Non-rigid multi-modal object tracking using Gaussian mixture models

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

by
Prakash Chockalingam
August 2009

Accepted by:
Dr. Stan Birchfield, Committee Chair
Dr. Robert Schalkoff
Dr. Brian Dean

Abstract

This work presents an approach to visual tracking based on dividing a target into multiple regions, or fragments. The target is represented by a Gaussian mixture model in a joint feature-spatial space, with each ellipsoid corresponding to a different fragment. The fragment set and its cardinality are automatically adapted to the image data using an efficient region-growing procedure and updated according to a weighted average of the past and present image statistics. The fragment modeling is used to generate a strength map indicating the probability of each pixel belonging to the foreground. The strength map provides vital information about new fragments appearing in the scene, thereby assisting in addressing problematic cases like self-occlusion. The strength map is used by the region growing formulation, reminiscent of discrete level set implementation, to extract accurate boundaries of the target. Significant speedup is achieved using the region growing procedure over traditional level set based methods. The joint Lucas-Kanade feature tracking approach is also incorporated for handling large unpredictable motions even in untextured regions. Experimental results on a number of challenging sequences demonstrate the effectiveness of the technique.

Dedication

I would like to dedicate this work to my parents for willingly or unwillingly supporting all my endeavours. I would also like to dedicate this work to all my friends for all the wonderful time that I had spent with them.

Acknowledgments

I would like to thank Dr Birchfield for all the brainstorming discussions and great inputs for this thesis work. I would also like to thank my friend and colleague, Nalin Pradeep, for his great help and support at all the stages of this work. My sincere thanks to my co-advisors, Dr Robert Schalkoff and Dr Brian Dean, for their insightful inputs and critics about the work.

Table of Contents

List of Figures

Chapter 1

Introduction

Object tracking is the process of extracting the spatial location and temporal trajectory of an object in a video. Additionally, the tracker can also report the properties of the tracked object like size, orientation, and shape. It is one of the widely researched problem in computer vision as it forms a critical task of many applications such as:

- Security and Surveillance: To detect, monitor and recognize people in a scene to provide better sense of security [29, 15]
- Automated Video Annotation and Retrieval: To automatically annotate temporal sequences with object tags and index them for efficient content-based retrieval of videos [2]
- Medical Imaging: To assist doctors during surgeries and enhance existing medical techniques [6]
- Human-Computer Interaction: Applications like face tracking, gesture recognition, and hand tracking can provide natural ways of interacting with intelligent systems and perceptual user interfaces [9]
- Traffic Management: To extract statistics about the traffic information from the cameras and automatically direct traffic flow based on the statistics [35]
- Video Editing or compression: To provide automated video editing and selective video compression based on the region of interest maintaining a high quality for the region of interest and low quality for the rest of the background scene [12]

- Augmented reality: To combine the real-world information with computer generated information based on the tracked data [24, 43]
- Behavior Analysis: To track persons in smart rooms and to interpret their behavior [57]
- Automobile Navigation: To assist drivers by tracking vehicles and other obstacles in the scene [4]

Tracking objects in a temporal sequence is a challenging task because of large unpredictable object and camera motion, non-rigid deformations of the object, complexity in the visual information of the object and background scene, similarity in the appearances of the object and the background, continually changing appearance of the object and scene, self-occlusion, partial and complete occlusion by the background, and compression quality of the video.

To overcome these difficulties, a large number of tracking algorithms are presented in the literature based on the specific tracking domain and the purposes for which the tracker is being built. These algorithms impose different constraints on the tracking problem to produce the best results for their specific needs. The various stages involved in building a general purpose tracker are:

- **Feature Selection:** Features that best discriminate the target from the background need to be chosen. Common features employed in tracking are color spaces (e.g. RGB or HSV), texture, motion, edges, and shape. Many algorithms use multiple features to obtain best results. Typically, feature selection is an offline decision process made based on the purpose of the tracker. However, there are online feature selection mechanisms [16] and boosting techniques [26] to adaptively increase the separability of the object from the background.
- **Target Model:** The next important task is to decide on a representation mechanism to model the object and background using the given features. Representation mechansims can be templates [25], active appearance models [20] or probability density of features which can either be parametric, such as Gaussian [47] and mixture of Gaussians [28] or non-parametric, such as histograms [60, 62], spatiograms [8], and Parzen windows [21].
- **Target Detection:** Target detection has to be performed on the first frame when the object appears in the scene. Targets are detected either by identifying the salient points in the image, such as SIFT detector [39], KLT detector [49], or by segmenting the image into perceptually

similar regions using mean-shift [17] or graph-based [23] segmentation. Another popular way for detecting targets is using the temporal information [57] over a sequence of frames.

- **Tracking Approach:** There are a large number of tracking approaches proposed. One of the most commonly employed technique is kernel tracking where the parametric motion of the target is iteratively computed between subsequent frames using the target representation by methods like mean-shift [17], continuously adaptive mean-shift (CAMSHIFT) [9], dense [31] and sparse [40] optical flow. Another line of approach is to use filtering techniques [32, 10] which use a state space approach to model the discrete-time dynamic properties of the object. If the state of the object, say position or shape, is assumed to have a Gaussian distribution, then the state of the object can be estimated reliably using a Kalman filter. A more efficient approach would be to consider a non-Gaussian distribution and use a particle filter. Multiple measurements leads to a large state space which is practically difficult for these filters to handle. Hence, to track multiple objects, a correspondence needs to be established across frames. Naive approaches like nearest neighbor might fail in cases of occlusions, entries, and exits of objects in the scene. A deterministic approach would be to formulate the correspondence problem into graph assignment problem and the cost function can be optimized using Hungarian method. A more recent technique is to use Joint Probability Density Association Filters [13] which uses a statistical approach to address the issue.

- **Target Update:** Target update is the process of updating the target model as the target evolves over time in a temporal sequence. This mechanism depends on the target representation. A naive solution is to update the target model with the tracked data available in the previous frame. If the tracking approach has errors, then such a naive update introduces small errors, termed as drift. Different techniques for updating models such as templates [42], GMMs [58] have been proposed to reduce the drift as much as possible.

1.1 Related work

Recent interest in visual tracking has centered around two major areas: (i) on-line learning of multiple cues to adaptively select the most discriminative ones, and (ii) extracting accurate contours of the objects being tracked.

Off-line boosting was first introduced by Tieu and Viola [51] for feature selection where the training procedure adds a weak classifier to the ensemble and evaluates it to obtain a weight for each weak classifier. The linear combination of the weights of the weak classifiers form the strong classifier. Significant progress has been achieved to make this learning process online. Collins *et al.* [16] evaluate multiple feature spaces every frame and choose the color space that gives the maximum separability between the object and background. This work is extended by Avidan [5] where a group of weak classifiers work together to separate the object from the background, and they are integrated over time to ensure the temporal coherence. The weak classifiers are combined with a strong classifier using AdaBoost to provide a confidence measure for each pixel. Oza and Russell [46] introduced the online boosting technique which uses a Poisson sampling process to approximate the reweighting algorithm of off-line boosting methods. Grabner *et al.* [26, 27] extend this work by applying the online boosting technique only to a subset of weak classifiers chosen based on an optimisation criterion.

Apart from on-line learning of features, one of the other most focused area is extracting accurate contours of the object. Contour-based tracking can be done either using an explicit representation [13] or an implicit representation [62, 60] of the contour. Condensation algorithm [32] uses a particle filter to estimate the current state of the system defined in terms of spline shape parameters and affine motion parameters by constructing prior probability density function using the previous states of the system and the observation density function using the image edges computed in the normal direction to the contour. However, such explicit representations with spline curves do not easily allow topological changes [44]. Hence most of the recent approaches use an implicit representation of contours using level sets and iteratively evolve the contour by directly minimizing the contour energy functional. The contour energy can be temporal information [18] or appearance information [60]. Cremers and Schnorr [18] use the optical flow as the contour energy. In [60], the tracking energy is based on the shape and a statistical semi-parametric model of the visual cues. Zhang and Freedman [62] follow a similar approach by combining the energy functional of the level sets based on density matching with the shape priors. Brox *et al.* [11] perform motion segmentation using level sets by defining the energy functional in terms of gray or color value constancy, gradient constancy, temporal smoothness and contour length. Shi and Karl [50] propose a fast implementation method of level sets to achieve real-time tracking.

1.2 Motivation

In most of these tracking approaches [26, 27, 5, 16], tracking is formulated as a classification problem in which the probability of each pixel belonging to the target is computed. While the results have been impressive, several limitations remain:

- Although the tracker locks onto the most discriminative cue, it ignores important but secondary cues. For example, the model may capture the skin of a person's face, but not the hair. This is because of the employment of linear classifiers to classify pixels either to foreground or background. Linear classifiers can produce excellent results if the underlying data is linearly separable. But, in most of the tracking scenarios, the object and the scene are multi-modal and complex, and the data is not linearly separable.

- These algorithms produce a strength image indicating the probability of each pixel belonging to the object being tracked, but they provide no mechanism for determining the shape of the object. And without a multi-modal distribution, the strength image does not make this possible.

- A parametric multi-modal representation such as mixture of Gaussians will solve the aforementioned problems but the process of breaking the multi-modal data into unimodal representation needs to be automatic, *i.e.*, the algorithm should automatically find the number of modes in a computationally efficient manner.

- Occlusion of the target can cause the learner to adapt to occluding surfaces, thus causing the model to drift from the target. If the occlusion is long enough, this can lead to tracking failure. An accurate representation of the contour would enable such errors to be prevented.

- Spatial information that captures the joint probability of pixels is often ignored. This leads to an impoverished tracker that is not able to take advantage of the wealth of information available in the spatial arrangement of the pixels in the target. This arrangement has been shown to be a rich source of information in both classic template-based and more recent techniques [34, 30].

1.3 Approach

This thesis work presents a technique that overcomes the limitations mentioned above. The multi-modal target is modeled using multiple unimodal regions by splitting the target into a number of fragments similar to Adam *et al.* [1]. This also preserves the spatial relationships of the pixels. Unlike their work, however, our fragments are adaptively chosen according to the image data by clustering pixels with similar appearance rather than using a fixed arrangement of rectangles. This adaptive fragmentation captures all the secondary cues and also ensures that each fragment captures a single mode of the distribution. We classify individual pixels, as in [5, 16], but by incorporating multiple fragments we are better able to preserve the shape of multi-modal targets. The boundary is represented using both an explicit and implicit model so that the tracker can evolve the contour from its previous position using an efficient discrete implementation that is more computationally efficient than level set based approaches. This work extends the variational work of [60] by allowing multimodal backgrounds, extreme shape changes, and unpredictable motion. Finally, to address the problem of drastically moving targets with untextured regions, the recently proposed approach of [7] is employed to impose a global smoothness term in order to produce accurate sparse motion flow vectors for each fragment. The fragment models are then updated automatically using the estimated contour and the image data. The employment of adaptive fragments using traditional level set framework to track multi-modal objects is published in [14].

