explain illumination change. In the future development, we will address these points to improve the appearance term, though here our aim was to propose a general method with the focus on the strong spatio-temporal model showing good results in terms of efficacy and efficiency. Using a remarkably simple, fixed appearance term, the tracking algo-

rithm provides competitive state-of-the-art results at the frame rate up to 100 `fps`. In addition, it estimates the scale and in-plane orientation change of the whole object, while also tracking and maintaining the spatial configuration of object parts. Finally, unlike the above benchmarked methods, the proposed model of our *tracker* can be reconfigured very simply and intuitively to represent more complex objects, as described in the next chapter.



(a) frame 1 (b) frame > 1

Figure 3.10: *Transformation of the bounding boxes of the object parts (yellow) to the detection bounding box of the whole object (blue), having the same size as the ground truth bounding box (green). (a) In the initial frame, the vector offset (green) is computed from the center of the detection bounding box to the center of the ground truth bounding box. (b) In each subsequent frame, once all the parts of the object are jointly detected, their locations are averaged to determine the new center of the object. This center is then shifted by the precomputed vector offset (green) to determine the new, detection bounding box (blue). The length of the offset (red) between the center of this box and the center of the ground truth bounding box (green) is computed as the mean distance precision.*

Method	Sequence				Avg
	Board	Box	Lemming	Liquor	
Pascal overlap [Santner et al., 2010]					
MIF[Wnuk and Soatto, 2011]	92.1	42.9	88.1	75.6	74.7
ORF[Saffari et al., 2009]	10.0	28.3	17.2	53.6	27.3
FragTrack[Adam et al., 2006]	67.9	61.4	54.9	79.9	66.0
MIL[Babenko et al., 2009]	67.9	24.5	83.6	20.6	49.2
PROST[Santner et al., 2010]	75.0	90.6	70.5	85.4	80.4
GD[Klein and Cremers, 2011]	**94.3**	**91.8**	78.0	91.4	88.9
TLD[Kalal et al., 2012]	87.1	**91.8**	85.8	91.7	**89.1**
Ours	90.7	63.1	**91.4**	**96.6**	85.5
Mean precision [Babenko et al., 2009]					
NN[Gu et al., 2010]	20.0	16.9	79.1	15.0	32.8
MIF[Wnuk and Soatto, 2011]	13.7	63.7	19.4	42.5	34.8
ORF[Saffari et al., 2009]	154.5	145.4	166.3	67.3	133.4
FragTrack[Adam et al., 2006]	90.1	57.4	82.8	30.7	65.3
MIL[Babenko et al., 2009]	51.2	104.6	14.9	165.1	84.0
PROST[Santner et al., 2010]	39.0	**13.0**	25.1	21.5	24.7
GD[Klein and Cremers, 2011]	14.7	13.2	28.4	11.9	17.1
TLD[Kalal et al., 2012]	**10.9**	17.4	16.4	6.5	**12.8**
Ours	18.9	43.2	**12.9**	**5.6**	20.2

Table 3.2: *The performance of our algorithm evaluated on the* PROST *database using* Pascal overlap *and* Mean precision *measures. For an overall comparison, we also provide the average score in the rightmost column. Best results are indicated in* ***bold****, for each measure individually.*

- Chapter 4 -

Model configurations for tracking various objects

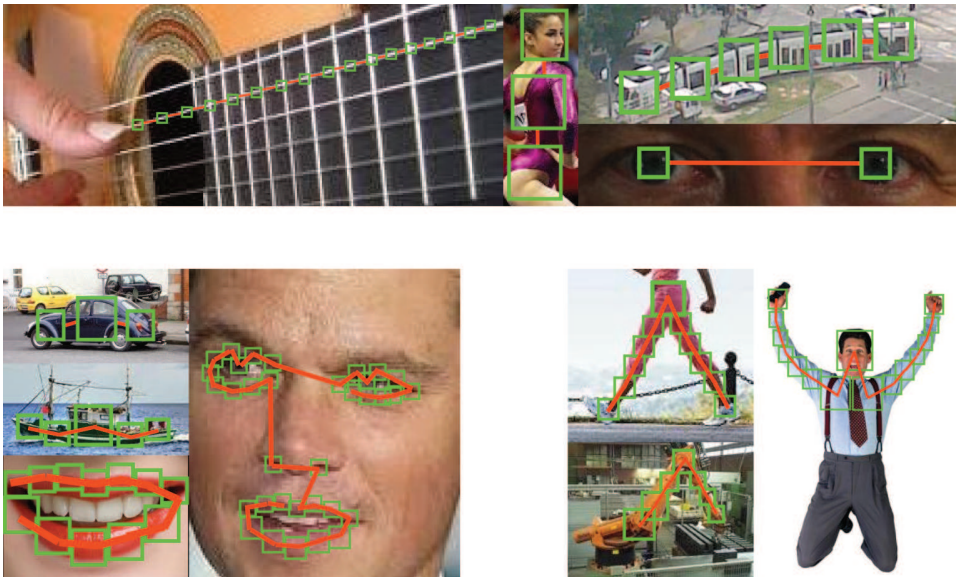


Figure 4.1: *The proposed model can be configured to track a plethora of objects, which differ in shape and motion dramatically. A particular configuration of the spatial layout of object parts categorizes objects as elongated and planar, which can shift, rotate, change scale, and deform by flexing, articulating, and vibrating.*

The goal of this chapter is to demonstrate the utility of the tracking algorithm described in Chapter 3. We experimentally challenge the *versatility* of our model from section 3.2, integrated into the tracking framework from section 3.4. As proof of concept, we show that this *one* model, with particular configurations from section 3.3, can be successfully applied to stably track a *plethora* of various objects that shift, rotate, change scale, and deform by flexing, articulating, and vibrating.

§ 4.1 Introduction

Under our model, objects can be assigned into three, general categories:

- straight objects,
- planar objects,
- heavily deformable objects,

which are defined in section 3.3. For each category, we follow the same manual initialization procedure up to the configuration of the temporal term (3.7). We specify K segments of the object (where only heavily deformable objects have more than one segment). Next, we specify k_i parts of i-th segment. We annotate the image region containing a given object with $\sum_{i=1}^{K} k_i - (K-1)$ bounding boxes that indicate object parts. We then form a chain graph linking the parts. The centers and sizes of the bounding boxes define the initial locations and scales of the parts in the first frame of a video sequence. Afterwards, the temporal term (3.7) is configured differently for the three categories to specify straight or planar layout of object parts. The three following sections describe such particular configurations.

Notably, we do not use any specific motion model, which would be tuned to a particular object. The sole, temporal constraint of our model manifests itself in the form of the global orientations over the segments. Estimated in the previous frame of the video sequence, these orientation priors are exploited in the current frame to keep the spatial coherency of the parts. In this way, the priors regularize the dynamics of each object by favoring shift motion and slow, incremental rotary changes. The influence of this inertial behavior of the model is analyzed in this chapter.

In order to model a particular object shape and its deformation, the initialization procedure[1] differs w.r.t. the number of segments, the number of parts and their locations and sizes, and the chain graphs. However, the histograms, mean distances,

[1] For some sequences we experiment with alternative settings to test the behavior of the tracking algorithm.

and mean angular displacements are then computed automatically from these *one-shot* annotations, always *in the same manner* as explained in section 3.5.1. Also, the remaining parameters of the model are *fixed* for all the sequences. Furthermore, even though we assign a coordinate system to each segment with a particular orientation, it is *not* necessary to do so to control the bend of each segment. If the true orientation is not required during tracking, the object can be generally considered as planar with coordinate system having arbitrary orientation in the first frame.

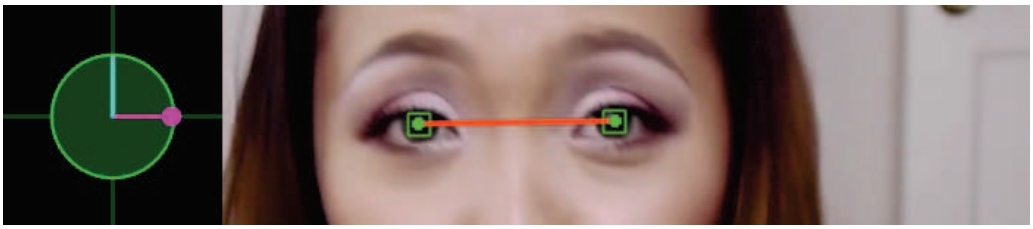
Finally, the following sections are organized according to the level of complexity of the model, which aims at explaining a range of object shapes and motions. This is evaluated qualitatively on the sequences listed in Table 4.1.

Sequence	**Main Challenges**
Eye pupils	occlusions, blur, fast shift
Skater fall	fast rotation, low contrast and quality
Tram	3D motion
Guitar string	long, thin, textureless object
'I love you so' (clip by Cassius)	deformations, occlusions
Face	3D motion, deformations
Robot welding	illumination
Med. thread[Padoy and Hager, 2012]	long, thin, heavily deformable object
Gestures1[Gomez Jáuregui et al., 2010]	self-occlusions
Gestures2[Gomez Jáuregui et al., 2010]	heavy foreshortening
Friends1[Sapp et al., 2011]	fast articulation
Friends2[Sapp et al., 2011]	heavy foreshortening

Table 4.1: *Listed sequences with their main challenges. Unreferenced sequences were downloaded from YouTube.*

§ 4.2 Tracking straight objects

Eye pupils: A successful localization of eye pupils can spur many applications. It can be integrated into gaze estimation to interact with machines or for medical studies such as psychology, sociology, or neuroscience. Moreover, the human gaze is correlated with human movement in particular direction, as both functionalities are coupled through human visual feedback. This might be exploited by a human tracker as a directed motion prior with a premise that *one moves where one looks*.



(a)

(b)

Figure 4.2: *Representation of eye pupils.* (a) *Initial location and size of both pupils which are enforced to lie within some distance from each other. On the left, the initial orientation is given.* (b) *Corresponding two-part model, configured with a single global orientation.*

Figure 4.3: Eye-pupils *sequence showing the eyes that shift and often blink. Eye-blinking occludes the pupils in frames* #20 *and* #36 *but the tracker remains untroubled, as shown in the subsequent frames. In this case, the head shifts fast causing blur artifacts, as shown in frame* #125. *Here, the tracker loses on accuracy but it recovers soon to yield a stable track.*

The eye pupils are represented as a pair of parts, connected by a distance constraint and configured with a single global orientation over the parts, as depicted in Fig. 4.2. Since the motion of eye pupils is correlated, our model-based tracker naturally exploits this feature of the human visual system by the distance constraint between the pupils. This is the simplest representation of our model.

The results are shown in Fig. 4.3. The eye tracker ran on several additional sequences, and in all cases it was stable during eye-blinking. Its robustness against this type of occlusion might be explained by the alike appearance of the pupils and eye-lashes. The main color of the pupils and the region around eye-lashes is black. As the model represents the appearance by color histograms, the eye-lashes do not distract the tracker while occluding the pupils.

Upper body: Smart home surveillance systems could take advantage of tracked human upper body. Apart from body's location, an activity recognition system might use a richer tracking cue in the form of in-plane orientation of the body, which is output by our tracker. This motion cue could facilitate recognition of basic daily activities like walking, running, standing up, sitting down, lying down, and most importantly falling down.

The segment of the upper body is split into $k = 3$ semantic parts, i.e. head, chest, and hips, which lie on the spine. It is parameterized by a single global orientation, as shown in Fig. 4.4. Potentially, one might also specify additional parts, e.g. neck, or specify non-semantic parts to describe the geometry of the upper body with finer details of appearance.

(a) (b)

Figure 4.4: *Upper body of an ice-skater is represented as head, chest, and hips, which lie on (elastic) human spine.* (a) *Upper body, which has an assigned initial global orientation, is split into three parts with annotated location and size.* (b) *Corresponding model configured with global orientation.*

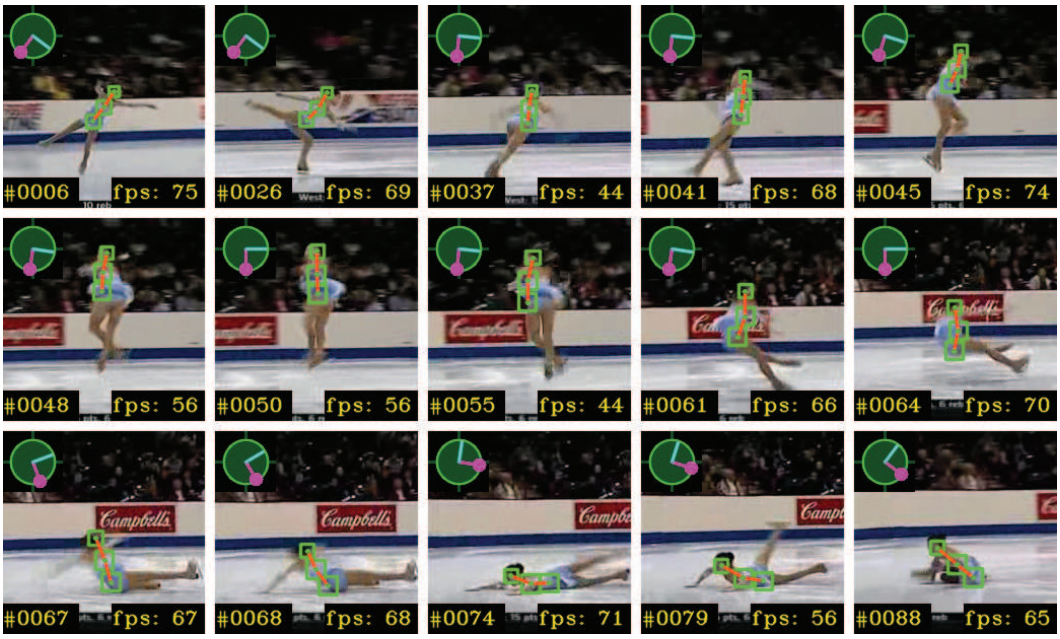
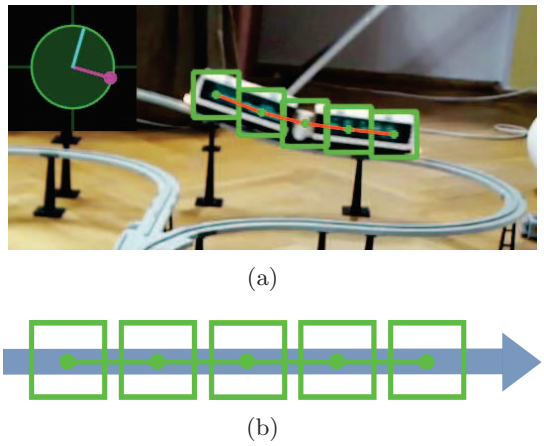


Figure 4.5: Skater fall *sequence shows the ice-skater performing an unsuccessful loop jump, which is followed by a very fast fall, as indicated by the frame index. Also, note low video quality and low foreground/background contrast in the bottom row of the sequence. The tracker estimates correctly the scale and orientation change. The obtained locations of the parts in frame* #79 *show the elasticity of the object, which is built into the model.*

Despite a rotary and very dynamic falling down motion of the skater in Fig. 4.5, the tracker can estimate reliably the location, orientation, and scale of the upper body. The prior orientation refers to the previous frame. This implies that the faster the rotary motion, the more penalty the model puts on the true orientation in the current frame – a limitation of our model. Nevertheless, the tracker was able to maintain all the upper body parts on the straight line in the current frame during the fast motion.

Street vehicle: Elastic street vehicles such as long buses, cars with trailers, trucks, trains, or trams can be represented with our model in a straightforward fashion. Any example of such elongated objects can be split into a chained assembly of parts. Tracking them could increase the awareness of outdoor surveillance systems. The systems would be able to e.g. report if a vehicle is not speeding or deciding who had the fault in causing an accident - a tram or a car. Additionally, by building such complex semantic summaries of street surveillance videos, the system would be able to store only *salient* data, in order not to overload the capacity of video storage systems, which is already an increasing problem.



(a) (b)

Figure 4.6: *Representation of a toy tram.* (a) *Initial location and size of the tram parts.* (b) *Corresponding model configured with single global orientation.*

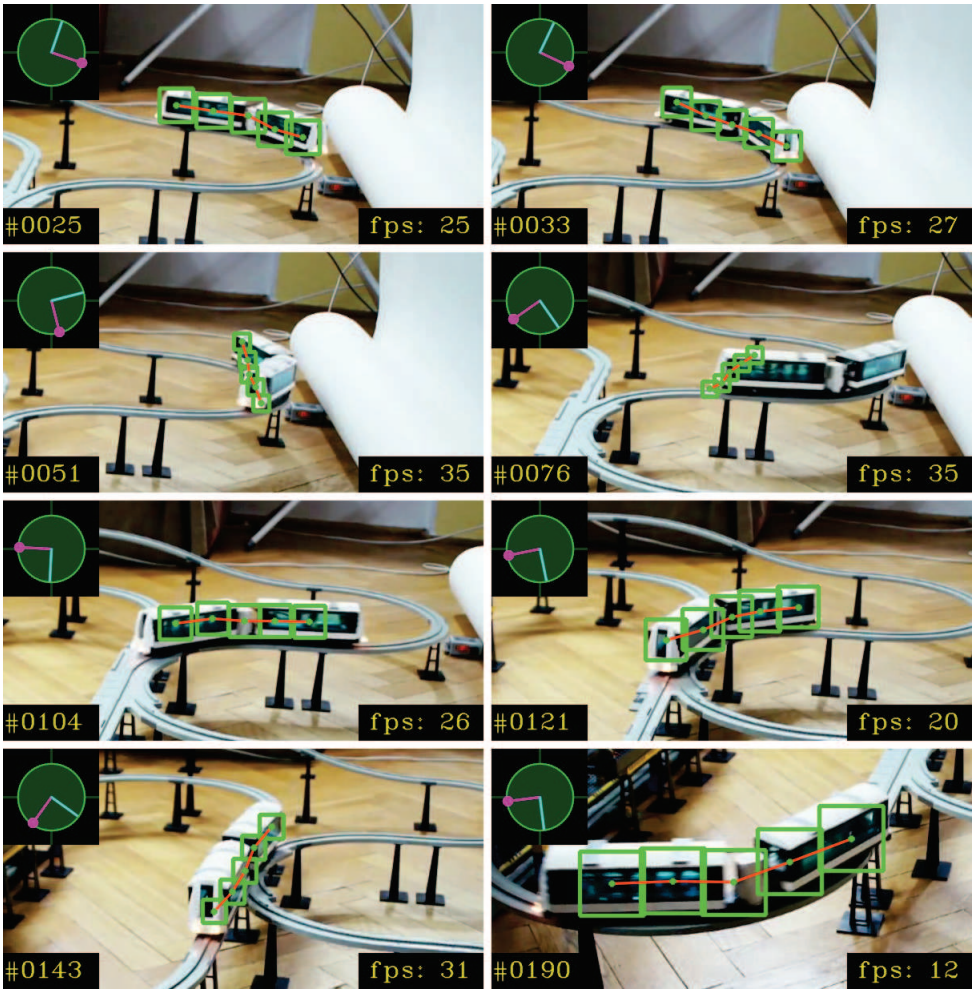


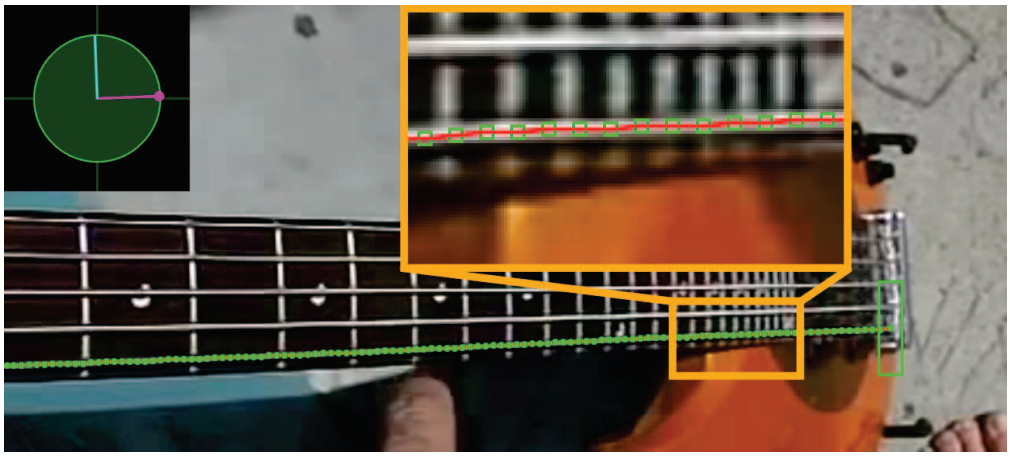
Figure 4.7: Tram *sequence features a toy tram that makes several turns on the railway. The tram bends, e.g. in frame* #121*, undergoes in-plane and out-of-plane orientation change, leading to foreshortening in frame* #51*, and changes scale, as shown in frame* #190*. Change in* 3*D pose results in change in appearance but the tracker yields a good track because of its flexibility.*

Street vehicles may differ in length and therefore in the number of parts. Here, we demonstrate an example of a toy tram with 5 parts, as shown in Fig. 4.6, which describe the image region containing the whole tram. The consecutive parts in the chain are linked pairwise by local distances. The tram is configured with a single global orientation, as it is assumed the above vehicles are not articulated.