The work is organized as follows. Chapter 2 describes the object representation using Gaussian mixture models and the computation of the strength image from the object model. Chapter 3 explains the novel region growing model and uses the proposed model to extract contours of the object from the strength image, and also to segment the image for finding the fragments and their parameters in the initial frame. Chapter 4 discusses the adaptive updating of the appearance and spatial parameters of the fragments. Chapter 5 presents the summary of the entire algorithm. Experimental results on several challenging sequences are shown in Chapter 6 and Chapter 7 concludes the work with the novel contributions and future directions.

Chapter 2

Tracking Framework

2.1 Bayesian formulation

Our goal is to estimate the contour from a sequence of images. Let $I_t : \mathbf{x} \to \mathbb{R}^m$ be the image at time t that maps a pixel $\mathbf{x} = [x\ y]^T \in \mathbb{R}^2$ to a value, where the value is a scalar in the case of a grayscale image ($m = 1$) or a three-element vector for an RGB image ($m = 3$). The value could also be a larger vector resulting from applying a bank of texture filters to the neighborhood surrounding the pixel, or some combination of these raw and/or preprocessed quantities. Similar to [60], we use Bayes' rule and an assumption that the measurements are independent of each other and of the dynamical process to model the probability of the contour Γ at time t given the previous contours $\Gamma_{0:t-1}$ and all the measurements $I_{0:t}$ of the causal system as

$$p(\Gamma_t | I_{0:t}, \Gamma_{0:t-1}) \propto \underbrace{p(I_t^+ | \Gamma_t)}_{\text{target}} \ \underbrace{p(I_t^- | \Gamma_t)}_{\text{background}} \ \underbrace{p(\Gamma_t | \Gamma_{t-1})}_{\text{shape}}, \tag{2.1}$$

where $I_t^+ = \{\xi_I(\mathbf{x}) : \mathbf{x} \in R^+\}$ captures the pixels inside Γ_t, $I_t^- = \{\xi_I(\mathbf{x}) : \mathbf{x} \in R^-\}$ captures the pixels outside Γ_t, and $\xi_I(\mathbf{x}) = [\mathbf{x}^T\ I(\mathbf{x})^T]^T$ is a vector containing the pixel coordinates coupled with their image measurements. Appendix A shows a derivation of the above equation.

Assuming conditional independence among the pixels, the joint probability of the pixels in a region is given by

$$p(I_t^\star | \Gamma_t) = \prod_{\mathbf{x} \in R^\star} p_\star(\xi_I(\mathbf{x}) | \Gamma_t), \tag{2.2}$$

where $\star \in \{-, +\}$.

2.2 Discriminant function

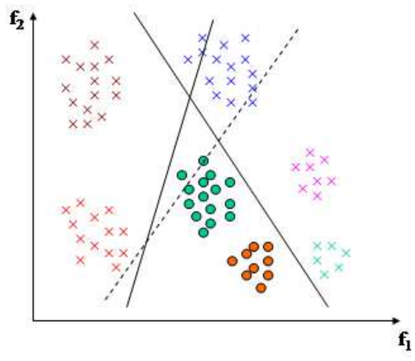
2.2.1 Linear classifiers

One way to represent the probability of a pixel $\xi_I(\mathbf{x})$ is to measure its signed distance to a separating hyperplane H in $\mathbb{R}^n$, where $n = m+2$, as in [5, 16]. The hyperplane $h(x)$ is characterized by $n+1$ parameters: $h(x) = a_0 + a_1x_1 + a_2x_2 + ... + a_nx_n = \sum_{j=0}^{n} a_jx_j$, where $x_0 = 1$. The coefficient a_0 defines the distance of the hyperplane from the origin and the rest of the coefficients determine the orientation of the hyperplane. The parameters of the hyperplane can be learnt offline through boosting techniques. Boosting techniques evaluate different possible hyperplanes termed as weak classifiers. Each weak classifier is given a weight, α, based on its ability to discriminate the classes and they are again combined to form a final strong classifier:

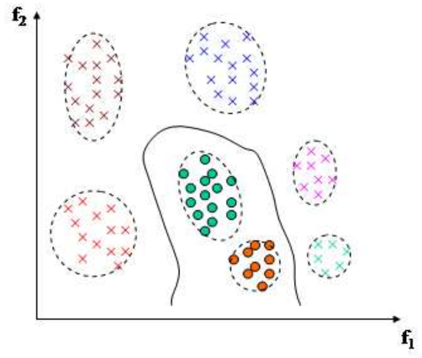
$$H(x) = \sum_{i=1}^{t} \alpha_i h_i(x) = \sum_{i=1}^{t} \alpha_i \sum_{j=0}^{n} a_{ij}x_j = \sum_{j=0}^{n} \sum_{i=1}^{t} (\alpha_i a_{ij})x_j \tag{2.3}$$

where t is total number of weak classifiers. It can be seen that the above linear combination of weak classifiers results in an optimum separating hyperplane whose coefficients are given by $\sum_{i=1}^{t} \alpha_i a_{ij}$. Recent approaches [5, 26] have proposed online boosting techniques to learn the parameters of this optimum separating hyperplance adaptively to improve the results. Avidan [3] uses support vector machine with a variable subset of support vectors to enhance tracking results. Fisher's linear discriminant projects the higher dimensional data onto a line and performs the classification in this one-dimensional space. Collins *et al.* [16] follow a similar approach where the parameters are learnt online by computing and evaluating a separability score for multiple color spaces.

All these approaches assume that the foreground and background samples obtained from the image data are linearly separable. However, in most tracking scenarios, the foreground and background are complex and multi-modal and such an assumption will lead to ignoring secondary cues as shown in Figure 2.1. As a result, the tracker captures the most discriminative cues but fails to handle secondary cues. Such errors might be acceptable for kernel tracking but not for accurate contour tracking. To extract accurate contours, the classifier should be sophisticated enough to discriminate all the object cues from the background.



(a) Linear Classifier



(b) GMM

Figure 2.1: A sample multi-modal data that is not linearly separable. Cross marks indicate foreground samples and circles indicate background samples. (a) Linear classifiers evaluate different hyperplanes represented by the lines. Even an optimum separating hyperplane, indicated by the dashed line, computed from a set of weak classifiers, cannot capture all the modes. (b) Gaussian Mixture Model (GMM): The multi-modal data is modeled as a mixture of Gaussians leading to a curved decision boundary.

2.2.2 Fragment modeling

Another way to represent the probability of a pixel $\xi_I(\mathbf{x})$ is to measure the distance of the pixel to a single covariance matrix as in [47]. A slightly more general approach would be to measure its Mahalanobis distance to a pair of Gaussian ellipsoids representing the target and background. Both these approaches do not capture a multi-modal distribution leading to an impoverished representation of the scene and object. As a result, we instead represent both the target and background appearance using a set of *fragments* in the joint feature-spatial space, where each fragment is characterized by a separate Gaussian hyperelliptical surface, similar to [28]. Letting $\mathbf{y} = \xi_I(\mathbf{x})$ for brevity, the likelihood of an individual pixel is then given by a Gaussian mixture model (GMM):

$$p_\star(\mathbf{y}|\Gamma_t) = \sum_{j=1}^{k_\star} \pi_j p_\star(\mathbf{y}|\Gamma_t, j), \tag{2.4}$$

where $\pi_j = p(j|\Gamma_t)$ is the probability that the pixel was drawn from the jth fragment, $k_\star$ is the number of fragments in the target or background (depending upon $\star$), $\sum_{j=1}^{k_\star} \pi_j = 1$, and

$$p_\star(\mathbf{y}|\Gamma_t, j) = \eta \exp\left\{ -\frac{1}{2}(\mathbf{y} - \mu_j^\star)^T \left(\Sigma_j^\star\right)^{-1} (\mathbf{y} - \mu_j^\star) \right\}, \tag{2.5}$$

where $\mu_j^\star \in \mathbb{R}^n$ is the mean and $\Sigma_j^\star$ the $n \times n$ covariance matrix of the jth fragment in the target or background model, and η is the Gaussian normalization constant.

2.3 Computing the strength image

The recent approach of formulating the object tracking problem as one of binary classification between target and background pixels [5, 26, 27] is employed. In this approach, a strength image is produced indicating the probability of each pixel belonging to the target being tracked. The strength image is computed using the log ratio of the probabilities:

$$S(\mathbf{x}) = \log\left(\frac{p_+(\mathbf{x})}{p_-(\mathbf{x})}\right) = \Theta^-(\mathbf{x}) - \Theta^+(\mathbf{x}), \tag{2.6}$$

where $\Theta^\star(\mathbf{x}) = -\log p_\star(\mathbf{x})$. Positive values in the strength image indicate pixels that are more likely to belong to the target than to the background, and vice versa for negative values. An example strength image is shown in Figure 2.3, illustrating the improvement achieved by considering spatial information. The strength image is used to update the implicit function, which enables the region growing formulation, discussed in the next chapter, to enforce smoothness on the resulting object shape.

2.4 Comparison with other representations

The fragment modeling is compared with two other discriminant functions: (i) A single Gaussian [47] where the foreground and background is modeled using a single Gaussian each and (ii) Collins *et al.* [16] approach where different color spaces are evaluated for maximum separability between the foreground and background. Figure 2.2 shows the strength image obtained using the above approaches on a synthetic toy sequence which simulates the sample distribution shown in Figure 2.1 on a 2D feature space consisting of the red and green channels. Figure 2.3 shows the comparison on a real world image.

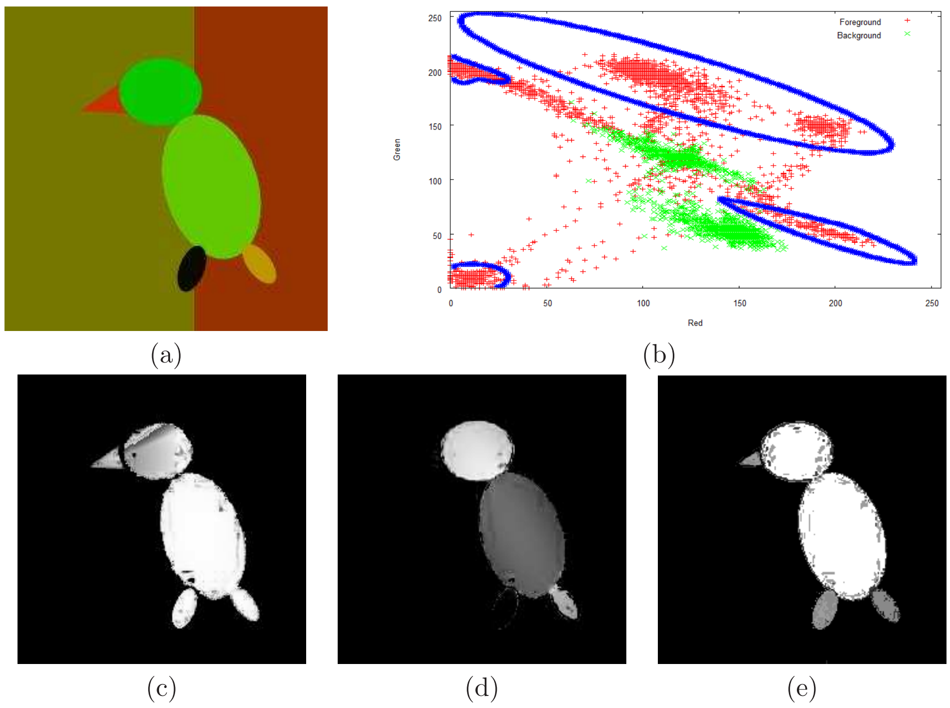


Figure 2.2: (a) Synthetic toy image generated using only the red and green channels (b) The 2D feature space showing the color distribution of the image. The foreground pixels correspond to red pluses and background pixels correspond to green crosses. The blue curves correspond to the actual decision boundary for the foreground computed by the fragment modeling approach. The bottom row shows the final strength image computed using (c) our approach, (d) a single Gaussian [47] and (e) a linear separation over a linear combination of multiple color spaces [16]. Our fragment-based GMM representation more effectively represents the multi-colored target.

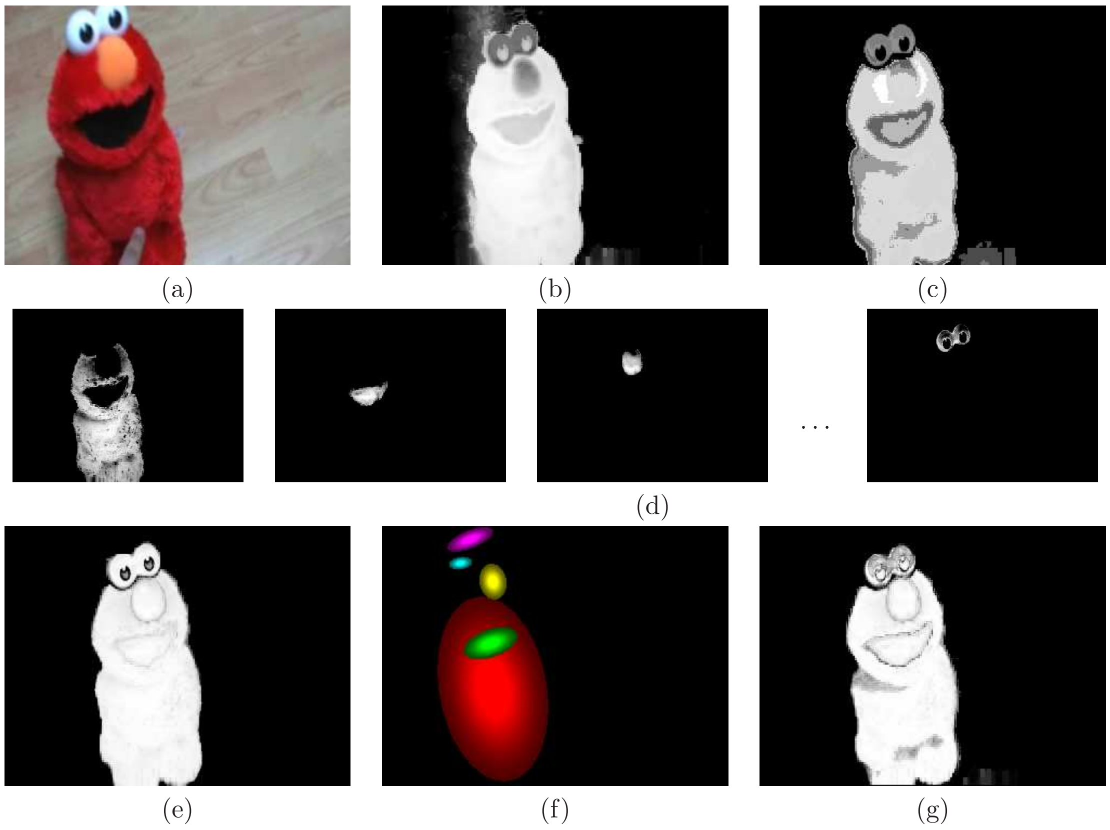


Figure 2.3: (a) Image of Elmo. The strength image computed using a single Gaussian [47] and linear combination of multiple color spaces [16] are shown in (b) and (c) respectively. The probabilities determined by individual fragments (d) are combined to form the strength image of our approach (e). The different foreground spatial ellipsoids that contributed to the strength image are shown in (f). To show the significance of capturing the spatial information in the object model, (g) shows the strength image computed without the spatial information.

Chapter 3

Region Growing

Region growing is the process of expanding or contracting a region based on its properties and those of its neighborhood. It is one of the classic approaches for solving computer vision problems like segmentation [63], region matching [52, 38], and stereo matching [55, 45]. Typically, a region growing algorithm starts with a seed point or seed area and then progressively evaluates and adds or discards neighbors to the region based on their similarity to the region until a stopping criterion is met.

One of the explicit earlier works on region growing is by Otto and Chau [45] for stereo matching based on the adaptive least squares correlation algorithm where patches between two satellite images are iteratively matched and grown based on the distortion parameters identified from the previous match. Vedaldi and Soatto [52] use the region growing approach to match and align regions using local affine features, and they demonstrate that aligning regions during the growing process provides more discriminative ability than incorporating the affine features as part of the region representation model. Wei and Quan [55] employ a growing-like process for stereo matching using a best-first strategy based on a cost function involving disparity smoothness and visibility constraints. A similar region growing based approach has also been employed for dense matching of textured scenes by Lhuillier [38]. Shi and Karl [50] propose a discrete implementation of level sets, similar to the region growing approaches, where the algorithm can expand or contract the region and also handle topological changes and multiple regions by switching boundary pixels between two linked lists. Our region growing model resembles this approach whereby the proposed model combines an explicit representation and an implicit representation of the region to handle both contour evolution and typical segmentation and region matching problems.

A generic region growing model is outlined in the next section. Section 3.2 explains the application of the region growing model for contour evolution as a better implementation alternative than the traditional level sets approach. Section 3.3 deals with the application of the region growing model for segmenting the image, the key initialization step in the tracking process, to extract the adaptive fragments from the image data.

3.1 Region growing model

The region growing algorithm starts with a seed region, Ω, and uses two kinds of representation: an explicit representation using a singly linked list, $\mathcal{L}$, and an implicit representation, Φ, similar to the level sets approach. Φ is initialized as follows:

$$\Phi(x) = \begin{cases} +1 & , x \in \Omega \\ -1 & , otherwise \end{cases} \tag{3.1}$$

The singly linked list, $\mathcal{L}$, is initialized as:

$$\mathcal{L} = \{x : x \in \Omega, \exists x' \in N_4(x) \text{ such that } x' \notin \Omega\} \tag{3.2}$$

At any instance of time during the region growing, $\mathcal{L}$ represents the boundary of the region being grown. The region evolution proceeds by evolving Φ over time to maximize the following energy functional:

$$E = \sum_{x \in \Lambda} \Phi(x)\Psi(x) + \alpha K(\Phi) \tag{3.3}$$

where Λ represents the image domain, $\alpha K(\Phi)$ is a regularization term to maintain the smoothness of the curve being evolved, α is the regularization parameter and $\Psi(x)$ represents the likelihood term obtained from a confidence measure of each pixel belonging to the actual region R. The smoothness constraint is enforced by smoothing the likelihood term using a Gaussian kernel G. The energy functional now becomes:

$$E = \sum_{x \in \Lambda} \Phi(x)\Psi(x) + \alpha \sum_{x \in \Lambda} G * \Psi(x) \tag{3.4}$$

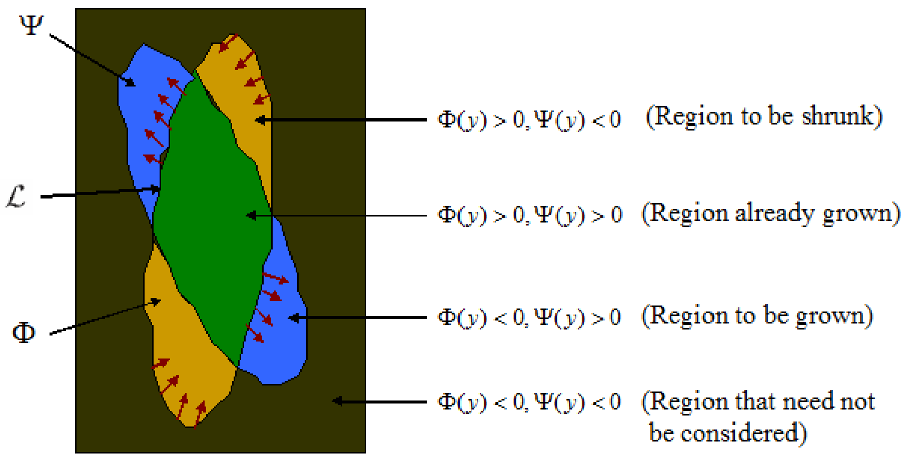

Ψ
$\mathcal{L}$
Φ
$\Phi(y) > 0, \Psi(y) < 0$ (Region to be shrunk)
$\Phi(y) > 0, \Psi(y) > 0$ (Region already grown)
$\Phi(y) < 0, \Psi(y) > 0$ (Region to be grown)
$\Phi(y) < 0, \Psi(y) < 0$ (Region that need not be considered)

Figure 3.1: Demonstration of the region growing algorithm where the frontier $\mathcal{L}$ evolves from its current position using the implicit representation Φ and the likelihood of the region Ψ. Four region types are possible using Φ and Ψ which are indicated by different colors.

where $*$ indicate convolution. The above enforcement of the smoothness constraint is similar to the work by Vedaldi and Soatto [52]. But their region growing algorithm does not support both expansion and contraction.