We tried an alternative configuration for the tram as well. We assumed the tram was indeed articulated and represented it as a composition of front and back wagons, which were linked by a hinge part in the middle. Hence, each wagon was configured with its own global orientation. We found that the first model, configured as in Fig. 4.6(b), was able to maintain the parts on a straight line during the out-of-plane motion between frames #0033 and #0121 in Fig. 4.7, when the object changed its appearance. On the other hand, the second, articulated model, collapsed both segments into one and tracked only the front wagon, as both wagons looked alike. Therefore, in this case we observed that it was favorable to enforce the elasticity of the object instead of allowing it to articulate. Such a configuration of the model proved to be robust against foreshortening, which was caused by out-of-plane motion.

String/Wire: An example of a straight, very long, elastic object is a string, which is mounted on e.g. a guitar, as depicted in Fig. 4.8(a). The tracked vibrations of the string could be animated in computer graphics. Another instance of this object category is electric wire, mounted on e.g. electric power transmission plant, which can vibrate due to wind, tampering, or earthquake, and which can stretch and shrink due to temperature change. Since the proposed approach can register vibrations of such long thin objects, it might also be applied in the safety systems of aerial lifts, which are used in e.g. skiing tourism.

We observed that the registered motion of the string becomes more precise and thus more realistic with the increased number of its parts. Therefore, the string was divided into 114 parts of size 4×4 pixels, as shown in Fig. 4.8(a), and the parts were linked with distance constraints. We expected the vibrating string to deform only slightly. Hence, this *entire* elongated segment of parts was configured by a single global orientation, as shown in Fig. 4.8(b).



(a)

(b)

Figure 4.8: *Representation of a vibrating guitar string.* (a) *The long, thin, and textureless string is partitioned evenly into a large collection of small parts, which lie at equal distances from each other. All the parts have whitish color, as shown in the zoomed image. The last part (on the far right) is larger and has different texture to prevent the model from sliding along the string.* (b) *Corresponding model of the string. It is configured only with a single, global orientation, as the deformations of the string are incremental.*

Since all parts had a homogeneous texture, we had to account for the lack of discriminative appearance by two modifications of our default model. Firstly, the chain of parts was anchored with an auxiliary, well textured part. However, even though this prevented the model from sliding freely along the string, the tracker was unable to update the scale correctly and it blindly shrank the model, as shown in Fig. 4.9. In view of this, the second modification relied on fixing the scale of the parts for the whole duration of the sequence. In effect, the tracker localized the parts accurately, as shown in Fig. 4.10, because the distance constraints were not rescaled incorrectly. Another workaround for this problem might be to assign a much lower variance in the appearance term of the anchor part. In this way, the appearance of the whole string would be dominated by this well textured part. Alternatively, in a controlled setting, a string, a wire, or a thread, could be textured, which would simplify the tracking problem.

The sequence was recorded at 1200 `fps`, as the vibrations of the string oscillate rapidly.

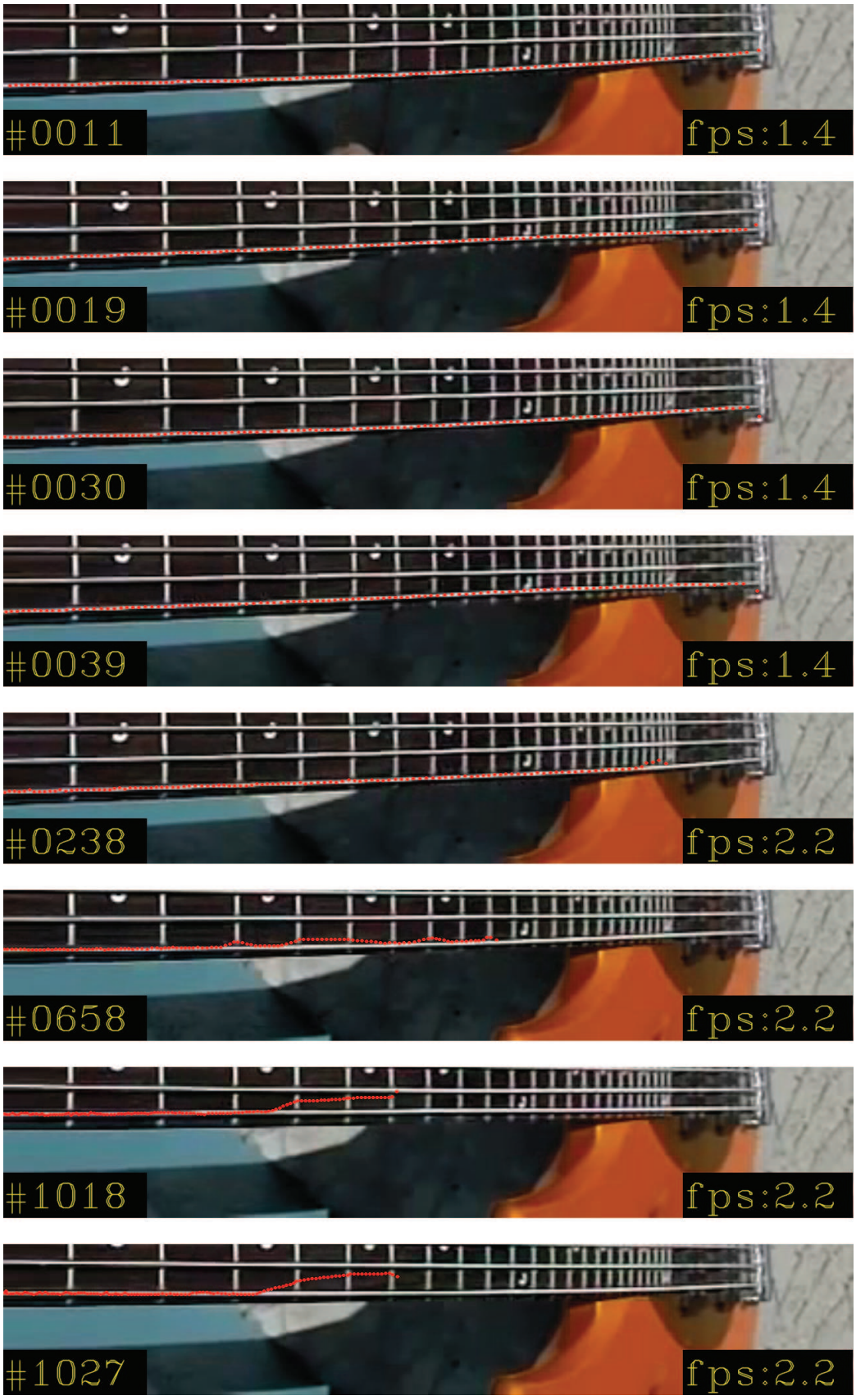


Figure 4.9: Guitar string *sequence showing vibration of the string, registered at* 1200 *fps. The algorithm updated the scale of each part but was unable to track the string correctly. As the string was textureless, the model could not discern between hypothesized scales of the parts. The incorrectly estimated scaling factor rescaled the distance constraints between the parts and, in effect, the model shrank.*

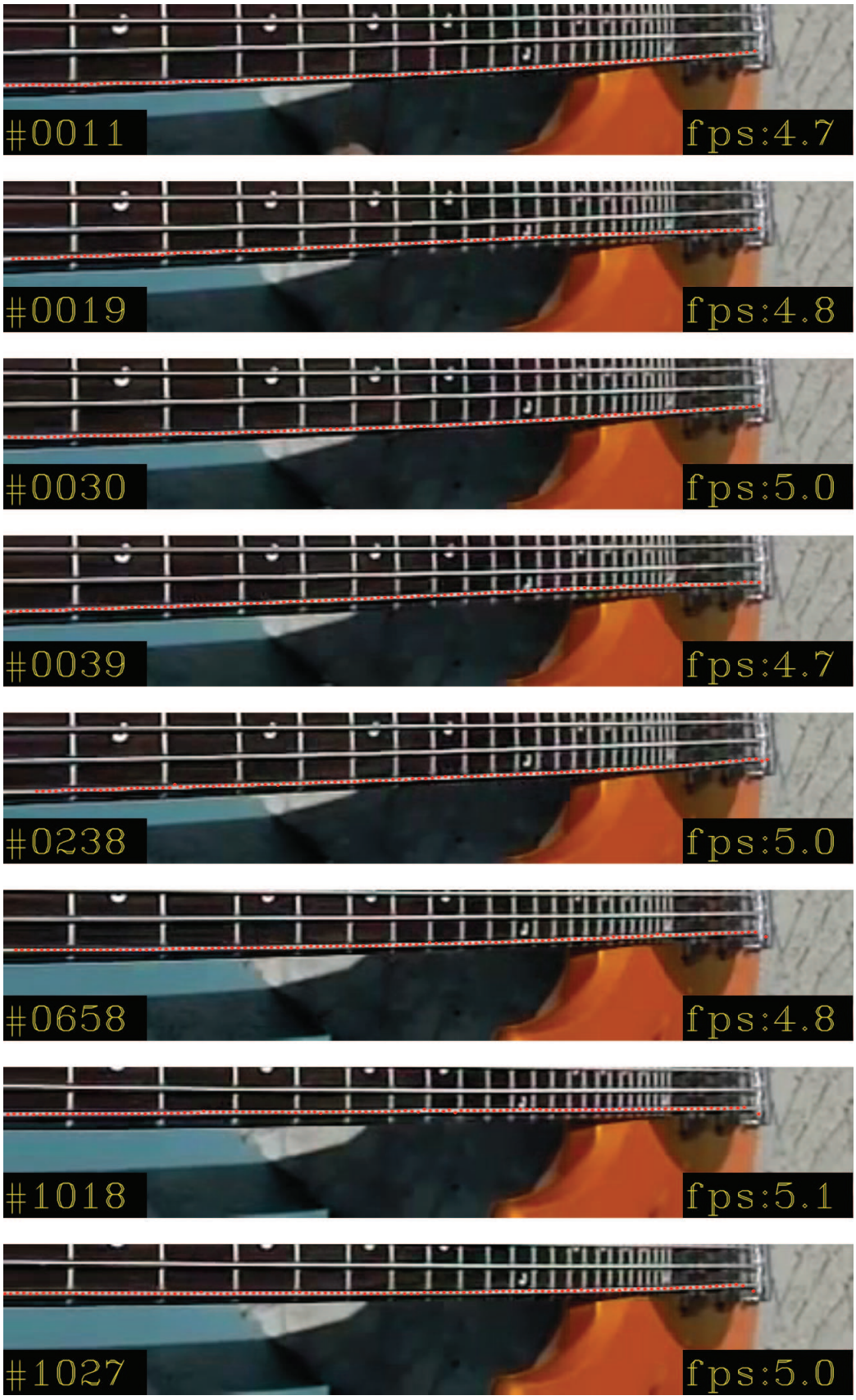


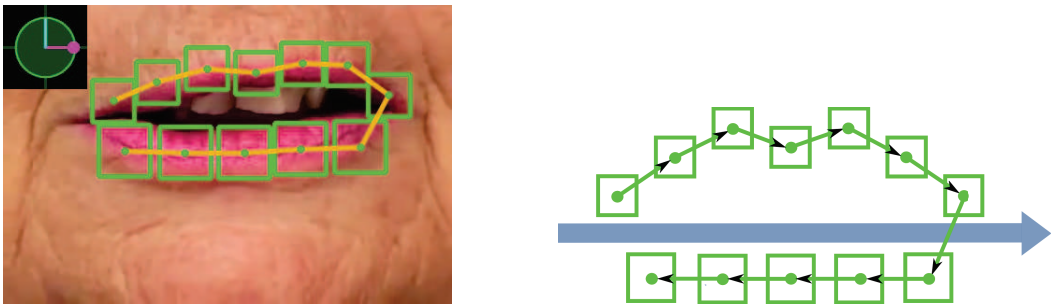
Figure 4.10: *Sequence showing vibration of the string, same as before. However, in this case, the algorithm did not update the scale of each part. This improved the tracking results, because the spatial constraints maintained the parts at preferred distances.*

§ 4.3 Tracking planar objects

Lips: Capturing the motion of the lips could be exploited in numerous ways, e.g. for lipreading, for facilitated learning of accents of foreign languages by comparing registered to the modeled lips motion, or for a human-machine interface (HMI).

We represent the curved image region of the lips with 12 parts and link them with local distances, as shown in Fig. 4.11(b). Each part contains colors of lips and skin around lips in order to capture more discriminative texture. We then configure the whole assembly of parts with a single global orientation. Finally, the angular displacements of the consecutive pairs of parts are computed automatically from the global orientation based on their initial locations. They are fixed for the whole duration of the sequence.

Even though the angular offsets are learned *only* from the first frame, they are represented as Gaussian variables. This allows our model to explain severe deformations of the lips as shown in Fig. 4.12. Moreover, the tracker remains stable during occlusions as the occluded parts are supported by their neighbors, owing to geometrical constraints built into the model. Alternatively, the lips could be modeled as an articulated object with two elastic segments, where one of the lip corners could be the hinge part. This configuration yielded results, which were comparable to the default configuration.



(a) (b)

Figure 4.11: *Part-based representation of human lips.* (a) *Initial location and size of the parts.* (b) *Corresponding model configured with single global orientation, with respect to which the angular displacements between consecutive pairs of parts are obtained. For instance, the angular displacements for the lower lip is* π.

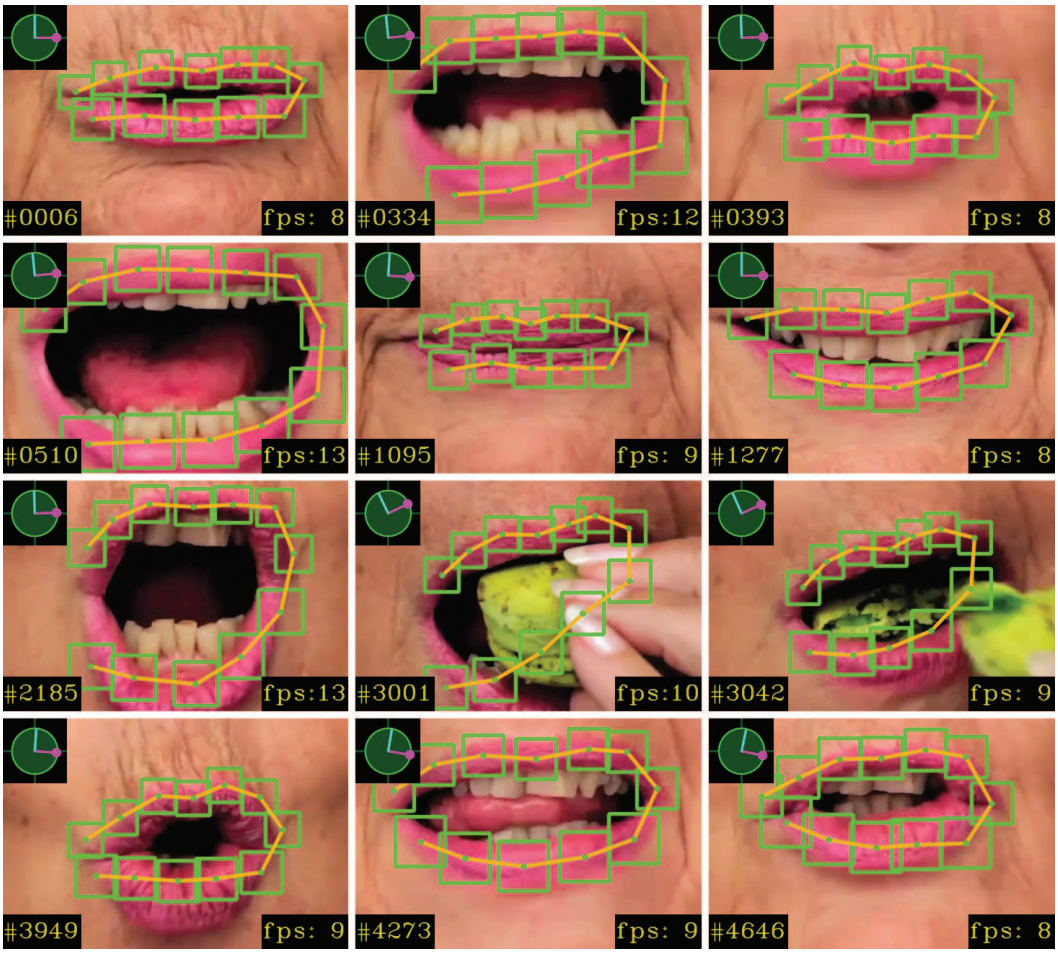
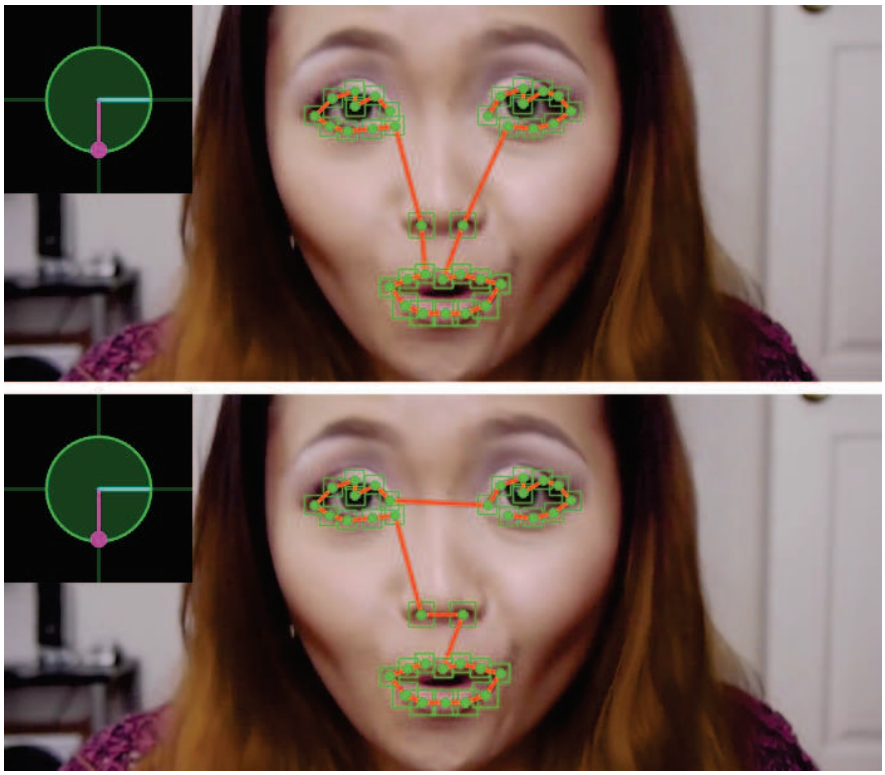


Figure 4.12: 'I love you so' *sequence shows heavy deformations of the lips during singing. The lips stretch and shrink as depicted in the frames* #0510, #2185, #3949. *Also, the model can handle and recover from partial occlusions well, when the person eats the green macaron in frames* #3001 *to* #3042. *The true orientation suffers from this occlusion, though the tracker recovers it correctly afterwards. The predefined spatio-temporal constraints allow the tracker to lock the parts on the lips, even in the presence of the tongue which looks similar, as shown in frame* #4273.

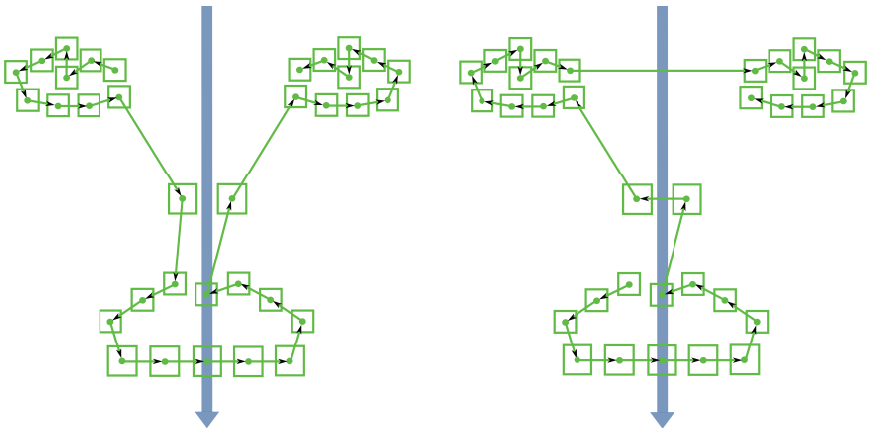
Face: Registering the motion of a human face can be used for animation in computer graphics, or expression and emotion understanding. If high precision was required, the proposed tracking might be used as the initialization for even finer search of facial landmarks.

We represent here the facial landmarks of a human with 34 parts, belonging to lips, nostrils, eye contours, and eye pupils. A chain structure is constructed by linking the parts which are visually close to each other. This approximative approach might not form the most optimal chain of parts. However, two examples of chain structures, shown in Fig. 4.13, performed similarly. Both sequences from Fig. 4.14

demonstrate that the models explain well the deforming lips and eye contours. Note, though, the results might be less satisfactory, indeed, had the parts been chained randomly.



(a)



(b)

Figure 4.13: *Two example representations of a human face.* (a) *Initial location and size of the parts of the face.* (b) *Corresponding models configured with single global orientation, which is the reference for the angular displacements between consecutive pairs of parts. The parts are linked with two different chain structures to test their relative performance.*

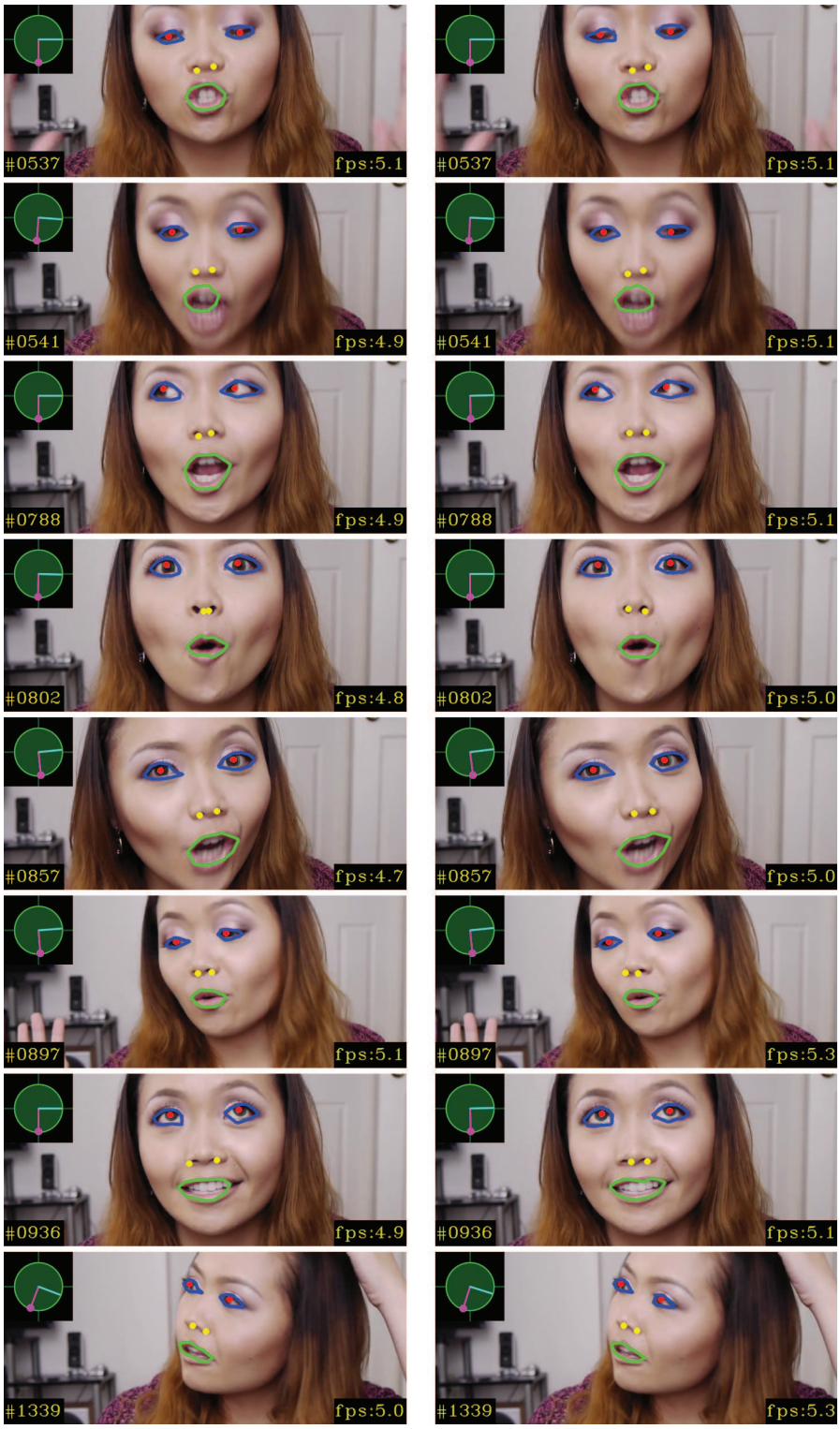
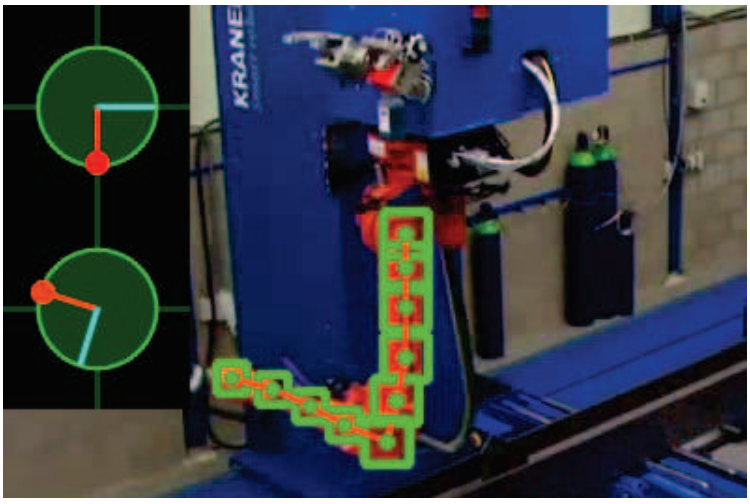


Figure 4.14: Face *sequence shows emotional facial expressions, changes of scale, head pose, and gaze, which all deform our model. Two different chains produce similar results, as shown in the left and right columns. Since the first chain lacks the graph edge between the nostrils, their locations are sometimes too close, as shown in frame* #802*. Frames* #1339 *and* #897 *show the in-plane orientation is estimated correctly and the tracker is robust against mild out-of-plane rotations in frame* #1339*. Also, scale change is tracked well between frames* #857 *and* #897*. The tracker deforms the model correctly to follow the motion of the eye-pupils throughout the sequence. It can also recover from severe blur artifacts in* #541*, caused by rapid head motion.*

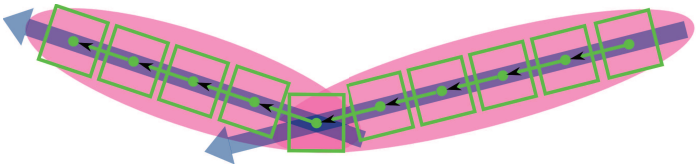
§ 4.4 Tracking heavily deformable objects

Robot arm: Cooperation of robots with humans-in-the-loop is an emerging trend in assembly line production. Owing to their speed, force, and precision, the robotic manipulators could augment the dexterity of people and increase their efficiency. Consequently, close interaction between both modalities requires imposing high safety standards to avoid collisions. The internal safety systems of the robots could be supported with external visual tracking systems. This could improve the redundancy of the safety system in case of malfunction of one of its components.

The proposed tracking system can be configured to track the articulated movement of a robotic arm and report on its pose independently from the robot controller. In the considered case, the arm is split into two $K = 2$ segments, which are split further into $k_1 = 6$ and $k_2 = 5$ parts, such that the parts occupy the whole image region of the arm, as shown in Fig. 4.15.



(a)



(b)

Figure 4.15: *Representation of robotic arm by two segments.* (a) *Initial location and size of the parts and orientations of both segments.* (b) *The model is configured by two segments with their own, orientated coordinate systems. The segments share one hinge part, which joins them.*

The tracker can follow the articulated movement of the robot arm as shown in Fig. 4.16. It was observed in section 3.5.2 of the previous chapter that the algorithm lost track of one object because of sudden change of illumination. This is here shown again. When the robot starts to weld, welding emits additional illumination which largely affects the appearance of the bottom segment of the arm. Eventually, the tracker drifts away from the segment.

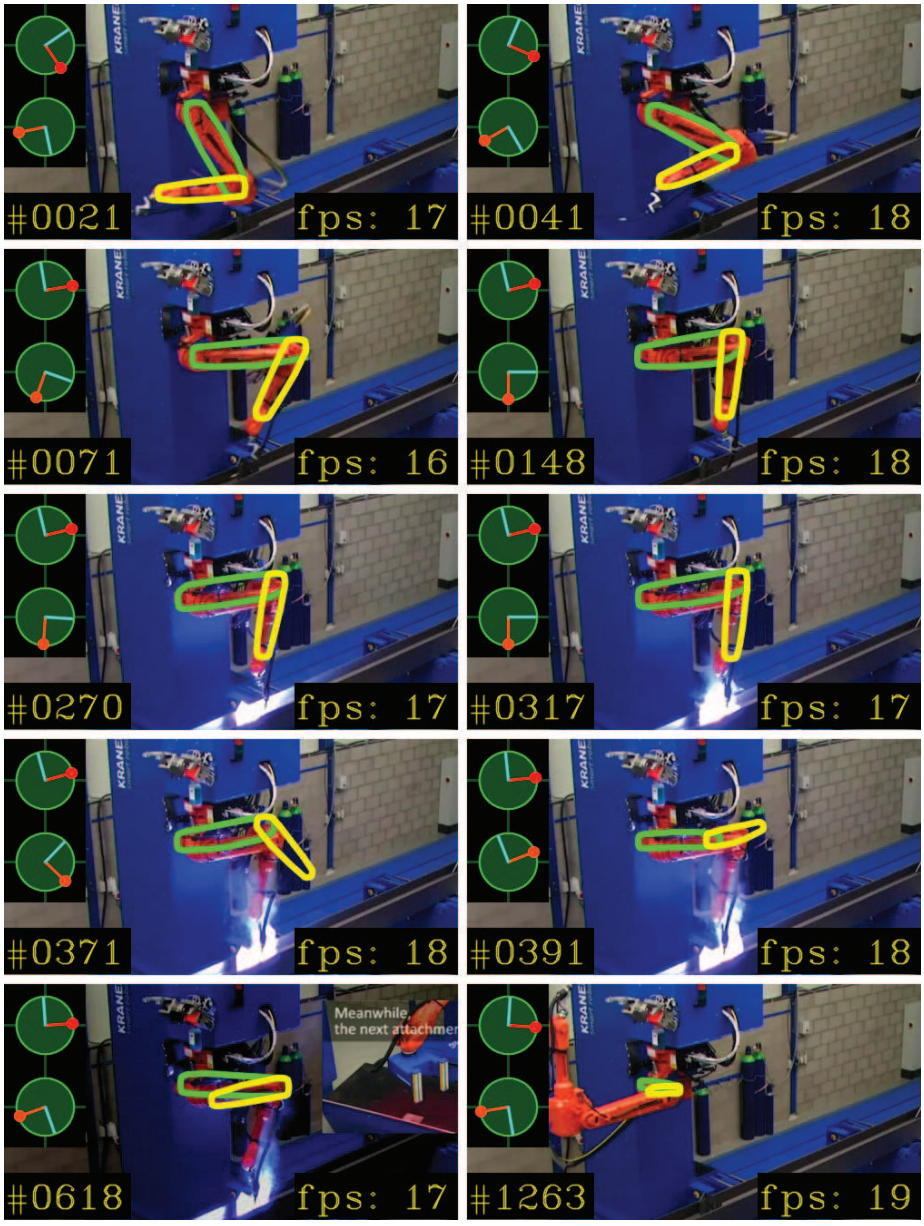
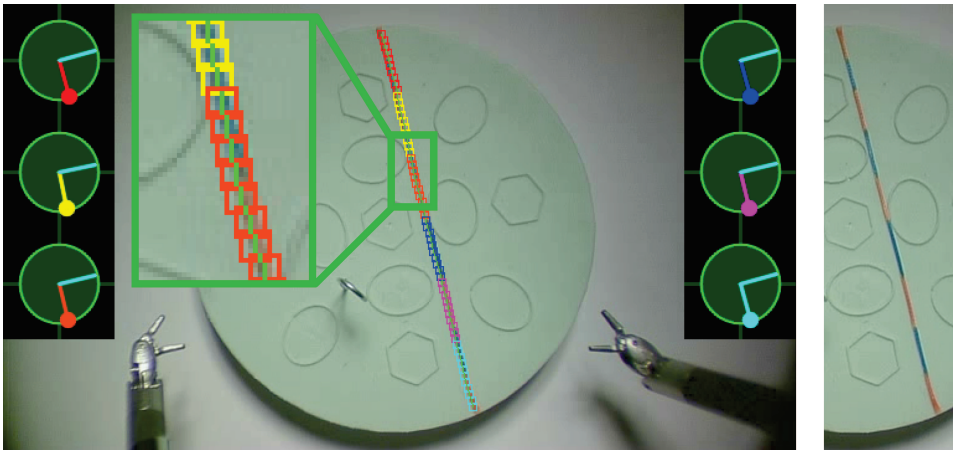


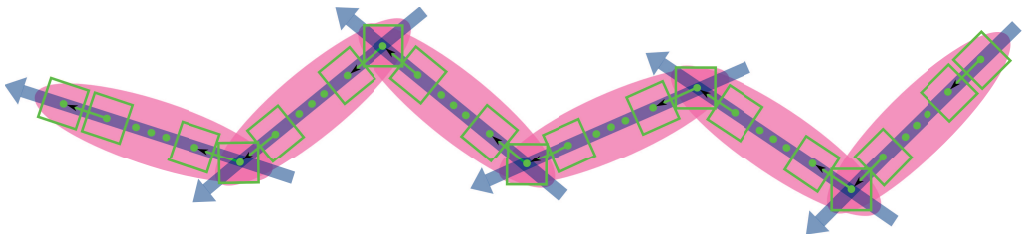
Figure 4.16: *Robot manipulator adjusts its pose for welding in sequence* Robot welding. *The robot articulates its two rigid segments up to frame* #148 *and begins to weld, which causes a gradual change in illumination. Eventually, the appearance of the lower segment is compromised to a great extent around frame* #371. *In effect, the tracker locks the lower segment on the upper segment as they both look alike.*

Medical thread: A surgical robot can perform complex surgical tasks with much higher precision than humans leading to reduced negative aftereffects of the operation. Tracking the medical thread would extend the abilities of the robot by sewing and knot tying with laparoscopic instruments [Padoy and Hager, 2012].