A demonstration of the region growing algorithm is shown in Figure 3.1. The region growing procedure updates the frontier $\mathcal{L}$ for every iteration by considering only the neighbors of the current frontier. Letting $\Psi_g(x) = G * \Psi(x)$ for brevity, $\forall x \in \mathcal{L}$,

$$\begin{aligned} \mathcal{L}^{k+1} = \quad & \mathcal{L}^k \oplus \{x' : x' \in N_4(x) \text{ and } \Phi(x') < 0, \Psi_g(x') > 0\} \\ & \ominus \{x : \forall x' \in N_4(x) \text{ such that } \Psi_g(x) > 0, \Psi_g(x') > 0\} \\ & \oplus \{x' : x' \in N_4(x) \text{ and } \Phi(x') > 0, \Psi_g(x') < 0\} \\ & \ominus \{x : \forall x' \in N_4(x) \text{ such that } \Psi_g(x) < 0, \Psi_g(x') < 0\} \end{aligned} \tag{3.5}$$

The $\oplus$ and $\ominus$ operator define the addition and removal of elements from the set. The first term deals with the expansion of the region. A neighboring pixel of the frontier that is not part of the current region is added to the frontier if it has a positive likelihood value. During such an expansion step, some frontier pixels may become interior and need to be removed. The second term removes such interior pixels. The third term correspond to the contraction of the region. A neighboring pixel of the frontier that is part of the current region with a negative likelihood is added to the frontier to contract the region. During this step, existing frontier pixels may become exterior and the fourth term removes such exterior pixels from the frontier.

Since Φ is an implicit representation of the region, there is no necessity of removing any interior or exterior pixels. For every iteration, Φ is updated as: $\forall x \in \mathcal{L}$,

$$\Phi^{k+1}(x') = \begin{cases} +1 & , x' \in N_4(x) \text{ and } \Phi^k(x') < 0, \Psi_g(x') > 0 \text{ (expansion)} \\ -1 & , x' \in N_4(x) \text{ and } \Phi^k(x') > 0, \Psi_g(x') < 0 \text{ (contraction)} \end{cases} \tag{3.6}$$

The region evolution is stopped when no more points are removed or added to the frontier. Figure 3.2 shows the expansion and contraction cases that satisfy the four constraints in equation (3.5). Some algorithms, like [52] and [55], use a best-first strategy based on the cost where precedence is given to one neighbor over the other. Using such a best-first strategy will assist in speeding up the algorithm only in greedy cases where a fixed number of neighbors need to be accepted into the region. In most of the region growing algorithms, all neighbors need to be evaluated for the region to be grown. Hence, we prefer to store the costs and their associated pixels using an unordered linked list rather than a heap or some ordered sequence thereby reducing the running time by a logarithmic amount of the perimeter of the region for every iteration of the region growing method.

In concept, this region growing model is similar to Shi and Karl's work [50]. However, this work differs from theirs in two ways: (1) The goal of expansion and contraction is achieved using only a singly linked list instead of two frontiers, and (2) They enforce the smoothness constraint only after the region growing iterations. Hence, their algorithm operates as two different iterations. But this work incorporates both the region growing and smoothness constraint in a single iteration, thereby maximizing the energy functional in (3.4) during every iteration of the region growing process.

The algorithm is presented in Algorithm 1. Lines 3-12 represent the expansion step in which 4-9 expand the region and 10-12 remove interior points from the frontier. Lines 13-22 correspond to the contraction step in which 14-19 perform the contraction of the region and 20-22 remove exterior points from the frontier.

3.2 Contour extraction

Recent state-of-the-art tracking algorithms are based on extracting accurate contours of the objects. Contours capture accurate spatial arrangement of the object than their rectangle and ellipti-

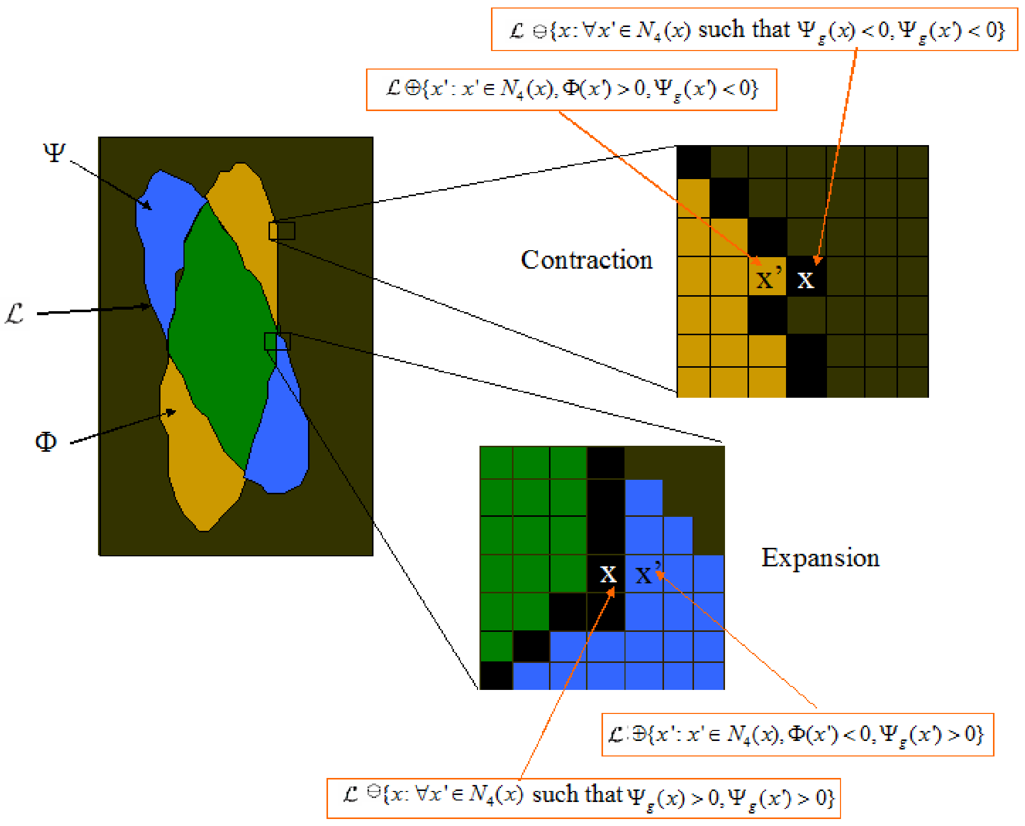

$\mathcal{L} \ominus \{x : \forall x' \in N_4(x)$ such that $\Psi_g(x) < 0, \Psi_g(x') < 0\}$
$\mathcal{L} \oplus \{x' : x' \in N_4(x), \Phi(x') > 0, \Psi_g(x') < 0\}$
Ψ
$\mathcal{L}$
Φ
Contraction
x'
x
Expansion
x
x'
$\mathcal{L} \oplus \{x' : x' \in N_4(x), \Phi(x') < 0, \Psi_g(x') > 0\}$
$\mathcal{L} \ominus \{x : \forall x' \in N_4(x)$ such that $\Psi_g(x) > 0, \Psi_g(x') > 0\}$

Figure 3.2: Cases of expansion and contraction are shown for the pixels where one of the four conditions in equation (3.5) are satisfied.

Algorithm 1 Region Growing Algorithm

Require: The likelihood term Ψ, Φ initialized as given in (3.1) and the list $\mathcal{L}$ initialized as given in (3.2)

Ensure: Φ represents the entire region and $\mathcal{L}$ represents the boundary of the region.

```
repeat
  for all $x \in \mathcal{L}$ do
    { Expansion Step: }
    for all $x' \in N_4(x)$ do
      if $\Phi(x') < 0$ AND $\Psi_g(x') > 0$ then
        $\mathcal{L} \leftarrow \mathcal{L} \oplus x'$
        $\Phi(x') \leftarrow +1$
      end if
    end for
    if $\Psi_g(x) > 0$ AND $(\Psi_g(x') > 0, \forall x' : x' \in N_4(x))$ then
      $\mathcal{L} \leftarrow \mathcal{L} \ominus x$
    end if
    { Contraction Step: }
    for all $x' \in N_4(x)$ do
      if $\Phi(x') > 0$ AND $\Psi_g(x') < 0$ then
        $\mathcal{L} \leftarrow \mathcal{L} \oplus x'$
        $\Phi(x') \leftarrow -1$
      end if
    end for
    if $\Psi_g(x) < 0$ AND $(\Psi_g(x') < 0, \forall x' : x' \in N_4(x))$ then
      $\mathcal{L} \leftarrow \mathcal{L} \ominus x$
    end if
  end for
until $\mathcal{L}$ is not modified
```

cal counterparts and more importantly they give valuable information about the shape of the object. Explicit representation of the contour using spline curves as done in Condensation algorithm [32] does not easily allow topological changes. Hence, many recent tracking approaches [18, 60, 62] use a level set framework to iteratively evolve the contour by minimizing the contour energy functional. Though level sets provide an excellent framework for extracting contours, the major drawback with level sets is the computational overhead which makes trackers based on such frameworks unsuitable for real-time tracking. To reduce the computational overhead, we demonstrate the process of extracting the contours using a discrete implementation of level sets by employing the region growing model proposed in the previous section.

The region growing model proposed in section 3.1 is adopted for extracting the contours by setting the initial seed area Ω to the previous contour Γ_{t-1}. The likelihood term, Ψ, is set to the strength map, S, obtained using the feature distributions as given in equation (2.6). Now the model evolves the contour Γ_t from its previous position Γ_{t-1} maximizing the energy functional in (3.4). Such a discrete approach is extremely faster than the traditional level sets approach since it considers only the neighborhood of the current frontier for evolution and there is also no necessity of solving any partial differential equations.

3.3 Region segmentation

Segmentation is the process of spatially grouping pixels that have similar photometric characteristics. The current tracking framework requires the multi-modal object and scene to be fragmented into unimodal regions for accurate modeling of the scene. Hence, in the initialization phase, after the object detection, the tracker should automatically learn the number of modes (fragments) in the object and scene and segment them into different unimodal regions based on the visual cues. This fragment-based representation of the target is similar to that of Adam *et al.* [1] but with two significant differences. First, fragments are used to model the background as well as the target, and secondly, the fragments are automatically determined and adapted by the image data rather than being fixed and hardcoded. The challenge is to compute the model parameters $\mu_1^+, \ldots, \mu_{k_+}^+, \Sigma_1^+, \ldots, \Sigma_{k_+}^+, \mu_1^-, \ldots, \mu_{k_-}^-, \Sigma_1^-, \ldots, \Sigma_{k_-}^-$ automatically without any prior information about the scene.

Broadly, segmentation methods can be classified into (i) Clustering methods, (ii) Graph-based methods, and (iii) Contour-based methods. A brief note on some of the popular segmentation algorithms that fall into these categories are discussed below, and finally a simple mode-seeking region growing approach is proposed to fragment the object and scene in a computationally efficient manner.

3.3.1 Clustering methods

Segmentation process can be formulated as a clustering problem in the feature space where subsets of the feature vector set can be formed into groupings or 'clusters' using unsupervised learning approaches. K-means algorithm [41] is one of the simplest parametric unsupervised learning algorithm where a centroid is defined for each cluster either randomly or using some heuristics. Each feature vector is assigned to one of the clusters based on the proximity of the feature to the centroids. After all the assignments are done, the centroids are re-calculated and this process is repeated until the centroids do not move. K-means essentially tries to minimize the squared error function $E = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2$, where $||x_i^{(j)} - c_j||^2$ is a distance measure between the feature vector $x_i^{(j)}$ and the centroid c_j, and n is the number of data points. Expectation-Maximization algorithm [19] is another indispensable unsupervised learning algorithm in data clustering where the parameters that characterize the clusters are iteratively refined to maximize a log likelihood function computed with respect to the current estimate of the distribution. The main disadvantages of both these approaches are the number of clusters or modes need to be known a priori and the algorithms are very sensitive to initial choice of parameters and might get trapped in the local maxima of the log likelihood function. Vlassis *et al.* [53] proposed a greedy Expectation-Maximization approach to overcome these limitations where new clusters are added sequentially in a two step procedure (i) a global search to find a new cluster and (ii) a local search with incremental density estimation to add the new cluster to the existing clusters. But when we tried this approach we found the estimation of the number of the modes were found to be too unreliable for our purposes. Mean-shift segmentation [17] is a non-parametric mode-seeking technique to segment the image. The algorithm is initialized with a large number of random cluster centers and each cluster center is moved to the mean of the data surrounding the cluster center. The mean-shift vector, defined by the old and new cluster centers, is computed iteratively until the centers do not change. Figure 3.3 shows the performance of the algorithm. Though the results are generally quite good, the algorithm was found to be too slow to be employed for tracker initialization

3.3.2 Graph-based methods

Graph-based methods represent the segmentation problem in terms of a graph where each node represents a pixel, and the edges connect a pixel with neighboring pixels in the image. Each edge is undirected and associated with a weight based on the pixel properties of the two nodes such as color or grayscale values. The work by Zahn [61] is one of the earliest graph-based methods to segment images which uses the minimum spanning tree of a graph to find the clusters in the feature space. Recent methods [23, 48] partition the graph into disjoint sets such that the similarity among nodes within a group is high and similarity among nodes across groups is low. The performance of the segmentation algorithm is highly dependent on the cut criterion. Zahn [61] and Wu and Leahy [59] use local properties of the graph as the cut criterion. Though they are computationally efficient, these methods are highly sensitive to noise and the size of regions. To overcome this issue, Shi and Malik [48] proposed the normalized cut which takes into account the global impression of the scene by formulating the cut criterion problem into a generalized eigenvalue problem. Its computational time is generally slow for tracking purposes. Felzenszwalb and Huttenlocher [23] propose a computationally efficient method that captures important non-local image characteristics and runs in $O(n \ \log n)$, where n is the number of pixels. The results of the algorithm are shown in Figure 3.3.

3.3.3 Contour-based methods

Contour-based methods extract a closed curve of the region in the image plane typically seeking towards the detected edge pixels, while satisfying smoothness constraints on the contour. Kass and Witkin [36] proposed a parametric *snake* contour approach. Variational level set based frameworks [11] are also employed for segmentation purposes.

3.3.4 Region growing model for segmentation

We have devised a segmentation algorithm based on the proposed region growing model. One of the drawbacks of a region growing algorithm is its sensitivity to seed points or areas. To alleviate this problem, the seed points for growing the regions are identified by computing a score for every pixel:

$$\eta(x) = \prod_{i=1}^{3} \lambda_i(x) = \det(C) \tag{3.7}$$

where $\lambda_i(x)$'s are the eigenvalues of C, which is a 3×3 covariance matrix constructed from the color distribution, $f(x) = [f_r(x)\ f_g(x)\ f_b(x)]^T$, over a window $\mathcal{R}$ centered at x:

$$C = \frac{1}{|\mathcal{R}|} \sum_{x \in \mathcal{R}} (f(x) - \mu)(f(x) - \mu)^T \tag{3.8}$$

where $\mu = \frac{1}{|\mathcal{R}|} \Sigma_{x \in \mathcal{R}} f(x)$ is the mean feature descriptor in $\mathcal{R}$, and $|\mathcal{R}|$ is the number of pixels in the region $\mathcal{R}$. The computed scores η are stored in an ordered list $\mathcal{S} =< \nu_1, ..., \nu_n >$ where $\nu_i = (x_i, y_i, \eta_i)$. The minimum element in $\mathcal{S}$ signifies that the region around this pixel is more homogeneous and can serve as a good seed point for the region growing algorithm. The likelihood term in the energy functional in (3.4) for a region is defined as:

$$\Psi_j(x) = MD(f(x), \mathcal{N}(\mu_j, \Sigma_j)) - \tau \tag{3.9}$$

where τ is a configurable parameter whose value can be tuned manually to obtain best segmentation results and $MD(f(x), \mathcal{N}(\mu_j, \Sigma_j))$ represent the Mahalanobis distance of the feature vector, $f(x)$, to the appearance of the region modeled as a single Gaussian using the mean μ_j and the covariance Σ_j of the features in the region. In our implementation, Σ_j is approximated by using only the diagonal elements which form the 3×1 variance vector, $\bar{\sigma_j}^2$, of the feature vectors. The parameters, mean μ_j and covariance Σ_j, of the region are continually updated using a running accumulation of first- and second-order statistics as given in Appendix B.

The entire algorithm can be summarized as:

- Step (i): The seed area Ω is initialized using the window $\mathcal{R}_j$ centered at the current minimum element of $\mathcal{S}$.
- Step (ii): The state information of the region growing algorithm, Φ and $\mathcal{L}$, are then initialized as given in equations (3.1) and (3.2) respectively.
- Step (iii): The parameters of the region, mean μ_j and covariance Σ_j, are computed from $\mathcal{R}_j$.
- Step (iv): The region is grown as given in Algorithm 1 with two additional steps carried out on each addition of a pixel to the growing region: (1) The parameters of the region, mean μ_j and covariance Σ_j, are updated based on the new pixel's value and (2) The corresponding element in $\mathcal{S}$ is removed.

- Step (v): Steps (i) - (iv) are repeated to grow more regions if $\mathcal{S}$ is not empty. If $\mathcal{S}$ is empty, it signifies that all the pixels in the image have already been associated with a fragment and the segmentation process can terminate.

Fragments smaller than a fixed area are discarded, and the remaining fragments are labeled as target or background depending upon whether the majority of pixels are within or without, respectively, a manually drawn initial contour Γ_0. Any fragment for which the pixels are roughly evenly distributed is split along Γ_0 to form two fragments, one labeled foreground and the other labeled background. Finally, the π_j's, used in equation (2.4), are computed based on the size of the fragments. The output of the region growing algorithm is presented in Figure 3.3. Our algorithm is not only faster but also better than the graph-based approach [23] and mean-shift [17] in the first five images. The last row shows a sequence where our algorithm accidentally merges two different regions and oversegments the image. Also, our approach is found to be slightly sensitive to the threshold τ given in equation (3.9).