Unlike the guitar string, whose deformations are incremental and oscillatory, the thread is an elongated object, which deforms heavily and unsystematically. Thus, it is represented here as an articulated, piece-wise linear object of $K = 6$ segments, which are equally[2] divided into 11 parts of size 6×6 pixels, as shown in Fig. 4.17(a). The parts of each segment lie densely on a straight line that specifies the orientation of the segment, as shown in Fig. 4.17(b). The model has 61 parts.



(a)



(b)

Figure 4.17: *Representation of a medical thread, which is divided into six elongated segments of parts.* (a) *The thread is partitioned evenly into a large collection of small parts, which lie at equal distances from each other. All parts have blueish or reddish color, as shown in the zoomed image. The initial orientations of the segments of parts are shown on the sides, with colors corresponding to the segments. The cropped image on the right shows the whole thread, which is made of a textured material.* (b) *The model is configured by six chained segments of parts with their own, orientated coordinate systems. The segments articulate around the hinge parts, so the model can explain heavy deformations of the thread.*

[2]In this experiment, it is assumed that the deformation points are not known. Though, they might be predicted based on the known gripping motion of the robot pincers during surgery.

Even though the thread is represented as a collection of piecewise linear segments, it can be observed in Fig. 4.18 that the flexibility of the object is recovered well. It was delightful to see that the tracker matched the articulated model precisely to high curvatures of the thread. The piecewise linearity is apparent at the hinge parts but it can be smoothed in postprocessing. The tracker also stabilized both ends of the thread correctly, which is challenging because of no constraints at the ends of the thread [Heibel et al., 2009]. This satisfactory behavior of the model might owe to the orientation constraints, which favor local shift motion. The tracker was also tested without scale update. It had comparable quality and its frame rate increased from ~ 2 `fps` to ~ 10 `fps`.

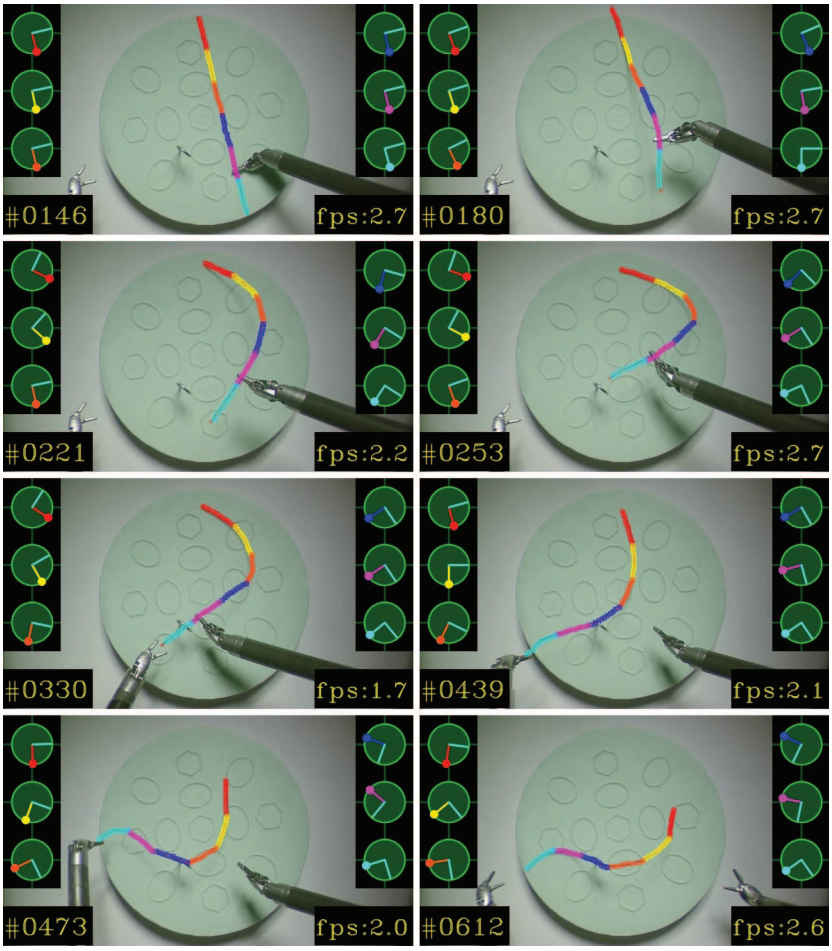
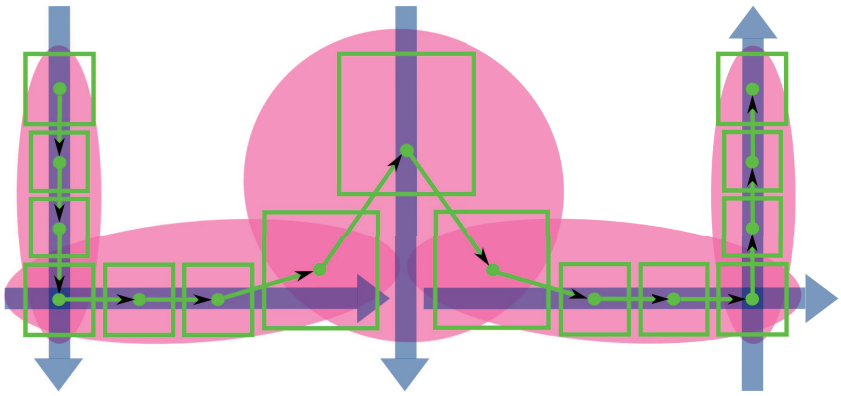


Figure 4.18: *Two pincers of a surgical robot manipulate a medical thread during a simulated surgery. Initially, the thread is straight, but then the robot deforms it heavily. The parts of each segment are modeled to lie on a straight line, whose orientation is updated over time, as shown on the image side. However, the soft, angular constraints allow each segment to bend, which can be observed e.g. for the light blue (last) and yellow segment in frame #439. The model can also account for heavy deformations throughout the sequence.*

Gestures: Humans can interface with machines through gestures. A notable example of a recent and successful gesture control system is Kinect©. Gestures can also be tracked to recognize activities of humans and to facilitate human-object-machine interaction, for instance on assembly lines.

(a)



(b)

Figure 4.19: *Representation of a human full upper body composed of hands, lower and upper arms, shoulders, and head.* (a) *Initial location and size of the parts for four sequences. On the left, initial orientations of the coordinate systems assigned to the segments of the parts.* (b) *Corresponding model configured with five segments. Each segment has its own coordinate system. The five oriented coordinated systems, shown in* (a) *from top to bottom, correspond to five segments, shown in* (b) *from left to right, respectively.*

The full human upper body is represented as a semantic composition of hands, lower arms, upper arms, shoulders, and head. It is partitioned into 15 non-semantic parts, as shown in Fig. 4.19(a). Since the upper body undergoes heavy deformations, it is modeled as an articulated, hybrid object composed of 5 rigid segments. The segments comprise two elongated lower arms, two planar upper arms, and one planar segment formed of shoulders and head. The upper arms are modeled as planar segments as they are anchored at shoulders, which are usually angularly offset w.r.t. the upper arms. The orientations of respective segments are updated independently. Alternatively, the orientations could be updated jointly, should the segments articulate at their revolute joints in constrained fashion.

Tracking gestures is very challenging and the algorithm is evaluated on a series of experiments. Two sequences in Fig. 4.20 show the same person performing gestures at different levels of difficulty. The person slowly waves his forearms, which cross and make in-plane and out-of-plane rotations. The tracker can estimate well the articulated motion of the arms undergoing in-plane orientation change. Interestingly, owing to the spatio-temporal constraints of the model, it does not confuse alike looking forearms when they cross. However, the tracker struggles with heavy forearm foreshortenings, caused by the out-of-plane rotation of the forearm segments. This behavior of the tracker is expected though. Foreshortening results in rapid change of in-plane orientation of the forearm segment. This is penalized by the tracking system, which favors slowly rotating segments. In effect, the parts belonging to this segment are missed by the tracker.

Next, two sequences in Fig. 4.21 show two actors gesturing in similar fashion but much more rapidly than in the previous sequence. The tracker successfully localizes all the parts in the left sequence despite fast in-plane rotation of one segment. However, in the right sequence it is again confused by frequent and rapid foreshortenings, which result in the lost track of the lower arms. The tracker is able to recover though from the repeatedly lost track of the arms. Extending the 2D coordinate systems over the segments into 3D coordinate systems is an interesting future work. In particular, the 3D systems could be supported by optical flow, which could hint at the 3D motion of the object.

Figure 4.20: *Person gestures gently in* Gestures1 *(left) and* Gestures2 *(right) sequences. For better visualization, the segments are shown instead of the parts. In the left sequence, the arms articulate with in-plane rotation and the tracker correctly localizes the parts building the segments and the orientations of the segments, as shown on the right side of each frame. Even though the person is symmetric in appearance, the tracker is not confused when the arms cross in frame* #1024*. In the right sequence, the tracker is challenged by the foreshortenings of the lower arms. It loses track in frame* #1052 *and recovers in frame* #1195*. It then loses track again in frame* #1197 *but does not recover. These heavy foreshortenings change the orientations of both segments too fast for our model.*

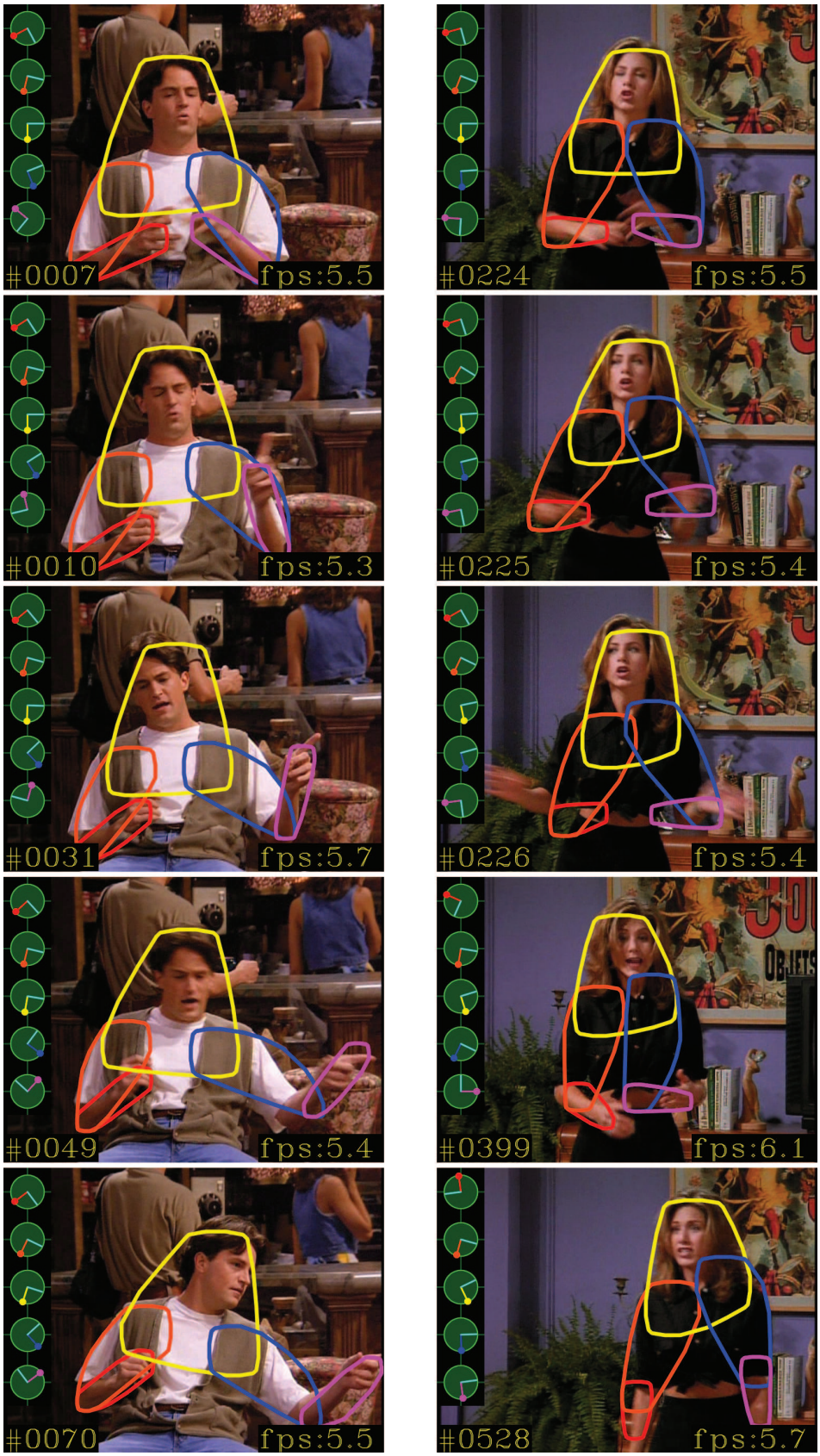


Figure 4.21: *Sequences* Friends1 *(left) and* Friends2 *(right) feature two actors gesturing impulsively. In spite of the rapid in-plane rotation of a lower arm in the left sequence, the algorithm yields a good track because the model can explain this type of orientation change. However, in the right sequence, the gestures are tracked not always correctly. The model is unable to explain the full foreshortening of the lower arms. This is apparent in frames* #225 *and* #399*, where the tracks are satisfactory only for the other body parts, which move affinely.*

§ 4.5 Conclusions

In this chapter, the proposed model-based tracking method was evaluated qualitatively on various challenging sequences. It is demonstrated that the model allows for tracking objects with various shapes that can shift, rotate, change scale, and deform by flexing, articulating, and vibrating. The model can be easily *reconfigured* to track rigid, flexible, and articulated objects, whose parts can have straight and planar layout. Specifically, it has been shown that the method can be applied to track:

- eye pupils,
- human upper body: head, chest, and hips,
- tram,
- guitar string,
- lips,
- facial expressions,
- robot manipulator,
- medical thread,
- human gestures.

The initialization differed for each object by the number of oriented segments, the number of their parts, and the chain that linked the parts. Then, initial one-shot annotations of object parts allowed automatic computation of the subset of remaining parameters. This automated procedure was the same for all the objects. Likewise, the rest of the parameters were fixed, without tweaking them separately to each object sequence.

The complexity of the matching algorithm grows linearly with the number of the parts. Therefore, the tracker works at high frame rates, even for complex object structures of many parts.

- Chapter 5 -

Efficient Convolution of Haar-like Features

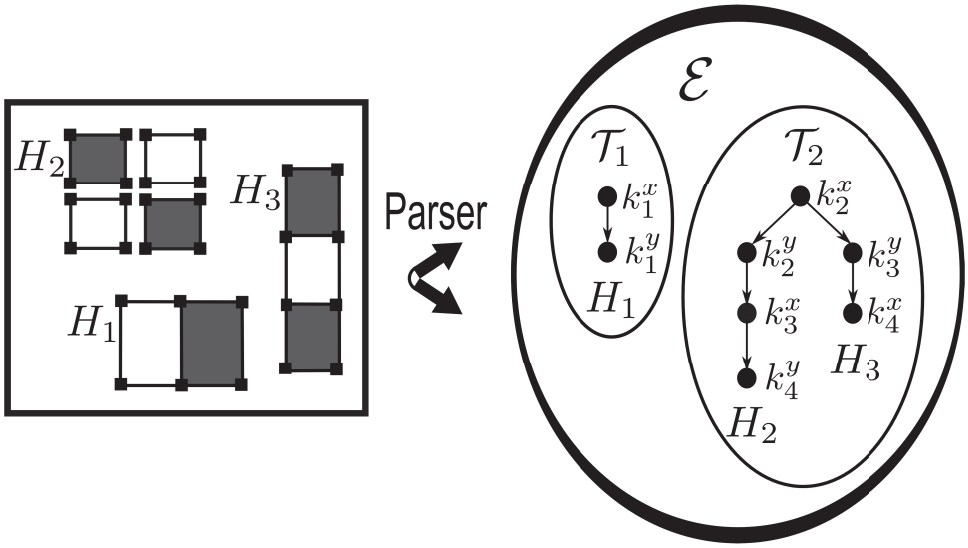


Figure 5.1: *Proposed parsing scheme transforms Haar-like features H, contained in a scanning window on the **left**, into an ensemble $\mathcal{E}$ of recursive trees $\mathcal{T}$ of kernels k, shown on the **right**. The ensemble produces the same result and it can be convolved more efficiently than with the classic approach of Viola and Jones.*

Haar-like features are ubiquitous in computer vision. Their popularity commenced to soar with the successful work of Viola and Jones on a real-time and accurate face detector [Viola and Jones, 2001]. Demand for fast trackers has urged to adopt the features, as they can be computed rapidly using integral image [Crow, 1984]. They have been widely integrated into applications for tracking vehicles [Chateau et al., 2002], faces [Cristinacce and Cootes, 2006], body limbs [Ramanan et al., 2007], eye-pupils [Swirski et al., 2012], or generic objects while learning their appearance

on-line [Saffari et al., 2009], [Babenko et al., 2009], [Stalder et al., 2009], [Zhang et al., 2012], to list only a few examples.

§ 5.1 Motivation

The observable phenomenon of ageing European society creates new social demands. The ultimate goal of CompanionAble, a project funded by the European Union, was the development of an artificially intelligent system for helping elderly people live safely and independently in their homes. Towards this end, one of its objectives was the development of a human tracking system for localizing people indoors that could further facilitate fall detection of elderly.

In effect, the CompanionAble project laid ground for this thesis. A human body detector of [Ramanan et al., 2007] was implemented into the autonomous surveillance system. The detector used two edge Haar-like features to localize human limbs for automatic initialization. Similarly to [Ramanan et al., 2007], we observed that the human torso was the most salient and thus the easiest body part to detect. We present exemplary results of automatic torso detection in Fig. 5.2.