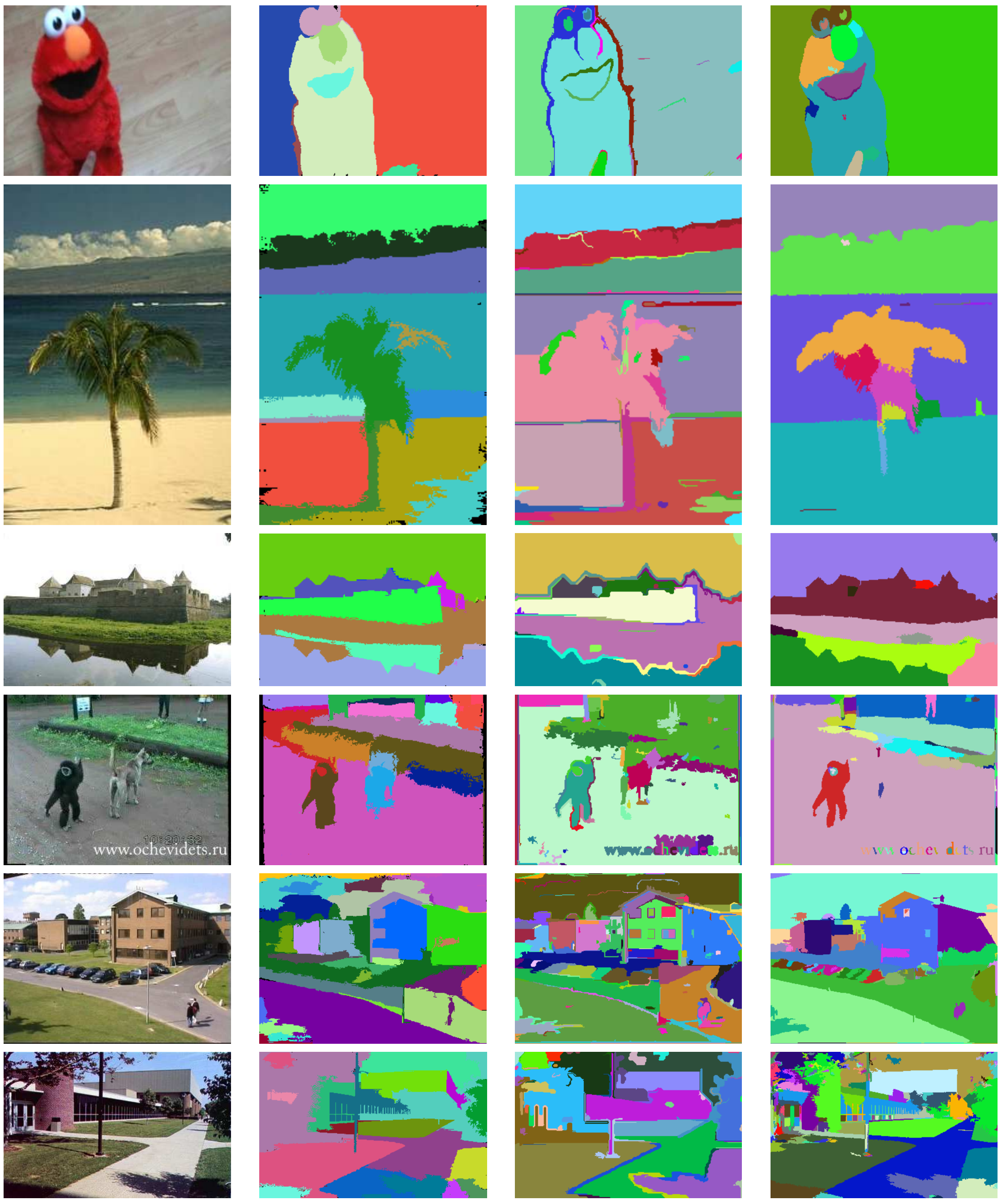

(a) Image (b) Ours (c) Graph (d) Mean-shift

Figure 3.3: Segmentation results of the region growing algorithm compared with other methods. (a) Original image. Segmentation results using (b) Our approach (c) Graph-based approach [23] and (d) Mean-shift algorithm [17]. Mean-shift algorithm takes roughly 30 seconds to segment an image of size 320 $\times$ 240. Our approach and graph-based approach completes the segmentation process in less than a second.

Chapter 4

Update Mechanism

One of the key tasks in a tracking system is to update the object model. In most of the tracking scenarios, the underlying image data, the object, and the scene, evolve over time in a temporal sequence. In such scenarios, the assumption of a constant object or background model over the entire sequence will lead to an impoverished tracker which cannot handle photometric differences and occlusions. Hence it is essential to learn the object model and adapt accordingly.

The most obvious solution of updating the model with the recently tracked output leads to the concept of model drift where the tracker slowly deviates from the object due to minor errors in the recently tracked output. Matthews *et al.* [42] update the model based on a drift correction strategy that evaluates a reference model and the recently tracked output. Klinkenberg and Joachims [37] propose a method based on support vector machines which maintains a temporal window of adaptive size on the previous samples (tracked outputs). Only the samples under the temporal window that maximizes the similarity with the current target model are used for updating the model. Widmer and Kubat [56] also use a similar adaptive temporal window-based strategy using heuristics to tackle drift problems. Wang *et al.* [54] propose a framework for handling drifts using weighted ensemble classifiers to throw away irrelevant data based on their expected classification accuracy on the samples. Though such adaptive temporal windows and ensemble classifiers help in throwing away irrelevant information, they come at the cost of computational overhead. To reliably update the target model and also circumvent the auxillary computational cost, we use a simple weighting mechanism involving all the past and recently tracked outputs.

4.1 Updation of appearance statistics

In this work, the objects are modeled using a Gaussian mixture of fragments in the feature-spatial domain, and it is essential to update the appearance and spatial parameters of all the components (fragments). In the context of tracking, update strategies should handle three cases: (i) Update the parameters of the existing components, (ii) Detect the outdated or occluded components, and (iii) Find new components. Our update mechanism handles all three cases as detailed below.

4.1.1 Updating statistics of existing fragments

Once the target has been tracked to the current image frame I_t, the existing GMMs representing the target and background is updated in the following manner. First, for each pixel we find the fragment that contributed most to its likelihood:

$$\zeta(\mathbf{x}) = \arg\max_{j=1,\ldots,k^\star} p_\star(\xi_{I_t}(\mathbf{x})|\Gamma_{t-1}, j). \tag{4.1}$$

where k^* is the number of fragments in the foreground ($\star = +$) or background ($\star = -$) model. The fragment association for each pixel on a video sequence is shown in figure 4.1. Then the statistics of each fragment are computed using its associated pixels:

$$\mu^\star_{j,t} = \frac{1}{|\mathcal{Z}^\star_j|}\sum_{\mathbf{x}\in\mathcal{Z}^\star_j} \xi_{I_t}(\mathbf{x}) \tag{4.2}$$

$$\Sigma^\star_{j,t} = \frac{1}{|\mathcal{Z}^\star_j|}\sum_{\mathbf{x}\in\mathcal{Z}^\star_j} \xi_{I_t}(\mathbf{x})\xi_I(\mathbf{x})^T, \tag{4.3}$$

where $\mathcal{Z}^\star_j = \{\mathbf{x} : \zeta(\mathbf{x}) = j, \mathrm{sgn}(\phi(\mathbf{x})) = b(\star)\}$, $b(+) = 1$, $b(-) = -1$, and $\mu^\star_{j,t}$ is $\mu^\star_j$ at time t. After computing the recent statistics, the appearance parameters are then updated using a weighted average of the initial values and a function of the recent values:

$$\mu^\star_{j,t} = \alpha^\star_j \bar{\mu}^\star_{j,0:t} + (1-\alpha^\star_j)\mu^\star_{j,0} \tag{4.4}$$

$$\Sigma^\star_{j,t} = \alpha^\star_j \bar{\Sigma}^\star_{j,0:t} + (1-\alpha^\star_j)\Sigma^\star_{j,0}, \tag{4.5}$$

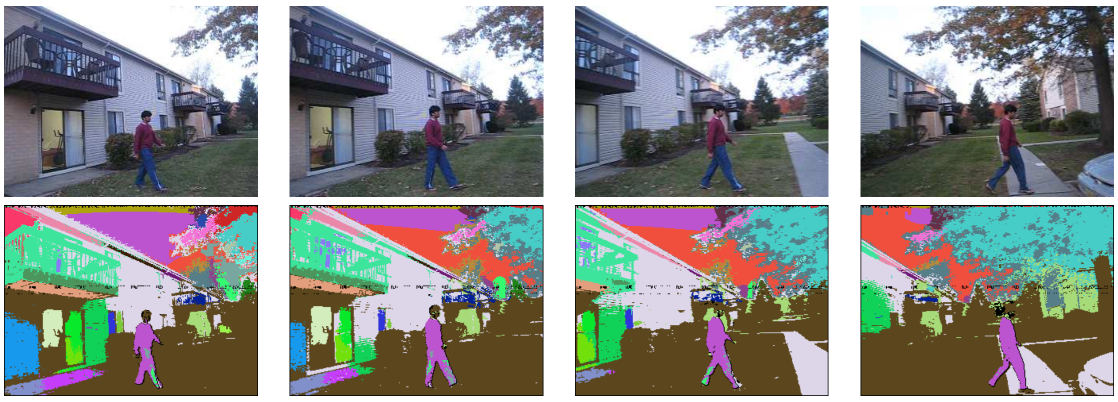


Figure 4.1: The initial model of the scene is created in Frame 000 using the region growing procedure. Fragment association for each pixel in some of the intermediate frames of the sequence based on equation (4.1) is shown.

where $\bar{\mu}^{\star}_{j,0:t}$ is a function of the past and present statistics, e.g.,

$$\bar{\mu}^{\star}_{j,0:t} = \frac{\sum_{\tau=0}^{t} e^{-\lambda(t-\tau)} \mu^{\star}_{j,\tau}}{\sum_{\tau=0}^{t} e^{-\lambda(t-\tau)}} \tag{4.6}$$

$$\bar{\Sigma}^{\star}_{j,0:t} = \frac{\sum_{\tau=0}^{t} e^{-\lambda(t-\tau)} \Sigma^{\star}_{j,\tau}}{\sum_{\tau=0}^{t} e^{-\lambda(t-\tau)}}, \tag{4.7}$$

where λ is a constant ($\lambda = 0.1$). Here, the function $e^{-\lambda(t-\tau)}$ was chosen because the additional old samples do not always help in producing a more accurate hypothesis than using the recent ones [22]. Keeping this analysis in mind, the recent statistics are given more importance than the older ones using this function. The weights are computed by comparing the Mahalanobis distance to the two models:

$$\alpha^{\star}_{j} = \frac{\beta^{\star}_{j,0}}{(\beta^{\star}_{j,0} + \bar{\beta}^{\star}_{j,0:t})} \tag{4.8}$$

where

$$\beta^{\star}_{j,0} = \sum_{\mathbf{x} \in \mathcal{Z}^{\star}_{j}} (\xi_{I_t}(\mathbf{x}) - \mu^{\star}_{j,0})^{T} (\Sigma^{\star}_{j,0})^{-1} (\xi_{I_t}(\mathbf{x}) - \mu^{\star}_{j,0})$$

$$\bar{\beta}^{\star}_{j,0:t} = \sum_{\mathbf{x} \in \mathcal{Z}^{\star}_{j}} (\xi_{I_t}(\mathbf{x}) - \bar{\mu}^{\star}_{j,0:t})^{T} (\bar{\Sigma}^{\star}_{j,0:t})^{-1} (\xi_{I_t}(\mathbf{x}) - \bar{\mu}^{\star}_{j,0:t}).$$

4.1.2 Detecting occluded fragments

The above updating strategy can be easily extended to find occluded fragments. For any fragment j, if $\aleph(\mathcal{Z}_j^\star) < \gamma$, where $\aleph$ represents the cardinality and γ is a constant ($\gamma = 20$), then the fragment is declared as occluded and the spatial model is adapted to that of the target as a whole and the appearance model remains unchanged throughout the occlusion. Finding such occluded fragments could be used as a good heuristic for handling partial occlusions, however this work does not handle partial occlusions.

4.1.3 Finding new fragments: Handling self-occlusion

The most intriguing task in update mechanism is to find new fragments that appear in the video sequences. Finding such new fragments and updating them as part of the object model can aid in self-occlusion cases like out-of-plane rotation of a person's head where the object's appearance shifts from hair color to face color and vice versa. To handle such difficult scenarios, we use the strength image, obtained in eqn (2.6), as $\Theta = \{\mathbf{x} : S(\mathbf{x}) \approx 0\}$. Applying connected components to Θ, all the new fragments are identified. If a new fragment is not adjacent to the object, it is straightaway added to the background model. For fragments that are adjacent to both the foreground and the background, we use the motion cues of the new fragment and the object. It is assumed that if a new fragment is part of the object, then the motion of the new fragment and a part of the foreground object near this new fragment will be similar. With this assumption, motion vectors for the feature points in the new fragment region and a part of the foreground region near the new fragment are obtained as explained in Section 4.2. If the Euclidean distance of the motion vectors for these two regions are less than a threshold (a value of 3 is used in the implementation), then the new fragment is classified as a part of the object and the appearance and spatial information of the new fragment is added to the object model so that the strength image captures the new fragment in the subsequent frames as a part of the object.

4.2 Updation of spatial statistics

The update strategy explained so far handles only appearance statistics. It can be noted that the object is modeled in a joint feature-spatial domain. Hence, updating the spatial statistics assists in aligning the coordinate systems of the target and the model fragments. Such alignment increases the accuracy of the strength image. As a result, we seek to recover, *prior* to computing the strength image, approximate motion vectors between the previous and current image frame for each fragment: $\mathbf{u}_i^\star = (u_i^\star, v_i^\star), i = 1, \ldots, k^\star$.

One way to solve this alignment problem would be to compute the motion of the target using traditional motion estimation techniques. However, existing dense motion algorithms do not perform well on complex imagery in which highly non-rigid, untextured objects undergo drastic motion changes from frame to frame, such as the videos considered in this work. Moreover, dense motion computation wastes precious resources for this application, since we only need approximate alignment between the fragments. In a similar manner, traditional sparse feature tracking algorithms are not suitable for recovering the motions of the individual fragments. Due to their independent handling of the features, such algorithms often yield some percentage of unreliable estimates.

To solve this dilemma, we utilize the recent joint feature tracking approach of [7]. Starting with the well-known *optic flow constraint equation*

$$f(u, v; I) = I_x u + I_y v + I_t = 0, \tag{4.9}$$

the traditional Lucas-Kanade and Horn-Schunck formulations are combined into a single differential framework. The functional to be minimized is given by

$$E_{JLK} = \sum_{i=1}^{N} (E_D(i) + \lambda_i E_S(i)), \tag{4.10}$$

where N is the number of feature points, and the data and smoothness terms are

$$\begin{aligned} E_D(i) &= K_\rho * \left((f(u_i, v_i; I))^2 \right) && (4.11) \\ E_S(i) &= \left((u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2 \right). && (4.12) \end{aligned}$$

In these equations, the energy of feature i is determined by how well its motion $(u_i, v_i)^T$ matches the local image data, and by how far the motion deviates from the expected value $(\hat{u}_i, \hat{v}_i)^T$. The latter is computed by fitting an affine motion model to the neighboring features, where the connections between features are computed by a Delaunay triangulation.

Differentiating E_{JLK} with respect to the motion vectors $(u_i, v_i)^T$, $i = 1, \ldots, N$, and setting the derivatives to zero, yields a $2N \times 2N$ sparse matrix equation, whose $(2i-1)$th and $(2i)$th rows are given by

$$Z_i \mathbf{u}_i = \mathbf{e}_i, \tag{4.13}$$

where

$$\begin{aligned} Z_i &= \begin{bmatrix} \lambda_i + K_\rho * (I_x I_x) & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & \lambda_i + K_\rho * (I_y I_y) \end{bmatrix} \\ \mathbf{e}_i &= \begin{bmatrix} \lambda_i \hat{u}_i - K_\rho * (I_x I_t) \\ \lambda_i \hat{v}_i - K_\rho * (I_y I_t) \end{bmatrix}. \end{aligned}$$

This sparse system of equations can be solved using Jacobi iterations of the form

$$\tilde{u}_i^{(k+1)} = \hat{u}_i^{(k)} - \frac{J_{xx}\hat{u}_i^{(k)} + J_{xy}\hat{v}_i^{(k)} + J_{xt}}{\lambda_i + J_{xx} + J_{yy}} \tag{4.14}$$

$$\tilde{v}_i^{(k+1)} = \hat{v}_i^{(k)} - \frac{J_{xy}\hat{u}_i^{(k)} + J_{yy}\hat{v}_i^{(k)} + J_{yt}}{\lambda_i + J_{xx} + J_{yy}}, \tag{4.15}$$

where $J_{xx} = K_\rho * (I_x^2)$, $J_{xy} = K_\rho * (I_x I_y)$, $J_{xt} = K_\rho * (I_x I_t)$, $J_{yy} = K_\rho * (I_y^2)$, and $J_{yt} = K_\rho * (I_y I_t)$. In practice, Gauss-Seidel iterations with successive overrelaxation yield increased convergence.

Once the N features have been tracked, the motion of each fragment is parameterized by mean and covariance. Mean is used to update spatial coordinate system, while covariance could be used to affect the computation of strength image using spatial information, however currently the covariance of motion features is not incorporated in the implementation. Note that there is little risk to this parameterization, since outliers are avoided by the smoothness term incorporated by the joint Lucas-Kanade approach, which enables features to be tracked even in untextured areas, as shown in [7]. Feature selection is determined by those image locations for which $\max(e_{\min}, \eta e_{\max})$, where $e_{\min}$ and $e_{\max}$ are the two eigenvalues of the 2×2 gradient covariance matrix, and $\eta < 1$ is a scaling factor. Figure 4.2 shows the motion vectors obtained from standard and Joint Lucas-Kanade approaches to demonstrate that the Joint Lucas-Kanade approach produces smoother and reliable motion vectors even in untextured areas. Figure 4.3 shows another comparison of the two methods for the object alone while emphasizing the association of motion vectors to fragments using different colors.

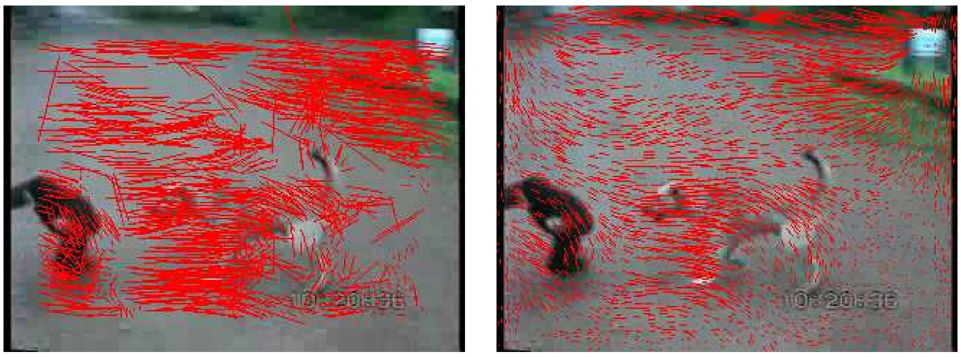


Figure 4.2: Performance of the optical flow computation using LEFT: standard Lucas-Kanade [40] and RIGHT: Joint Lucas-Kanade [7]. It can be seen that the motion vectors obtained from the Joint Lucas-Kanade approach is much smoother than the standard one.

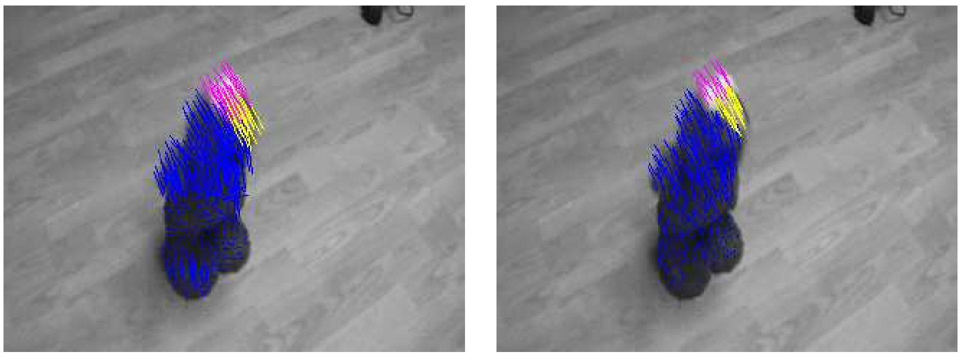


Figure 4.3: Another comparison between standard and Joint Lucas-Kanade approaches where the motion vectors for the foreground region are colored by the fragment in which they are contained.

Chapter 5

Algorithm Summary

The entire algorithm discussed in parts in the previous chapters is sequentially summarized here. To begin, since there is no target detection mechanism, the target is manually initialized by the user with a polygon. The pixels inside the polygon form the seed area Ω. The parameters of the region growing algorithm, Φ and $\mathcal{L}$, are then initialized using Ω as given in equations 3.1 and 3.2 respectively. To proceed with the Bayesian tracking approach, the number of fragments and the initial parameters of the GMM model have to be computed. This is accomplished by segmenting the entire scene using the region growing procedure outlined in Section 3.3.4.

In the tracking phase, the first step is to align the spatial coordinates of the computed model and the actual image data in the current frame by computing an average displacement vector for each fragment using the joint Lucas-Kanade approach described in section 4.2. The average displacement vector is then used to *update* or *pre-correct* the spatial mean compoment of the GMM model parameters. This pre-correction helps in handling large unpredictable frame-to-frame motion. After this updating of spatial statistics, each pixel is classified either into foreground or background using the strength image obtained from equation (2.6). Instead of calculating the strength for each pixel in the frame, we provide an efficient approach using the region growing model given in Algorithm 1. The algorithm uses Φ_{t-1} and $\mathcal{L}_{t-1}$ computed in the previous frame and evolves the contour with the help of the strength image. Since the region growing model evaluates only the pixels near the evolving frontier, it is sufficient that the strength for these pixels alone is computed, thereby reducing the computational complexity of the algorithm. Once the contour, represented by Φ_t and $\mathcal{L}_t$ are obtained, the image data inside this new target is used to update the appearance parameters of the GMM model as given in Section 4.1.

Algorithm: `GMM - Discrete Level Set Implementation`

In first image frame,

1. Manually initialize the contour of the target Γ_0 with a polygon. The pixels inside the polygon represent the seed area Ω.
2. Initialize Φ and $\mathcal{L}$ as given in 3.1 and 3.2 respectively.
3. Initialize GMM models using the region growing procedure of Section 3.3.4 and obtain the model parameters $\mu_1^+, \ldots, \mu_{k_+}^+, \Sigma_1^+, \ldots, \Sigma_{k_+}^+, \mu_1^-, \ldots, \mu_{k_-}^-, \Sigma_1^-, \ldots, \Sigma_{k_-}^-$ for each region in the foreground and background.