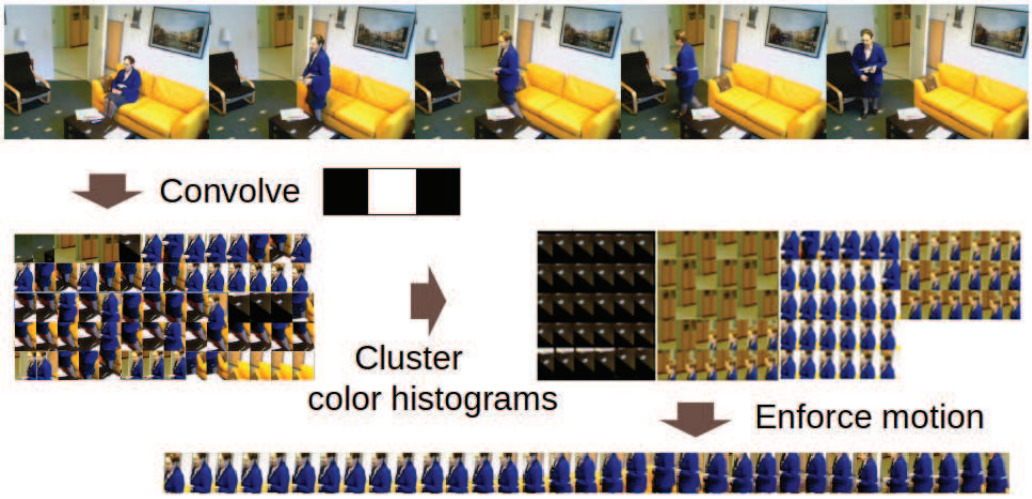


Figure 5.2: *Automatic torso detection as part of the framework for full body appearance learning and tracking, following* [Ramanan et al., 2007]. *A batch of images is first convolved with a torso template in the form of Haar-like features that approximate two parallel edges. The detected torso candidates are clustered in the color histogram feature space to enforce coherent appearance and then assembled within clusters with motion constraints to yield a valid track.*

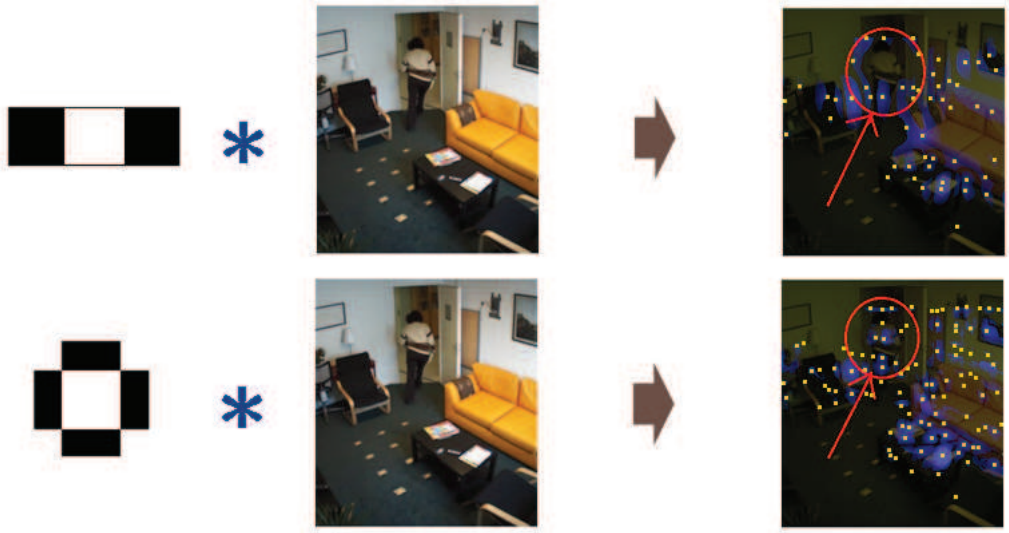


Figure 5.3: *The motivation behind the development of a parsing algorithm for Haar-like features is the observation that augmenting the set of features can improve detection. In the top, an examplary image is convolved with the two vertical edge Haar-like features, as in* [Ramanan et al., 2007]*, that miss the human torso due to low contrast with the background. On the other hand, an augmented set of four edge Haar-like features in the bottom successfully fires on the torso mainly owing to two horizontal edges. The maximal detections are obtained after non-maximum suppression and marked yellow.*

However, we found that the template formed of two side edges was too coarse to score accurately the human torso and other body parts. It was often confused with background clutter or yielded imprecise candidates after non-maximum suppression. When we simply augmented the Haar-like features into an ensemble of two vertical and two horizontal edges, we obtained more reliable detections, as shown in Fig. 5.3. Consequently, we built even more complex templates of human upper body parts that were formed of Haar-like features, as depicted in Fig. 5.4, and integrated them into our model from chapter 3 to tune it for automatic upper body detection.

Arguably, Haar-like features can be computed rapidly with integral image. But, since we used templates with rich configurations of features, such augmented sets inevitably implied increased computational cost. Fortunately, since Haar-like features are composed of repeated weighted boxes, they could be convolved recursively. Yet striving to expand the convolution for arbitrary Haar-like features, we elaborated a new, general parsing formalism, which creates recursive convolutions of trees of kernels from ensembles of the features. The parser is explained in the next section.

Figure 5.4: *Examples of finer contour templates, each composed of a configuration of an ensemble of Haar-like features. We integrate such shape templates for detection of e.g. human upper body parts. We first convolve snippets of images with each of the three templates of human head and shoulders, torso, and hips across several scales. The detections are then assembled spatially to draw N-best* [Nilsson and Goldberger, 2001] *upper body configurations under the deformable parts model, proposed in chapter 3. Finally, we enforce motion constraints on each hypothesized configuration to infer the best track. Note we require upright people to initialize detection and thus configure the model with vertical, global orientation. The sequence comes from CAT dataset* [Micusk, 2011].

§ 5.2 Parsing Haar-like features

The problem of efficiently convolving an image with Haar-like features is addressed in this section. The features are classically computed in one pass over integral image by reading the values at the feature corners. Here we present the parsing formalism for convolving them more efficiently. Our method is fully automatic and applicable to an arbitrary set of Haar-like features. The parser reduces the number of memory accesses which are the main computational bottleneck during convolution on modern computer architectures. It first splits the features into simpler kernels. Then it aligns and reuses them where applicable forming an ensemble of recursive convolution trees, which can be computed faster. This is illustrated with experiments, which show a significant speed-up over the classic approach.

The following sections are organized as follows. First, previous and related work is presented in section 5.2.1. Then, we formulate our problem in section 5.2.2. In section 5.2.3 we describe an algorithm for parsing Haar-like features into recursive trees of kernels, as shown in Fig. 5.1, to reduce the number of memory accesses. We

achieve this by decomposing the features into smaller kernels and by aligning them. An efficient implementation for computing the trees is also proposed. Section 5.3 experimentally compares the baseline classical approach of Viola and Jones [Viola and Jones, 2001] with our approach showing time results on CPU. We conclude in section 5.4.

5.2.1 Background

The following general definition of Haar-like features is adopted: *kernels of Haar-like features are matrices of coefficients obtained after double differentiating piecewise flat patterns.*

In 1984, Crow [Crow, 1984] introduced summed-area table (SAT) to the computer graphics community as a generalized method for mip-mapping. In [Heckbert, 1986], Heckbert used SAT for efficient convolution by repeatedly integrating differentiated box filters in 1D and 2D domains. This was formulated in mathematical terms as $f * k = f^{[n]} * k^{[-n]}$, where $f^{[n]}$ and $k^{[-n]}$ mean n-fold integration and n-fold differentiation of image f and kernel k, respectively.

Unlike [Heckbert, 1986], where the kernels k were quantized, Simard [Simard et al., 1998] quantized images f forming boxlets. They were then also differentiated to produce trains of impulses along image axes. The coinciding impulses of neighbouring boxlets often cancelled out leading to a reduced representation of the image. Hence, it could be convolved with an arbitrary kernel more rapidly. The algorithm was formulated as $f * k = (f^{[-n]} * k^{[-m]})^{[n+m]}$.

We note this is a general, efficient scheme, which boosts speed performance of convolution primarily by introducing a reduced representation $f^{[-2]}$ of the image f. In their paper, the authors actually do not differentiate the kernel k. This would require recursively integrating the response four times, instead of two, as implied by $f * k = (f^{[-2]} * k^{[-2]})^{[2+2]}$. However, should this be of no concern for any reason, the boxlets scheme could also be applied to the kernel. If the kernel consisted of e.g. an ensemble of Haar-like features, the scheme would create their second-order derivatives $k^{[-2]}$, which this chapter considers. Hence, our scheme takes boxlets one step further. Namely, it transforms the boxletization $k^{[-2]}$ of the kernel k to an

ensemble of recursive convolution trees of simpler kernels which eventually require less data to produce exactly the same result as the original kernel $k^{[-2]}$. Therefore, our trees can be well applied together with the boxletization $f^{[-2]}$ of the image f, keeping in mind that the final result would then require recursive integration of the response four times.

Later, Viola and Jones [Viola and Jones, 2001] rephrased SAT to integral image and showed how to compute several Haar-like features from it by summing weighted boxes. The attractivity of the integral image comes from the fact that the sum of pixels under e.g. a rectangular area can be computed in constant time at any scale and location by reading the values at four corners of the rectangle. This approach is very simple and thus has become very popular. For example, it has been implemented in the OpenCV library in the framework for rapid object detection [Lienhart and Maydt, 2002].

Due to the overall simplicity of this approach, little work has been done on efficient computation of Haar-like features. Since all features are composed of boxes, some approaches consist in first computing boxes and combining them to obtain more complex features [Ouyang et al., 2010], [Terriberry et al., 2008], [Lienhart and Maydt, 2002]. In [Ouyang et al., 2010], the authors concentrated on reducing the number of arithmetic operations by introducing a strip sum data structure. However, it needs to be emphasized that nowadays the computation time is bounded primarily by memory accesses [Drepper, 2007]. In view of this, we focus on reducing input/output data transfer without regard to the number of required arithmetic operations and we show that this strategy leads to a faster algorithm. Previously, in [Terriberry et al., 2008] the authors also considered reducing the number of memory accesses for Haar-like features, though with regard to GPU architecture. While both methods improve on the naive classic computation [Viola and Jones, 2001], they refer only to specific features and are embedded into other algorithms, whereas we give theoretical explanations behind practice and generalize to any Haar-like features.

In this chapter we propose a novel algorithm for convolving an image with any set and type of Haar-like features by reducing memory accesses *jointly* for all features. This is challenging when multiple features are considered and no assumption is made

on their mutual positions. In order to reduce memory accesses during convolution, our idea is to decompose the set of features into smaller kernels, thus forming multi-pass convolutions, and align the kernels within and between passes. This scheme results in an ensemble of recursive convolution trees which reuse previously computed responses of smaller kernels, possibly shared by some subset of features.

5.2.2 Problem formulation

In this section we present our model for reducing inputs and outputs of multiple features during pixel-wise convolution. An input I is a memory read operation, whereas an output O is a memory write operation. We first introduce the parameters and degrees of freedom of the problem. Then, we describe the actions which can be performed on kernels in order to reduce the total sum of inputs and outputs of the convolution.

5.2.2.1 Model

Let $\mathcal{K} = \{k_i\}_{i=1}^{N}$ be a set of N convolution kernels of Haar-like features and W be a 2D scanning window with its own coordinate system and origin in the upper left corner, as we employ left-to-right topology on the memory layout. We require all kernels $k_i \in \mathcal{K}$ to be computed explicitly within W. For brevity, we restrict ourselves to case $\text{rank}(k_i) = 1$, but extending it to kernels of higher rank is straightforward. Kernel k_i is a one- or two-dimensional vector or matrix, respectively, parameterized by:

- *input positions*, indicated by non-zero coefficients;
- *size*, determined uniquely by the layout of non-zero coefficients;
- *offset position* $x_i \in \mathbb{N}^2$ from the origin of W, which we write as $k_i(x_i)$.

In the following, apart from $\text{rank}(k_i) = 1$, we have no assumptions concerning the cardinality of $\mathcal{K}$, the number of inputs of k_i, its size nor position in W. We thus say that $\mathcal{K}$ has arbitrary configuration. Consequently, we assign two relation properties to $\mathcal{K}$. A pair of kernels k_i and k_j, where $i, j \leq N$, is:

- *equivalent* if $k_i = \alpha k_j$, where $\alpha \in \mathbb{R}$;
- *n-coinciding* if their n non-zero coefficients are at the same positions in W,

where $n = 0, \ldots, \min(\eta(k_i), \eta(k_j))$ and $\eta(k_i)$ is an operator returning the number of non-zero coefficients of kernel k_i. For instance, window W can contain two equivalent kernels $k_i(x_i)$ and $k_i(x_j)$, which are located at different positions in W, such that $x_i < x_j$. Also, none of their non-zero coefficients may coincide. We then call such kernels 0-coinciding.

Let us now briefly review how to implement a multi-pass convolution scheme, obtained after decomposing a single kernel into P smaller kernels. A small kernel, which is assigned to the first pass, is convolved with the whole image. Then, the second small kernel is convolved with the response obtained after the convolution with the previous kernel. The process is repeated recursively with the remaining $P - 2$ smaller kernels to produce the final response, which is equivalent to the convolution with the original kernel. One can observe this P-pass convolution has in total P outputs, whereas the single-pass convolution has 1 output. The last output, which writes the final result into memory, is unavoidable for both convolutions. Since we aim at comparing the efficiency of convolution schemes based on the total number of memory accesses required to compute the final result, the count of the final output is ignored in our I/O analysis.

Within the context of memory accesses, which have the cost of several orders of magnitude higher than arithmetic operations on modern CPU architectures, a good strategy for realizing the above multi-pass convolution is to turn it into a buffered recursive single pass. That is, it becomes the single pass convolution over the image, but a multi-pass convolution over the buffer. Namely, in each iteration of the convolution (i.e. at each pixel), a result obtained by one kernel is stored in a temporary variable (ALU register) in order to be used twice in the same iteration between two consecutive passes: 1) it is taken as the last input of the next kernel, and 2) it is output to the buffer so that it can be input to the next kernel in subsequent iteration(s). Therefore, this multi-pass convolution over the buffer reduces the number of inputs of the multi-pass convolution over the image by $P - 1$ because it combines

the output of the kernel in one pass with the input of the kernel in the next pass in the buffer $P-1$ times. We note that accessing registers has negligible cost.

We formalize the above discussion by introducing three actions which can be performed on a kernel k_i in order to reduce total I/O count of $\mathcal{K}$ while keeping the final results unchanged, namely decomposition, permutation, and alignment. Each action implies cost of c_I inputs and cost of c_O outputs, where $c_I, c_O \in \mathbb{Z}$. The total I/O cost is $c_{I/O} = c_I + c_O$. If $c_{I/O} < 0$, then the actions reduce the total sum of inputs and outputs w.r.t. the initial configuration of $\mathcal{K}$.

Decomposition of kernel k_i into P_i smaller kernels $k_i = k_i^1 * \ldots * k_i^p \ldots * k_i^{P_i}$ generates a recursive, multi-pass convolution over the buffer, where p indicates the p-th pass (i.e. k_i^1 is convolved with the image as first). The input cost of this action is $c_I = \sum_{p=1}^{P_i} \eta(k_i^p) - (P_i - 1) - \eta(k_i)$, whereas the output cost is $c_O = P_i - 1$. When $P_i > 1$, this action creates 1-coinciding kernels for $p = 1, P_i$ and 2-coinciding kernels for the remaining passes.

Permutation of smaller kernels, e.g. $k_i^1 * \ldots * k_i^{P_i} = k_i^{P_i} * \ldots * k_i^1$, is performed to assign them to specific passes thus changing their positions in W. It has costs $c_I = 0$ and $c_O = 0$. It can lead to reduction of inputs only when combined with alignment with other kernels in the case of multiple features. Yet, if i equivalent kernels $(1 < i \leq N)$, unfolded across multiple features after their particular decompositions, are permuted to the same pass and preceded by equivalent kernels in previous passes, then a joint recursion is continued leading to the output cost $c_O = 1 - i$ for this pass.

Alignment of two kernels k_i and k_j is an action which shifts k_i *rightwards* (due to the left-to-right topology) until it coincides with k_j in at least one position. It is analyzed in two possible cases.

1° Single feature. In this case, one can only align last input of kernel k_i^p with the first input of kernel k_i^{p-1} to allow recursion. Aligning a kernel k_i^p with k_i^{p-1} reduces the number of k_i^p inputs by 1. However, this implies that k_i^p is detached from k_i^{p+1}, what increases the number of inputs of k_i^{p+1} by 1. Therefore, this action does not introduce either loss or gain in the number of inputs, and hence it has cost $c_I = 0$. The output cost is $c_O = 0$ as it is not possible to reduce the number of outputs

by aligning k_i^p with k_i^{p-1}. There are two special cases. Firstly, shifting k_i^1 has cost $c_I = 1$, as it is not preceded by other kernel. Secondly, since we need to know the value of the feature at specific, predefined location in W, aligning $k_i^{P_i}$ with already shifted $k_i^{P_i-1}$ requires to assign an output to $k_i^{P_i}$ which will be read with single input by additional delta impulse $\delta(x_i)$, placed at the original location of k_i. Hence, if the last kernel is moved, the I/O count increases as $c_I = 1$ and $c_O = 1$. Clearly, shifting a kernel from any pass may reduce total I/O count only in the case of multiple features.

2° Multiple features. This case is more involved for several reasons. Again, we note that it is possible to align kernels of different features only if they are in the same pass and are preceded by equivalent kernels in previous passes to yield recursion. Now, a kernel in a given pass can be *initially* n-coinciding, i.e. before being aligned with another kernel located elsewhere in W. Therefore, if it is aligned with another kernel becoming m-coinciding, the input cost is $c_I = n - m$. This implies that it is possible that aligning kernels may not lead to an improvement, even for equivalent kernels. The output cost for aligning kernels of different features is $c_O = 0$. Moreover, even in the simple case, where all kernels have 2 inputs, aligning them is an NP-complete problem. We motivate our argument by constructing a simple example. Let 4 kernels with 2 inputs have the following sizes: 1, 2, 4, 7. Clearly, there is only one best alignment as $1+2+4 = 7$, which would result in total of 4 inputs. However, this generally requires to compute all possible sums of integers to decide which subset sum equals to another subset sum. Subset sum problem is NP-complete. So, we try all possible alignments to decide which one yields minimum number of inputs and outputs.

5.2.2.2 Example

As a simple example supporting our discussion, we consider two kernels of Haar-like features. They are defined in a window $W^{8\times 8}$ as:

- $k_1(1,0) = \begin{bmatrix} +1 & 0 & 0 & 0 & 0 & -1 \\ -3 & 0 & 0 & 0 & 0 & +3 \\ +3 & 0 & 0 & 0 & 0 & -3 \\ -1 & 0 & 0 & 0 & 0 & +1 \end{bmatrix}$,
- $k_2(0,5) = \begin{bmatrix} -1 & 0 & 0 & +1 & 0 & +1 & 0 & 0 & -1 \\ +2 & 0 & 0 & -2 & 0 & -2 & 0 & 0 & +2 \\ -1 & 0 & 0 & +1 & 0 & +1 & 0 & 0 & -1 \end{bmatrix}$.

Initially, their positions are 0-coinciding in W, as depicted in Fig. 5.5(a). In classical computation, the number of inputs corresponds to the number of non-zero coefficients in the matrices, what here amounts to 20 inputs. After decomposing the features into simpler kernels, we have:

- $k_1 = k_1^x(1,3) * k_1^y(1,1) * k_2^y(1,0) = \left[\,+1\;0\;0\;0\;0\;-1\,\right] * \left[\,+1\;-2\;+1\,\right]^{\mathrm{T}} * \left[\,+1\;-1\,\right]^{\mathrm{T}}$,
- $k_2 = k_1^x(3,7) * k_2^x(0,7) * k_1^y(0,5) = \left[\,+1\;0\;0\;0\;0\;-1\,\right] * \left[\,-1\;0\;0\;+1\,\right] * \left[\,+1\;-2\;+1\,\right]^{\mathrm{T}}$,

which amounts to 10 inputs and 4 outputs (Fig. 5.5(c)). Now, we observe that the features share two equivalent kernels, i.e. one vertical k_1^y, which is explicit (directly indicates side size of two boxes in both features) and one horizontal k_1^x, which is implicit (does not directly indicate side size of any box(es) in k_2). By permutation, we assign k_1^x to the first pass, k_1^y to the second pass, and the remaining kernels k_2^x and k_2^y to the third, last pass. Finally, we align the kernels within passes. The kernel $k_1^x(1,3)$ is shifted rightwards to coincide fully with its equivalent kernel $k_1^x(3,7)$ in the first pass. The same is repeated for the next equivalent kernel $k_1^y(1,1)$ in the second pass. Since the remaining kernels are in the last pass and are not equivalent, it is not efficient to align them as it would increase the current total I/O count. Hence, the kernel k_2^y remains at its original position. Concluding, these three passes require 7 inputs and 2 outputs to compute both features exactly (Fig. 5.5(d)). Theoretically predicted improvement translates into 2.1-fold speed-up.

5.2.3 Proposed parsing algorithm

This section describes the algorithm. The input of the algorithm is an arbitrary set of Haar-like features assigned to arbitrary positions within a scanning window W. The output is another signal representation of the features in the form of recursive collection of smaller kernels. The algorithm acts like a parser. It splits a particular set of features into smaller kernels, assigns them into passes, aligns them within passes, and creates joint recursions for features if they share equivalent kernels while counting at each step the number of inputs and outputs. We call such a parsed kernel representation an *ensemble of recursive trees*.

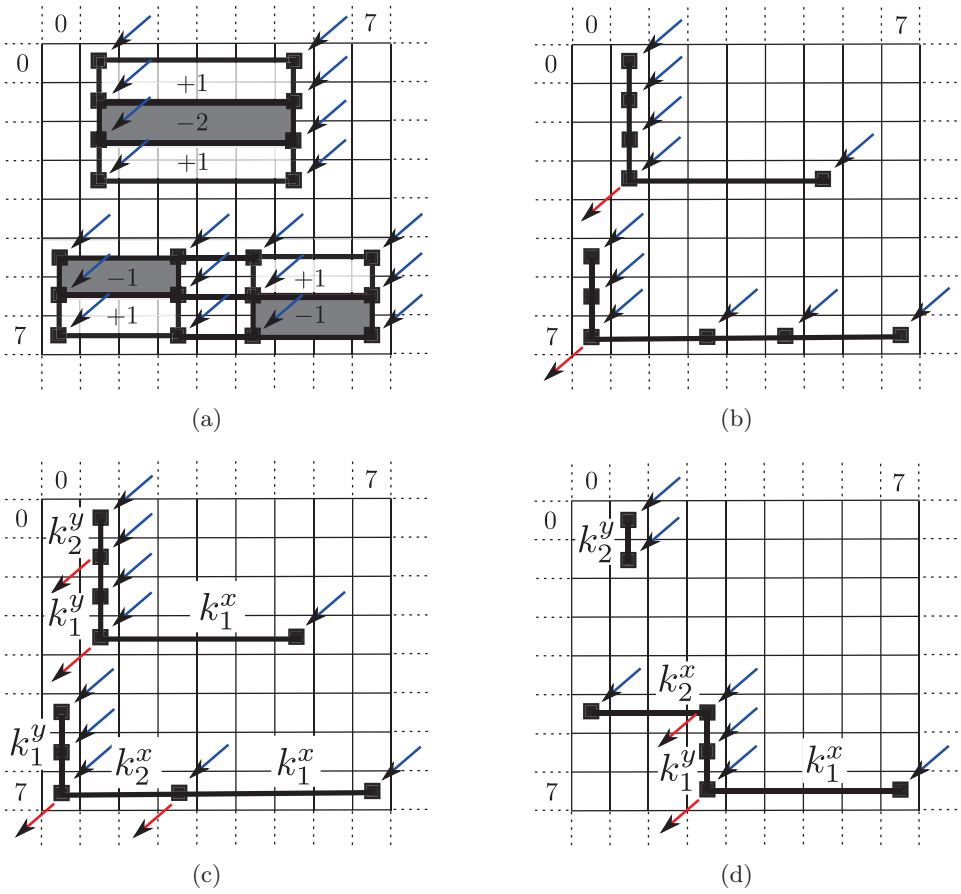


Figure 5.5: *(a,b,c,d) illustrate the steps of decomposition, permutation, and alignment. Each black square indicates non-zero coefficient. The blue arrows incoming to the squares denote inputs from memory, whereas the red outcoming arrows denote outputs to memory. (a) is an example configuration of two features (top - k_1, bottom - k_2), (b) their decomposition from 2D into 1D using SVD, (c) their further 1D decomposition which discovers two equivalent kernels, and (d) a recursive convolution tree of kernels after their assignment to and alignment between and within convolution passes.*

Specifically, the proposed automatic, *off-line* formalism creates recursive convolution trees of decomposed Haar-like features, which require in total less inputs and outputs to produce the same result as original configuration of the features. First, we procedurally describe our method. The idea is to create recursive multi-pass trees of kernels and align them within each pass in such a way that the total sum of inputs and outputs is minimal. This problem is NP-complete. However, since the number of features is typically not large, it is practical to solve it with a brute-force search. This suggests the following approach:

1. *Decompose* features into smaller kernels in all possible ways such that the

number of inputs is reduced.

2. *Assign* kernels of each feature to passes by permuting them.

3. *Align* kernels of all features within and between subsequent passes.

4. *Choose* ensemble of recursive trees with minimal sum of inputs and outputs.

In the last part of this section, a simple yet efficient buffering strategy is proposed for implementing the ensemble by increasing the locality of memory reference during convolution.

5.2.3.1 Decomposing features into smaller kernels

Feature decomposition transforms feature kernel $k_i \in \mathbb{Z}^{Y \times X}$ into a convolution product of vectors $a_{j,t}$, whose first element is equal to 1, last to $j \in \{-1, 1\}$, and the other $(t-2)$ elements are equal to 0, for instance $a_{-1,3} = [\,+1\ 0\ -1\,]$. We call vectors $a_{j,t}$ as *primitive kernels* of size t $(t \geq 2, t \in \mathbb{Z})$. P-decomposition of k_i exists if there exist a_{j_p,t_p} s.t. $k_i = \alpha \left(a_{j_1,t_1} * \ldots * a_{j_p,t_p} * (a_{j_{p+1},t_{p+1}})^T * \ldots * (a_{j_P,t_P})^T\right)$, where $\alpha \in \{-1, 1\}$. The problem is then to find all existing P-decompositions of k_i s.t. $P \leq \eta(k_i)$.

In the following it is assumed that k_i is separable. Thus the SVD $k_i = k_i^x * k_i^y$ exists, where $k_i^x \in \mathbb{Z}^{1 \times X}$ and $k_i^y \in \mathbb{Z}^{Y \times 1}$. For simplicity, further presentation concerns decomposition of a vector. We refer to $k_i^{x,y}$ simply as k.

Let $\phi(k, l)$ be an operator returning set $\mathcal{S}$, containing all $a_{j,t}$ such that $t \leq l$ and $\varphi(k, a_{j,t}) = 1$, where

$$\varphi(k_1, k_2) = \begin{cases} 1 & \text{if } \exists\ k' \text{ s.t. } \psi(k_1, k_2) = k' \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

and ψ denotes deconvolution operator i.e. $\psi(k_1, k_2) = k' \Leftrightarrow k_1 = k_2 * k'$. We denote the cardinality of S with L. Note that P-decomposition of vector k exists if $\max_p(t_p) + \min_p(t_p) > 2\frac{\eta(k)}{P}$, where $\{t_p\}_{p=1}^P$ is the size of p-th primitive kernel forming P-decomposition of k. As P-decomposition can yield multiple primitive kernels, e.g. $a_{j_1,t_1} = a_{j_2,t_2}$, we need function $\theta(k, a_{j,t})$ returning max (m) s.t $\varphi(k, (a_{j,t})^m) = 1$,

where $(a_{j,t})^m = \underbrace{a_{j,t} * \ldots * a_{j,t}}_{m}$. Finally we introduce function:

$$\xi(\mathcal{S}, \mathcal{M}) = (a_{j_1,t_1})^{m_1} * \ldots * (a_{j_L,t_L})^{m_L}, \tag{5.2}$$

where $a_{j_l,t_l} \in \mathcal{S}$ and $\mathcal{M} = \{m_l\}_{l=1}^{L}$ s.t. $m_l = \theta(k, a_{j_l,t_l})$.

We then propose to apply to k the procedure summarized in Algorithm 2, returning $\mathcal{R}_P$ s.t. the convolution product of all P primitive kernels in $\mathcal{R}_P$ gives αk. Using the introduced notation we explain how to create signal s_{full} in each iteration i of Algorithm 2. Firstly, we update $\mathcal{S}^{(i)}$ with $\phi(k^{(i-1)}, \eta(k^{(i-1)}) - p + 1)$. Then we set $\bar{t} = \max(t')$ s.t. $a_{j',t'} \in \mathcal{S}^{(i)}$, and we remove the elements of set $\mathcal{S}^{(i)}$ whose size does not satisfy the condition $t > 2\frac{\eta(k^{(i-1)})}{p} - \bar{t}$. Then the set $\mathcal{M}^{(i)}$ is updated using $m_l^{(i)} = \theta(k^{(i-1)}, a_{j_l,k^{(i-1)}})$. Finally, we calculate s_{full} as $\xi(\mathcal{S}^{(i)}, \mathcal{M}^{(i)})$. One can observe that if $\varphi(s_{full}, k^{(i-1)}) = 0$, then P-decomposition of k does not exist. This simple test limits our search space and allows us to terminate before testing numerous combinations of convolution products of primitive kernels $\in \mathcal{S}^{(i)}$. If the test is successful, we find primitive kernels required to satisfy the condition $\varphi(s'_{full}, k^{(i-1)})$. Since P-decomposition does not exist without these primitive kernels, we know that the resulting set $\mathcal{R}_P$ includes them. If there exists at least one primitive kernel without which the condition is not satisfied, the $k^{(i-1)}$ is simplified and the procedure is repeated iteratively. If not, we are forced to test equality of $k^{(i-1)}$ with all possible combinations of convolution product of p primitive kernels belonging to a set of elements $a_{j_l,k_l} \in \mathcal{S}^{(i)}$ occurring exactly $m_l^{(i)} \in \mathcal{M}^{(i)}$ times. In practice, usually the combinations are tested when L is already small.

The procedure given in Algorithm 2 is executed for $P = 1, \ldots, \eta(k)$ resulting with the set $\mathcal{D}$ of all possible $\mathcal{R}_P$, i.e. all possible P-decompositions of k, which is used as input to the procedure described next.

5.2.3.2 Ensembles of trees

Here we describe two steps of our parsing procedure, namely assignment of kernels to passes combined with their alignment within and between passes for N Haar-like features. Let $\mathcal{D}_{k_i}$ be the set of all possible P-decompositions of k_i. We augment $\mathcal{D}_{k_i}$

Algorithm 2 P-decomposition of k

Set $k^{(0)}$ *to* k, p *to* P, $\mathcal{R}_P \longleftarrow \emptyset$
For $i = 1 \ldots$
  Create signal s_{full} *knowing* $k^{(i-1)}$ *and* p
  If $\left(\varphi(s_{full}, k^{(i-1)}) = 0\right)$
    P-decomposition of k *does not exist;* break
  else
    $m = \theta(s_{full}, k^{(i-1)})$
    $s_{temp} \longleftarrow \psi(s_{full}, (k^{(i-1)})^{m-1})$,
    $L \longleftarrow \left|\mathcal{S}^{(i)}\right|$ and $Q \longleftarrow 1$
    For $l = 1 \ldots L$, where $a_{j_l,t_l} \in \mathcal{S}^{(i)}$
      $s'_{full} \longleftarrow \psi(s_{temp}, a_{j_l,t_l})$
      If $\left(\varphi(s'_{full}, k^{(i-1)}) = 0\right)$
        Add a_{j_l,t_l} *into resulting set* $\mathcal{R}_P$
        Decrease the number of searched primitive kernels $p \longleftarrow p - 1$
        $Q = Q * a_{j_l,t_l}$
    If (Q is equal 1 or p is equal 0)
      Combinatorial update of resulting set $\mathcal{R}_P$
      P-decomposition finished successfully; break
    else
      $k^{(i)} = \psi(k^{(i-1)}, Q)$

by decompositions into kernels (not necessarily primitive kernels) resulting from all unique convolutions of primitive kernels for each element in $\mathcal{D}_{k_i}$, s.t. their number of inputs $\leq \eta(k_i)$.

The problem now is to choose a single element from each $\mathcal{D}_{k_i}$, where $1 \leq i \leq N$, such that after their particular alignment the number of memory accesses is minimal, thus creating the best ensemble of recursive trees of kernels. It is not necessary to permute kernels in each element of $\mathcal{D}_{k_i}$ in all possible ways to enumerate all combinations of alignments. This would be straightforward but inefficient. Kernels of some subset of N features can be aligned within given pass only if *all* kernels of this subset are equivalent in the preceding passes, so allowing a recursion. We call this a *proper assignment*. Otherwise, their alignment is not possible reducing the problem to multiple cases of single features. Of course, all unique kernels from all features can be aligned within the first pass of the convolution, but the alignments in the next passes depend on the above condition. The pseudo-code is in Algorithm 3.

Algorithm 3 N-features assignment and alignment

Define ensemble $\mathcal{E}_1 \longleftarrow \emptyset$ *and label it open*
Define set of ensembles $\mathcal{S}_\mathcal{E} \longleftarrow \mathcal{E}_1$
Set merged trees $\left(\mathcal{T}^0_{k_i} \longleftarrow \mathcal{D}_{k_i}\right)_{1 \le i \le N}$
Label each first kernel $\in \mathcal{T}^0_{k_i}$ *with* $\mathcal{E}_1$
For $p = 1 \ldots$
  For $i = 1 \ldots N$
    For each tree $\mathcal{T}^{p-1}_{k_i}$
      For each branch of tree $\mathcal{T}^p_{k_i}$ from *leaf*
      up to kernel with any $\mathcal{E}$-label in pass p
        Set n *to number of unique kernels in branch*
        Replicate branch $(n-1)$ *times*
        Add new branches to $\mathcal{T}^{p-1}_{k_i}$
        Permute unique kernels of all branches to p
      Merge all branches into single kernel in pass p
        which have equivalent kernels in pass p
  For each $\mathcal{E}_j \in \mathcal{S}_\mathcal{E}$ labeled as open
    For each combination t of $\mathcal{E}_j$-labeled kernels
     of different features in pass p
      Create all possible alignments of kernels
      Set n *to number of alignments*
      Replicate $\mathcal{E}_j$ $(n-1)$ *times. Create set* $\mathcal{A}$ *with* $\mathcal{E}_j$
      Add to each $\mathcal{E}_j \in \mathcal{A}$ *kernels in* t
        with particular alignment
      Update current cost I/O *for each* $\mathcal{E}_j \in \mathcal{A}$
      $\mathcal{S}_\mathcal{E} \longleftarrow \mathcal{S}_\mathcal{E} \cup \mathcal{A}$
  If (Exist $\mathcal{E}_j \in \mathcal{S}_\mathcal{E}$ not having proper assignment)
    For each $\mathcal{E}_j$ not having proper assignment
      For kernels of $\mathcal{E}_j$ *in pass* p *create all*
       combinations of branches from leaf to pass p
      Set n *to number of combinations*
      Label $\mathcal{E}_j$ *as closed*
      Replicate $\mathcal{E}_j$ $(n-1)$ *times. Create set* $\mathcal{A}$ *with* $\mathcal{E}_j$
      Add a combination of branches to each $\mathcal{E}_j \in \mathcal{A}$
      Compute total cost I/O *for each* $\mathcal{E}_j \in \mathcal{A}$
      $\mathcal{S}_\mathcal{E} \longleftarrow \mathcal{S}_\mathcal{E} \cup \mathcal{A}$
  If (Exist $\mathcal{E}_j \in \mathcal{S}_\mathcal{E}$ having proper assignment)
    For each $\mathcal{E}_j$ having proper assignment
      Label children nodes of $\mathcal{E}_j$ *kernels in pass* p *as* $\mathcal{E}_j$
  else
    break

5.2.3.3 Choosing the best ensemble

After creating all possible unique ensembles of recursive trees $\mathcal{E}_j$, the one is chosen from $\mathcal{S}_\mathcal{E}$ which yields minimal sum of inputs and outputs. However, it is possible

that there will be multiple such ensembles. In this special case, we first choose a configuration which has minimal total number of inputs as they are more sparsely referenced than outputs (see section 5.2.3.4 for details). If there are still multiple equal ensembles, we prefer ones with minimal number of inputs in the first pass, then which are more local in this pass, and finally which form buffer of smallest height. If there are any ensembles left after these heuristics, we propose to choose an arbitrary one. We emphasize that these detailed rules are seldom necessary though.

5.2.3.4 Implementation: B-channel buffer

The parsed, best ensemble of recursive trees requires multiple outputs, say B, at each iteration of the pixel-wise convolution with the image. Its straightforward implementation would consist of individual circular buffers storing individual outputs, which would be then reused as inputs in subsequent iterations. Buffer sizes would differ from kernel to kernel. For example, a buffer for a vertical kernel would have height of this kernel and width equal to the image width. On the other hand, a buffer for a horizontal kernel would only have height of one row and width equal to the width of the kernel. This buffer clearly would occupy less memory than the former. However, since each one would reserve different memory block, such a buffering solution would result in non-local memory reference and thus be cache-unfriendly.

In view of this, we propose to reserve a single contiguous memory block for a buffer which, at each iteration, stores all B outputs of the ensemble of recursive trees in one B-element contiguous data array, similarly to RGB image data structure - hence the appellation B-channel buffer. The inputs defined by kernels parsed into the first pass are read from the image, while the inputs of kernels from the remaining passes are read from the buffer. The input locations are specified by the kernels' positions computed after alignment. Additional predefined offset to particular channel is required for inputs in the buffer. Hence, the width of the buffer equals the image width, while its height depends on a particular alignment of kernels within passes. Indeed, such a buffer occupies more memory than actually required but, in the context of convolution, this is not prohibitive on modern CPU architectures, which suffer from limited memory bandwidth (memory wall) and not

from limited memory space. Consequently, such a buffering approach increases locality of memory reference, thus making it a cache-more-friendly-strategy.

§ 5.3 Experiments

In this section, we present the performance of ensembles of recursive trees of kernels parsed by our algorithm. We illustrate their behavior on two examples having practical importance in computer vision. The results are evaluated in terms of time efficiency by comparing the proposed convolution method with the classical approach of Viola and Jones [Viola and Jones, 2001]. We use an integral image of size 4096×4096 in all experiments, though all the results are repeatable for other image sizes. The improvement between both methods is predicted theoretically as:

$$\text{Predicted Improvement} = \frac{I_{class} + 1}{I_{prop} + O_{prop} + 1} \tag{5.3}$$

where I_{class} refers to the total number of inputs of the classical method, and I_{prop} and O_{prop} refer to the total number of inputs and outputs of the proposed method, counted per pixel. Classical method requires only inputs to compute all Haar-like features, whereas both methods require an additional output to store the final result. The performance tests were run on 2 Ghz Pentium 4 processor which was connected to 3.5 GB RAM unit through 32-bit data bus with the support of 4 MB cache. The code was compiled with VC++ compiler under Windows environment.

Example SURF: First example illustrates behavior of our algorithm on Haar-like features approximating Hessian of Gaussians in SURF [Bay et al., 2006]. They are specified by an offset from the origin of the scanning window defined in image coordinate system. Their kernels have the following non-zero coefficients using standard matrix notation:

1. $k_1(2,0) \in \mathbb{Z}^{10\times 6}$, where: $k_{1,1} = +1$, $k_{1,6} = -1$, $k_{4,1} = -3$, $k_{4,6} = +3$, $k_{7,1} = +3$, $k_{7,6} = -3$, $k_{10,1} = -1$, $k_{10,6} = +1$;

2. $k_2(1,1) \in \mathbb{Z}^{8\times 8}$, where: $k_{1,1} = +1$, $k_{1,4} = -1$, $k_{1,5} = -1$, $k_{1,8} = +1$, $k_{4,1} = -1$, $k_{4,4} = +1$, $k_{4,5} = +1$, $k_{4,8} = -1$, $k_{5,1} = -1$, $k_{5,4} = +1$, $k_{5,5} = +1$, $k_{5,8} = -1$,

$k_{8,1} = +1$, $k_{8,4} = -1$, $k_{8,5} = -1$, $k_{8,8} = +1$;

3. $k_3(0,2) \in \mathbb{Z}^{6\times 10}$, where: $k_{1,1} = +1$, $k_{1,4} = -3$, $k_{1,7} = +3$, $k_{1,10} = -1$, $k_{6,1} = -1$, $k_{6,4} = +3$, $k_{6,7} = -3$, $k_{6,10} = +1$.

The proposed approach parses the SURF features jointly, producing recursive trees illustrated in Fig. 5.6. Their kernels are listed in Table 5.1, where $0_{[n]}$ denotes a zero vector of size n. One can observe that the number of memory accesses is reduced from 32 to 19, which results in the theoretical time improvement of 1.65. The measured improvement is 1.63, hence confirming the theory.

$k_1^x(2,9)$	$[+1\ 0_{[4]}\ -1]$
$k_1^y(2,0)$	$[+1\ 0_{[2]}\ -3\ 0_{[2]}\ +3\ 0_{[2]}\ -1]^{\mathrm{T}}$
$k_2^x(2,9)$	$[+1\ 0_{[2]}\ -1]$
$k_2^y(5,5)$	$[+1\ 0_{[2]}\ -1]^{\mathrm{T}}$
$k_3^x(1,5)$	$[+1\ 0_{[3]}\ -1]$
$k_4^y(1,1)$	$[+1\ 0_{[3]}\ -1]^{\mathrm{T}}$
$k_3^y(6,2)$	$[+1\ 0_{[4]}\ -1]^{\mathrm{T}}$
$k_4^x(0,2)$	$[+1\ 0_{[2]}\ -2\ 0_{[2]}\ +1]$

Table 5.1: *Kernels in recursive trees for SURF example.*

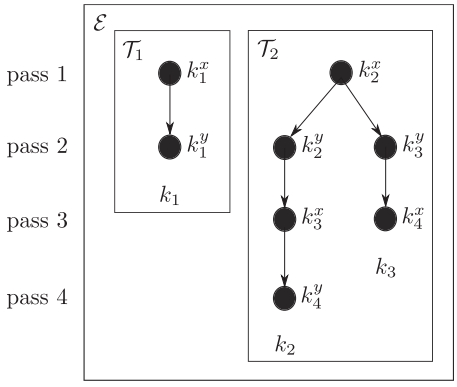


Figure 5.6: *SURF features parsed into ensemble $\mathcal{E}$ of recursive trees. Tree $\mathcal{T}_1$ represents feature k_1, while $\mathcal{T}_2$ represents features k_2 and k_3. The nodes illustrate the decomposed kernels. The directed edges indicate the order of kernels in multi-pass convolution. Each branch of the trees (from root to leaf) corresponds to one feature. For instance, feature k_2 is parsed as: $k_2 = k_2^x * k_2^y * k_3^x * k_4^y$.*

$k_1^y(1,12)$	$\left[+1\ 0_{[3]}\ -1\right]^{\mathrm{T}}$
$k_2^x(1,2)$	$\left[-1\ 0_{[5]}\ +3\ 0_{[5]}\ -3\ 0_{[5]}\ +1\right]$
$k_1^x(1,16)$	$\left[+1\ 0_{[14]}\ -1\right]$
$k_3^y(1,7)$	$\left[-1\ 0_{[2]}\ +3\ 0_{[2]}\ -3\ 0_{[2]}\ +1\right]^{\mathrm{T}}$
$k_2^y(1,12)$	$\left[-1\ 0_{[1]}\ +2\ 0_{[1]}\ -1\right]^{\mathrm{T}}$
$k_3^x(3,7)$	$\left[+1\ 0_{[13]}\ -1\right]$

Table 5.2: *Kernels in recursive trees for FACE example.*

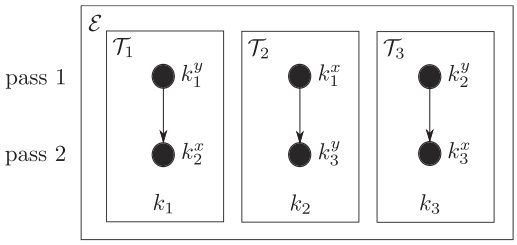


Figure 5.7: *FACE features parsed into trees. Trees $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ represent features k_1, k_2, k_3, respectively.*

Example FACE: Similarly, the Haar-like features in the first stage of the face detection cascade [Lienhart and Maydt, 2002] are defined as:

1. $k_1(1,2) \in \mathbb{Z}^{5\times 19}$, where: $k_{1,1} = -1$, $k_{1,7} = +3$, $k_{1,13} = -3$, $k_{1,19} = +1$, $k_{5,1} = +1$, $k_{5,7} = -3$, $k_{5,13} = +3$, $k_{5,19} = -1$;

2. $k_2(1,7) \in \mathbb{Z}^{10\times 16}$, where: $k_{1,1} = -1$, $k_{1,16} = +1$, $k_{4,1} = +3$, $k_{4,16} = -3$, $k_{7,1} = -3$, $k_{7,16} = +3$, $k_{10,1} = +1$, $k_{10,16} = -1$;

3. $k_3(3,7) \in \mathbb{Z}^{5\times 15}$, where: $k_{1,1} = -1$, $k_{1,15} = +1$, $k_{3,1} = +2$, $k_{3,15} = -2$, $k_{5,1} = -1$, $k_{5,15} = +1$,

which are parsed into recursive trees shown in Fig. 5.7, with kernels given in Table 5.2. The resulting representation reduces the number of inputs and outputs from 22 to 16, which translates into 1.35-fold speed-up. The measured time improvement is 1.36.

The presented experiments are summarized in Table 5.3. As expected, the measured time improvement is proportional to the ratio between the sum of inputs of the classical approach and the reduced sum of inputs and outputs of our approach. The

Example	Approach	I/O	t[ms]	Improvement	
				Predicted	Measured
SURF	Classical	32/0	663.70	1.65	1.63
	Proposed	14/5	407.54		
FACE	Classical	22/0	408.60	1.35	1.36
	Proposed	13/3	300.47		

Table 5.3: *Comparison of classical and our approach in terms of time efficiency for SURF and FACE examples.*

results clearly indicate that the ensembles of recursive trees, parsed for the above examples using our method, are computed more rapidly than with the classical approach. The time improvements differ between both examples as each one contains different configuration of Haar-like features. In general, a particular speed-up depends on a particular configuration of the features to be parsed.

§ 5.4 Conclusions and Discussion

In this chapter, we have presented a strategy for speeding-up convolution with an arbitrary configuration of Haar-like features by parsing them *jointly* into an ensemble of recursive trees of simpler kernels. No approximation of the final result is made. Since the main computational bottleneck of any convolution is memory access, we reduce the number of inputs and outputs jointly for all features. Namely, apart from decomposing the 2D kernels into a horizontal and vertical vector, which is a standard SVD procedure, the parser decomposes them *further* to unfold hidden simpler kernels. This process is controlled under reducing/increasing I/O count criterion in order not to decompose blindly, being neither efficient nor necessary. If the parser discovers cases in which these kernels are shared across the features, it forms a joint recursion tree for them and adds it to the set of other trees as next potential solution. After all potential solutions are formed, the one is chosen which yields minimum total I/O count.

Since we reduce the number of memory accesses jointly for all features inside the scanning window, features of several multiple scales can be parsed jointly as well and convolved with an image in one single pass. This may prove efficient especially for such configurations which share equivalent kernels across features and across scales.

Moreover, our framework can be scaled into 3D Haar-like features as well. The decomposition step of the parser would have to be further developed into the 3D case.

Finally, we emphasize that the recursive trees are not limited to convolutions with integral image representation. This is well justified by integration-differentiation property of convolution published by Heckbert in [Heckbert, 1986]. They can be convolved also with a differentiated image representation, which could be obtained using e.g. boxlets method developed by Simard in [Simard et al., 1998]. The boxlets scheme would create second-order derivative kernels of Haar-like features, which we take as input to our parsing algorithm. Therefore, our method takes boxlets one step further with respect to multiple Haar-like kernels. Consequently, boxlets method cannot replace our scheme, as it stops at the point where our scheme starts. Whether our method can be applied efficiently also to the boxletized image remains an open problem. We leave it as an interesting future work.

It is possible that future object detectors will require thousands of templates to cope with high variability of object categories [Bourdev and Malik, 2009], [Yang and Ramanan, 2011]. It is therefore desirable to provide tools for computing a set of templates jointly in efficient and rapid manner. We presented a parsing scheme which achieves this goal.

- Chapter 6 -

Conclusions

This thesis has presented a configurable tracking system for articulated and flexible objects which estimates object location, scale, orientation, and deformation.

§ 6.1 Contributions of the thesis

The main contributions of this dissertation are summarized below:

- The developed model-based tracking system is applied to track various objects within the tracking-by-detection framework. Since an object is represented by an ensemble of parts, the tracker can recover complex motion of objects. Notably, the system can track string vibrations, heavy thread deformations, facial mimics, and arm articulations. It can also localize simple, rigid objects with state-of-the-art performance at frame rates far exceeding real-time. This is achieved with a novel but simple hierarchical model proposed in chapter 3. The tracker is based on Pictorial Structures and merges spatial and temporal priors into a collaborative scheme of deforming segments of parts. The model can be reconfigured to track various objects, which can vary in shape and motion dramatically, as shown in chapter 4. Having no particular prior on dynamics, the algorithm can track objects that shift, rotate, change scale, vibrate, flex, articulate, and the tracker estimates explicitly the scale and in-plane orientation of the object. Finally, the computation time scales linearly with the number of object parts.

- Haar-like features are used extensively in Computer Vision and, in particular, by some of the state-of-the-art trackers described in the thesis. A unified procedure has been proposed for parsing ensembles of Haar-like features for object detection. The parsing formalism is presented in chapter 5. The method is automatic and applicable to an arbitrary set of Haar-like features for convolving them more efficiently than classical approach of Viola and Jones. The parser reduces the number of memory accesses which are the main computational bottleneck during convolution on modern computer architectures. It first splits the features into simpler kernels. Then it aligns and reuses them where applicable forming an ensemble of recursive convolution trees. The trees yield the same convolution result, which can be obtained faster than by the classical approach.

§ 6.2 Potential of the tracking approach

To summarize the advantages of the tracker, the qualitative and quantitative evaluation from chapter 3 and chapter 4 show that the tracker can:

- be reconfigured to represent elongated and planar objects which are rigid, flexible, and articulated;
- follow fast shift, in-plane rotation, and scale change;
- recover from and be robust against occlusions and mild 3D pose change;
- be robust against local deformations;
- work in the presence of low foreground/background contrast, moving background, low quality of the video, and blur;
- process videos at high frame rate, exceeding 100 `fps` on a consumer PC for objects composed of several parts.

§ 6.3 Limitations of the approach

The experiments conducted in chapter 3 and chapter 4 show the tracker still struggles with:

- illumination change. We use very simple appearance features in the form of color histograms, which are invariant to orientation, scale, and interior deformations of the parts and can be computed rapidly. However, these merits of using the color histograms come at the cost of reduced robustness against heavy illumination change, as shown in the robot arm sequence.

- textureless objects. Likewise, regions with utterly homogeneous texture are not discriminative for the model. In such a case, the model cannot discern between the correct and smaller scale of object parts. Effectively, it shrinks by blindly rescaling the pairwise distance constraints, as in the case of the guitar string.

- foreshortening. Each segment of the object has an assigned 2D coordinate system, whose in-plane orientation is assumed to change slow between frames. If the segment rotates out of the image plane, this results in foreshortening. Hence, the in-plane orientation of the segment can then change fast between consecutive frames and thus render the model unsuitable. This is especially evident in tracking human gestures.

Another limitation of our tracking approach is the manual configuration of the model. The external user has to split the object into segments of parts and determine the chain that links the parts. Optionally, the user has to determine initial orientations of the segments if their true inclination has to be known during tracking. These steps constitute a relatively involved initialization procedure.

§ 6.4 Future work

Future work will firstly address the limitations of the tracking approach. The appearance model could integrate gradient-based features, which respond to image edges. As we wish to track objects that rotate, the learned gradient-based features would have to be updated w.r.t. to the current orientation of the object. This is challenging assuming the tracker might sometimes estimate the orientation erroneously but possible and relevant under the proposed model. Such an improved appearance term could increase tracking stability, as edges are more robust against illumination change than regions modeled by color histograms. Since Haar-like features can be used as coarse edge and bar templates, the presented theoretical insights into their recursive decomposition properties set up an interesting avenue to be explored by tracking in the presence of varying illumination. Assigning chained ensembles of HLFs along object contours and regions might thus improve tracking stability while the computational performance would not be much compromised as complex HLFs could be obtained by recursively convolving basic HLFs.

Should the tracker work well for objects not only with rich but also scarce texture, a segmentation method might effectively address this challenge. Snake models, which can also be formulated as chain graphs, are well suited to track such homogeneous foregrounds. Since Snakes usually use gradients as image features, the contours can be computed more reliably on scarce than on highly textured foregrounds. The problem of merging the snakes with our model as well as elaborating a general online segmentation mechanism are an attractive future work to expand the utility of the tracking approach in this direction.

Building a 3D model of a tracked object is another step to further develop the tracking system. A starting point would be to lift the auxiliary coordinate systems from 2D to 3D space in order to explain the foreshortening of the segments. Thus the spatio-temporal term in the proposed objective function could be augmented by optical flow. Integrating temporal information would yield 3D motion cues, such as change of scale and in- and out-of-plane rotation, that could facilitate updating 3D model structure over time.

Each object changes its appearance in a specific way over time. Temporal change of shape and texture, first acquired through tracking, could further help a computer in recognizing arbitrary objects and their actions. This, in turn, emphasizes the need for generic object tracking. The general objective of tracking objects of various shapes and motions has been here approached with one object model. The model is manually configurable and can be applied to track rigid, flexible, and articulated objects. Complementary to the popular *on-line object appearance update* algorithms, the task of choosing automatically the most appropriate number of parts and articulating segments, which are assembled by a particular graphical structure at a given time instant, can be formulated as an *on-line object configuration update* problem. Update mechanisms that could reconfigure on-line the object model between deformable and rigid structures will also be pursued in future work.

List of Figures

1.1 Applications of visual object tracking. Source images were downloaded from Google Images and YouTube. In clockwise order: Polish Main News TVP1, Avatar movie, American Beauty movie, MIT 3D printed key, CompanionAble EU-project, Minority Report movie . . 2

1.2 Unconstrained deformations of a plastic bag 4

1.3 Fast motion 5

1.4 Occlusions 5

1.5 Constrained deformations 6

1.6 Semi-constrained deformations 6

2.1 Image primitives 12

2.2 Texture deformations 13

2.3 Tracking heavy texture deformations 14

2.4 Second-order derivatives of Haar-like features 15

2.5 Tracking approaches 17

2.6 Illumination change 23

3.1 Hierarchical model employed for tracking 37

3.2 Tracking results for different model configurations 45

3.3 Three object categories 47

3.4 Tracking procedure 50

3.5 Recovery of rotation from point correspondences 51

3.6 Qualitative results on Board sequence 56

3.7 Qualitative results on Box sequence 57

3.8 Qualitative results on Lemming sequence 58
3.9 Qualitative results on Liquor sequence 59
3.10 Transformation of tracker output 61

4.1 Objects of various shapes and motions 63
4.2 Model configuration for eye pupils 66
4.3 Qualitative tracking results on Eye pupils sequence 66
4.4 Model configuration for human upper body 67
4.5 Qualitative tracking results on Skater fall sequence 68
4.6 Model configuration for street vehicle 69
4.7 Qualitative tracking results on Tram sequence 69
4.8 Model configuration for string 71
4.9 Qualitative tracking results on Guitar string sequence with scale update 72
4.10 Qualitative tracking results on Guitar string sequence without scale update 73
4.11 Model configuration for lips 74
4.12 Qualitative tracking results on 'I love you so' sequence 75
4.13 Model configuration for face 76
4.14 Qualitative tracking results on Face sequence 77
4.15 Model configuration for robot arm 78
4.16 Qualitative tracking results on Robot welding sequence 79
4.17 Model configuration for medical thread 80
4.18 Qualitative tracking results on Medical thread sequence 81
4.19 Model configuration for full human upper body 82
4.20 Qualitative tracking results on Gestures 1 and Gestures 2 sequences 84
4.21 Qualitative tracking results on Friends 1 and Friends 2 sequences . . 85

5.1 Parsing Haar-like features 87
5.2 Human Tracking with Haar-like features 88
5.3 Need for Ensembles of Haar-like features (EHLFs) 89
5.4 Examples application of EHLFs 90
5.5 Example of parsed Haar-like features 98

5.6 Parsed SURF Haar-like features 105
5.7 Parsed FACE Haar-like features 106

List of Tables

3.1 Challenges of PROST benchmark 55
3.2 Quantitative tracking results 62

4.1 Listed sequences with their main challenges. Unreferenced sequences were downloaded from YouTube. 65

5.1 Kernels in recursive trees for SURF example. 105
5.2 Kernels in recursive trees for FACE example. 106
5.3 Comparison of classical and our approach in terms of time efficiency for SURF and FACE examples. 107

Bibliography

A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *Computer Vision and Pattern Recognition*, pages 798–805, 2006. 53, 62

A.A. Amini, T.E. Weymouth, and R. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990. 32

P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989. 20

M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition*, 2008. 29, 30, 34, 40

M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition*, pages 1014–1021, 2009. 29, 40

B. Babenko, M.H. Yang, and S. J. Belongie. Visual tracking with online Multiple Instance Learning. In *Computer Vision and Pattern Recognition*, pages 983–990, 2009. 24, 39, 53, 55, 62, 88

B. Babenko, M.H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:1619–1632, 2011. 26

S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004. 19

O. Barnich and M. Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6): 1709–1724, 2011. 16

H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, pages 404–417, 2006. 14, 15, 104

S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *International Conference on Intelligent Robots and Systems*, pages 943–948, 2004. 19

J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, pages 237–252, 1992. 19, 20

R. Berger, S. Dubuisson, and C. Gonzales. Fast multiple histogram computation using Kruskal's algorithm. In *International Conference on Image Processing*, 2012. 14

J.Y. Bouguet. Pyramidal implementation of the lucas-kanade feature tracker description of the algorithm, 2000. 21

L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *International Conference on Computer Vision*, pages 1365–1372, 2009. 108

Y. Boykov and D.P. Huttenlocher. Adaptive bayesian recognition in tracking rigid objects. In *Computer Vision and Pattern Recognition*, 2000. 21

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. 24

M.A. Brubaker, D.J. Fleet, and A. Hertzmann. Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision*, 87(1-2): 140–155, 2010. 30

P.J. Burt. Fast filter transform for image processing. *Computer Graphics and Image Processing*, 16(1):20–51, 1981. 15

J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, 1986. 15

L. Cehovin, M. Kristan, and A. Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *International Conference on Computer Vision*, pages 1363–1370, 2011. 27, 39, 40

C. Chang and R. Ansari. Kernel particle filter: iterative sampling for efficient visual tracking. In *International Conference on Image Processing*, pages 977–980, 2003. 22

W.Y. Chang, C.S. Chen, and Y.P. Hung. Tracking by parts: A bayesian approach with component collaboration. *IEEE Transactions on Systems, Man, and Cybernetics*, 39(2):375–388, 2009. 33

T. Chateau, F. Jurie, M. Dhome, and X. Clady. Real-time tracking using wavelet representation. In *DAGM-Symposium*, Lecture Notes in Computer Science, pages 523–530, 2002. 87

R.T. Collins. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recognition*, pages 234–240, 2003. 20

D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition*, 2000. 19, 32

T.F. Cootes and C.J. Taylor. Active shape models - 'smart snakes'. In *British Machine Vision Conference*, 1992. 28

T.F. Cootes, D.H. Cooper, C.J. Taylor, and J. Graham. A trainable method of parametric shape description. pages 54–61, 1991. 28

T.F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Training models of shape from sets of examples. In *British Machine Vision Conference*, pages 9–18, 1992. 28

T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001. 28

D. Cristinacce and T.F. Cootes. Feature detection and tracking with constrained local models. In *British Machine Vision Conference*, pages 929–938, 2006. 32, 87

F.C. Crow. Summed-area tables for texture mapping. *SIGGRAPH Computer Graphics*, 18:207–212, 1984. 16, 87, 91

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, pages 886–893, 2005. 15

J. Daugmann. Uncertainty relation for resolution in space, spatial frequency and orientation optimized by two-dimensional visual cortical filters. *Journal of the Optical Society of America*, 2(7):1160–1169, 1985. 14

J. Deutscher, A. Blake, and I.D. Reid. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition*, pages 2126–2133, 2000. 34

T. Dietterich, R. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997. 24

U. Drepper. What every programmer should know about memory. 2007. 92

M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *British Machine Vision Conference*, 2009. 29, 40

Z. Fan, M. Yang, and Y. Wu. Multiple collaborative kernel tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1268–1273, 2007. 32, 40

P.F. Felzenszwalb and D.P. Huttenlocher. Efficient matching of pictorial structures. In *Computer Vision and Pattern Recognition*, 2000. 34

P.F. Felzenszwalb and D.P. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004. 34

P.F. Felzenszwalb and D.P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005. 29, 34, 38, 40

P.F. Felzenszwalb, R.B. Girshick, D.A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 29

V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Computer Vision and Pattern Recognition*, Alaska, 2008. 35

M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, C-22(1):67 – 92, 1973. 29, 34, 38

W.T. Freeman and E.H. Adelson. The design and use of steerable filters. 13(9): 891–906, 1991. 14

Y. Freund and R.E. Schapire. A decision-theoretic generalization of the on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–27, Barcelona, Espanha, 1995. 24

K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32 – 40, 1975. 32

M. Godec, P.M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In *International Conference on Computer Vision*, pages 81–88, 2011. 6, 27, 39, 40

D.A. Gomez Jáuregui, P. Horain, M.K. Rajagopal, and S.S.K. Karri. Real-time particle filtering with heuristics for 3D motion capture by monocular vision. In *International Workshop on Multimedia Signal Processing*, pages 139–144, 2010. 34, 65

H. Grabner and H. Bischof. On-line boosting and vision. In *Computer Vision and Pattern Recognition*, pages 260–267, 2006. 24, 25

H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*, pages 234–247, 2008. 24

S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, 1987. 23

S. Gu and C. Tomasi. Branch and track. In *Computer Vision and Pattern Recognition*, pages 1169–1174, 2011. 34

S. Gu, Y. Zheng, and C. Tomasi. Efficient visual object tracking with online nearest neighbor classifier. In *Asian Conference on Computer Vision*, pages 271–282, 2010. 14, 53, 62

G.D. Hager, M. Dewan, and C.V. Stewart. Multiple kernel tracking with SSD. In *Computer Vision and Pattern Recognition*, pages 790–797, 2004. 20

P.S. Heckbert. Filtering by repeated integration. In *Proceedings of the 13th annual conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 315–321, New York, NY, USA, 1986. 15, 91, 108

Tim Hauke Heibel, Ben Glocker, Martin Groher, Nikos Paragios, Nikos Komodakis, and Nassir Navab. Discrete tracking of parametrized curves. In *CVPR*, pages 1754–1761, 2009. 81

G. Hua and Y. Wu. Sequential mean field variational analysis of structured deformable shapes. *Computer Vision and Image Understanding*, 101(2):87–99, 2006. 33

M. Isard. PAMPAS: Real-valued graphical models for computer vision. In *Computer Vision and Pattern Recognition*, pages 613–620, 2003. 33

M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. 21, 33

A. Jezierska, H. Talbot, O. Veksler, and D. Wesierski. A fast solver for truncated-convex priors: Quantized-convex split moves. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 45–58, 2011. 10

Hao Jiang, Tai-Peng Tian, Kun He, and Stan Sclaroff. Scale resilient, rotation invariant articulated object matching. In *CVPR*, pages 143–150. IEEE, 2012. 41

F. Jurie and M. Dhome. Real time robust template matching. In *British Machine Vision Conference*, 2002. 18

W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828, 1978. 52

Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012. 25, 39, 53, 60, 62

M. Kass, A. Witkin, and D. Terzopoulos. Snakes - active contour models. *International Journal Of Computer Vision*, 1(4):321–331, 1987. 28

D.A. Klein and A.B. Cremers. Boosting scalable gradient features for adaptive real-time tracking. In *International Conference on Robotics and Automation*, pages 4411–4416, 2011. 24, 39, 53, 60, 62

I. Kokkinos. Rapid deformable object detection using dual-tree branch-and-bound. In *Neural Information Processing Systems*, pages 2681–2689, 2011. 35

M.P. Kumar, P.H.S. Torr, and A. Zisserman. Learning layered pictorial structures from video. In *Conference on Computer Vision, Graphics and Image Processing*, pages 158–164, 2004a. 34

P.M. Kumar, P.H.S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *British Machine Vision Conference*, 2004b. 29

J. Kwon and K.M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *Computer Vision and Pattern Recognition*, 2009. 14, 27, 39

S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005. 14

I. Leichter. Mean shift trackers with cross-bin metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):695–706, 2012. 20

S.X. Li, H.X. Chang, and C.F. Zhu. Adaptive pyramid mean shift for global real-time visual tracking. *Image and Vision Computing*, 28(3):424–437, 2010. 21

S.Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *European Conference on Computer Vision*, pages 67–81, 2002. 15

R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *International Conference on Image Processing*, pages 900–903, 2002. 92, 106

D.G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999. 14

B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence*, pages 674–679, 1981. 12, 18, 19

E. Maggio and A. Cavallaro. Hybrid particle filter and mean shift tracker with adaptive transition model. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 221–224, 2005. 22

E. Maggio, F. Smeraldi, and A. Cavallaro. Combining colour and orientation for adaptive particle filter-based tracking. In *British Machine Vision Conference*, 2005. 42

B. Martinez and X. Binefa. Piecewise affine kernel tracking for non-planar targets. *Pattern Recognition*, 41(12):3682–3691, 2008. 33

L. Masson, M. Dhome, and F. Jurie. Tracking 3d objects using flexible models. In *British Machine Vision Conference*, 2005. 28

T. Mauthner, M. Donoser, and H. Bischof. Robust tracking of spatial related components. In *International Conference on Pattern Recognition*, pages 1–4, 2008. 33, 40

P.F. McLauchlan, D. Beymer, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Computer Vision and Pattern Recognition*, pages 495–501. IEEE Computer Society, 1997. 21

Branislav Micusk. Trajectory reconstruction from non-overlapping surveillance cameras with relative depth ordering constraints. In *International Conference on Computer Vision*, pages 922–928, 2011. 90

D. Nilsson and J. Goldberger. Sequentially finding the N-best list in hidden markov models. In *International Joint Conferences on Artificial Intelligence*, 2001. 90

K. Nummiaro, E. Koller-Meier, and L. Van Gool. A color-based particle filter. In *International Workshop on Generative-Model-Based Vision*, pages 53–60, 2002. 21

W. Ouyang, R. Zhang, and W.K. Cham. Fast pattern matching using orthogonal haar transform. In *Computer Vision and Pattern Recognition*, pages 3050–3057, 2010. 15, 92

N.C. Oza. *Online Ensemble Learning*. PhD thesis, The University of California, Berkeley, CA, 2001. 24

N. Padoy and G.D. Hager. Deformable tracking of textured curvilinear objects. In *British Machine Vision Conference*, 2012. 65, 80

D. Park and D. Ramanan. N-best maximal decoders for part models. In *International Conference on Computer Vision*, pages 2627–2634, 2011. 30, 34

M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *Computer Vision and Pattern Recognition*, pages 1353–1360, 2011. 35

P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision*, volume 2350, pages 661–675, 2002. 21, 40, 54

F. M. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Computer Vision and Pattern Recognition*, pages 829–836, 2005. 14

D. Ramanan. Learning to parse images of articulated bodies. In Bernhard Schlkopf, John Platt, and Thomas Hoffman, editors, *Neural Information Processing Systems*, pages 1129–1136. MIT Press, 2006. 29, 40

D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):65–81, 2007. 15, 29, 30, 34, 87, 88, 89

S. Romdhani. *Face Image Analysis using a Multiple Features Fitting Strategy*. PhD thesis, University of Basel, Basel, Switzerland, 2005. 28

R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *European Conference on Computer Vision*, pages 700–714, 2002. 29, 40

Remi Ronfard. Region-based strategies for active contour models. *International Journal of Computer Vision*, 13(2):229–251, 1994. 28

R. Rosales and S. Sclaroff. 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *Computer Vision and Pattern Recognition*, pages 2117–2123, 1999. 21

A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *International Conference on Computer Vision Workshop on On-line Learning for Computer Vision*, 2009. 25, 39, 53, 62, 88

J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition*, pages 723–730, 2010. 5, 23, 24, 25, 39, 45, 53, 55, 62

B. Sapp, C. Jordan, and B. Taskar. Adaptive pose priors for pictorial structures. In *Computer Vision and Pattern Recognition*, pages 422–429, 2010. 29, 40

B. Sapp, D. Weiss, and B. Taskar. Parsing human motion with stretchable models. In *Computer Vision and Pattern RecognitionPR*, pages 1281–1288, 2011. 65

J.M. Saragih, S. Lucey, and J.F. Cohn. Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision*, 91(2):200–215, 2011. 32

G. Schindler and F. Dellaert. A rao-blackwellized parts-constellation tracker. In *International Conference on Computer Vision Workshop on Dynamical Vision*, pages 178–189, 2005. 34

S. Sclaroff and J. Isidoro. Active blobs: region-based, deformable appearance models. *Computer Vision and Image Understanding*, 89(2-3):197–225, 2003. 28, 29

D.W. Scott. Multivariate density estimation. In *Wiley*, 1992. 13

S. M. N. Shahed, J. Ho, and M.H. Yang. Online visual tracking with histograms and articulating blocks. *Computer Vision and Image Understanding*, 114(8):901–914, 2010. 40

L.G. Shapiro and G.C. Stockman. Computer vision. In *Prentice-Hall*, 2001. 13

H. Y. Shum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, 2000. 19

H. Sidenbladh, M.J. Black, and D.J. Fleet. Stochastic tracking of 3D human figures using 2D image motion. In *European Conference on Computer Vision*, pages 702–718, 2000. 30

P. Simard, L. Bottou, P. Haffner, and Y. LeCun. Boxlets: A fast convolution algorithm for signal processing and neural networks. In *Neural Information Processing Systems*, pages 571–577, 1998. 91, 108

C. Sminchisescu and B. Triggs. Covariance-scaled sampling for monocular 3D body tracking. In *Computer Vision and Pattern Recognition*, volume 1, pages 447–454, Hawaii, 2001. 34

C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3D human tracking. In *Computer Vision and Pattern Recognition*, pages 69–76, 2003. 34

Stefano Soatto and Anthony J. Yezzi. Deformotion: Deforming motion, shape average and the joint registration and segmentation of images. In *ECCV*, volume 2352, pages 32–57, 2002. 30

S. Stalder, H. Grabner, and L. Van Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *International Conference on Computer Vision Workshop on On-line Learning for Computer Vision*, pages 1409 –1416, 2009. 88

C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, pages 2246–2252, 1999. 16

E.B. Sudderth, A.T. Ihler, E.T. Ihler, W.T. Freeman, and A.S. Willsky. Nonparametric belief propagation. In *Computer Vision and Pattern Recognition*, volume 1, pages 605–612, 2003. 33

L. Swirski, A. Bulling, and N.A. Dodgson. Robust real-time pupil tracking in highly off-axis images. In *Eye Tracking Research and Applications*, pages 173–176, 2012. 87

F. Tang, R. Crabb, and H. Tao. Representing images using nonorthogonal Haar-like bases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29: 2120–2134, 2007. 15

T.B. Terriberry, L.M. French, and J. Helmsen. GPU accelerating speeded-up robust features. In *3D Data Processing Visualization and Transmission*, Atlanta, GA, USA, 2008. 92

C. Tomasi and J. Shi. Good features to track. In *Computer Vision and Pattern Recognition*, pages 593–600, 1994. 19

C. Tomasi, S. Petrov, and A. Sastry. 3d tracking = classification + interpolation. In *ICCV*, pages 1441–1448, 2003. 30

R. Urtasun, D.J. Fleet, and P. Fua. 3D people tracking with gaussian process dynamical models. In *Computer Vision and Pattern Recognition*, pages 238–245, 2006. 30

P.A. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, pages 511–518, 2001. 15, 16, 87, 91, 92, 104

P.A. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *International Conference on Computer Vision*, pages 734–741, 2003. 15

M. Vondrak, L. Sigal, and O. C. Jenkins. Physical simulation for probabilistic motion tracking. In *Computer Vision and Pattern Recognition*, 2008. 30

D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, (2):224–241, 2011. 30

M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *British Machine Vision Conference*, 2009. 12

D. Wesierski, P. Horain, and Z. Kowalczuk. EBE: Elastic Blob Ensemble for coarse human tracking. In *International Conference on Image Processing*, pages 1625–1628, 2012a. 10

D. Wesierski, M. Mkhinini, P. Horain, and A. Jezierska. Fast recursive ensemble convolution of Haar-like features. In *Computer Vision and Pattern Recognition*, pages 3689–3696, 2012b. 10

K. Wnuk and S. Soatto. Multiple instance filtering. In *Neural Information Processing Systems*, pages 370–378, 2011. 39, 53, 62

L. Xiao and P. Li. Improvement on mean shift based tracking using second-order information. In *International Conference on Pattern Recognition*, pages 1–4, 2008. 20

Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *Computer Vision and Pattern Recognition*, 2011. 29, 108

Y. Yang, S. Baker, A. Kannan, and D. Ramanan. Recognizing proxemics in personal photos. In *Computer Vision and Pattern Recognition*, 2012. 29

A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006. 12

K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *European Conference on Computer Vision*, pages 864–877, 2012. 88

Song Chun Zhu and Alan L. Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900, 1996. 28

X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition*, 2012. 29, 34

Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *International Conference on Pattern Recognition*, volume 2, pages 28 – 31, 2004. 16

Zoran Zivkovic and Ben J. A. Kröse. An EM-like algorithm for color-histogram-based object tracking. In *Computer Vision and Pattern Recognition*, pages 798–803, 2004. 20