To track the target from one image frame to the next,

1. Compute sparse motion vectors $u_i^\star$ for each foreground and background fragment using (4.14) and (4.15). Update the spatial mean component of $\mu_i^\star$ for each fragment using $\bar{u_i}^\star$, the average of the motion vectors in the fragment.
2. Obtain Φ_t and $\mathcal{L}_t$ using Φ_{t-1} and $\mathcal{L}_{t-1}$ as described in Algorithm 1 on page 18. The strength of each pixel $S(x)$ is progressively computed using equation (2.6) only for the pixels near the evolving frontier $\mathcal{L}$.
3. Use Φ_t to update the appearance parameters of GMMs using the procedure given in Section 4.1.
4. $\Phi_{t-1} \leftarrow \Phi_t$ and $\mathcal{L}_{t-1} \leftarrow \mathcal{L}_t$

Figure 5.1: GMM - Discrete level set tracking algorithm summary.

Chapter 6

Results

The algorithm was tested on a number of challenging sequences captured by a moving camera viewing complex scenery. Most of the sequences presented here were chosen so that the tracker could be evaluated for objects undergoing significant scale changes, extreme shape deformation, and unpredictable motion. Some of these sequences were obtained from Internet sources, with high compression, to demonstrate the performance of the algorithm even in poor quality videos. The algorithm was implemented in Visual C++ and runs at 6-10 frames per second on an Intel Core 2 Duo 1.5 GHz machine, depending upon the size of the object and motion. The contour extraction using the region growing procedure provides significant speedup over using the traditional level sets based approach [60] which runs at 1 frame per second.

6.1 Results of the tracker framework

Figure 6.1 shows the results of the algorithm on a sequence of a Tickle Me Elmo doll [1]. The benefit of using a multi-modal framework is clearly shown, with accurate contours (green outlines) being computed despite the complexity in both the target and background as Elmo stands tall, falls down, and sits up. Figure 6.2 shows the output on a poor quality image sequence in which a monkey undergoes rapid motion and drastic shape changes. For example, as the monkey swings around the tree, its shape changes substantially in just a few image frames, yet the algorithm is able to remain locked onto the target as well as compute an accurate outline of the animal. A mosaic of the object

[1]The original datasets and the results of the algorithm on all these sequences are available at http://www.ces.clemson.edu/~stb/research/adafrag

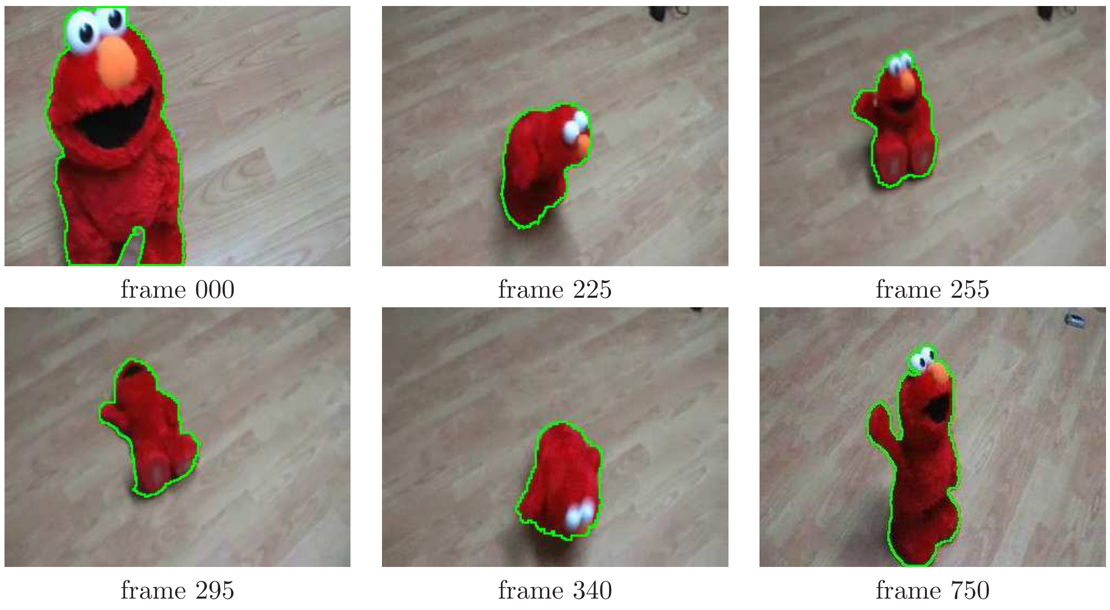


Figure 6.1: Results of the algorithm on the Elmo sequence. Accurate contours are extracted despite noteworthy non-rigid deformations of the Elmo.

contours in the intermediate frames for the Elmo and monkey sequence is shown in Figure 6.5.

Figure 6.3 shows a sequence in which a person walks in a complex scene with many colors. Despite the complexity in the background and the foreground, the person is tracked by the algorithm. Figure 6.4 shows the results of tracking multiple fish in a tank. Multiple fishes are tracked independently by running mulitple instances of the core tracker class. The fish are multicolored and swim in front of a complex, textured, multicolored background. Note that the fish are tracked successfully despite their changing shape. Moreover, note that the small blue fish near the bottom of the tank is camouflaged and yet is recovered correctly due to the effective representation of the object and the background using multiple GMMs.

6.2 Self-occlusion

We also handle cases of self-occlusion, as explained in Section 4.1.3, where new regions or fragments that appear in the scene are identified using the strength image and added to either foreground or background model based on the adjacency and motion of the new region with the target. This particular module was tested with the Elmo sequence with an intermediate frame, where the nose and eyes of the Elmo are invisible, set as the initial frame. The results are presented

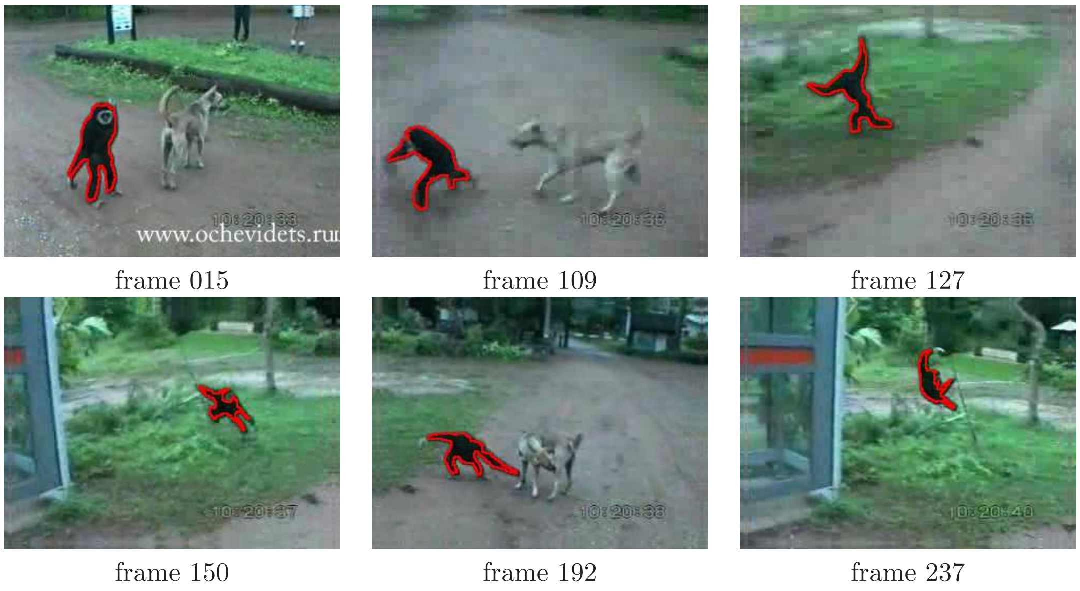


Figure 6.2: Another sequence where the target undergoes significant shape deformation. The target is tracked properly despite both shape deformation and large unpredictable motion.

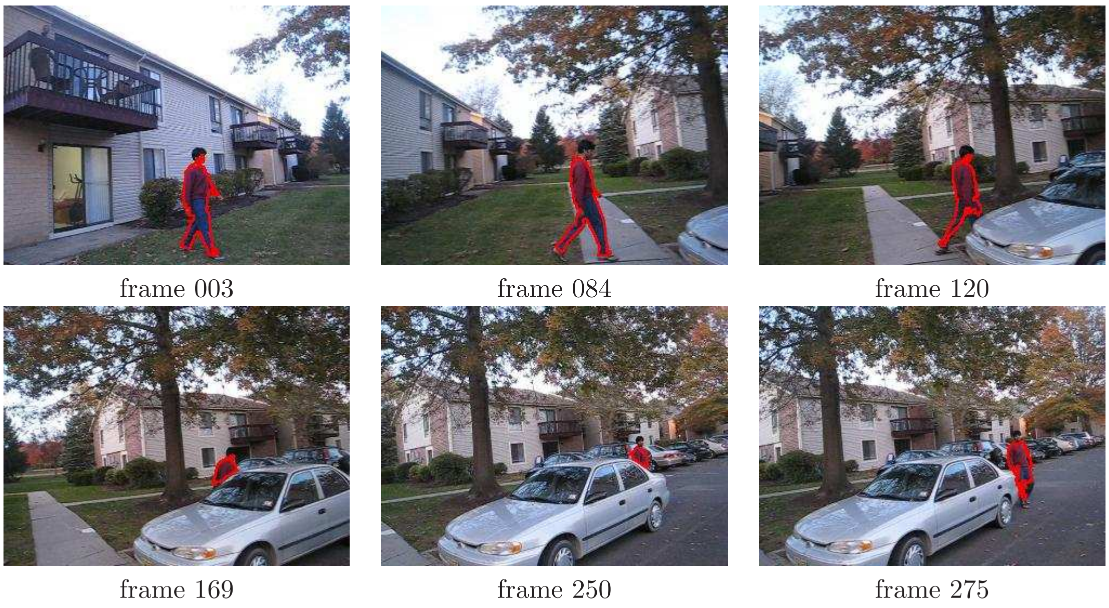


Figure 6.3: Results of the algorithm on the sequence in which a person walks in a complex scene with lot of colors. Note that though the contours are extracted accurately for the body of the person, there is some errors in extracting the contours for the face region due to the complexity of skin color.

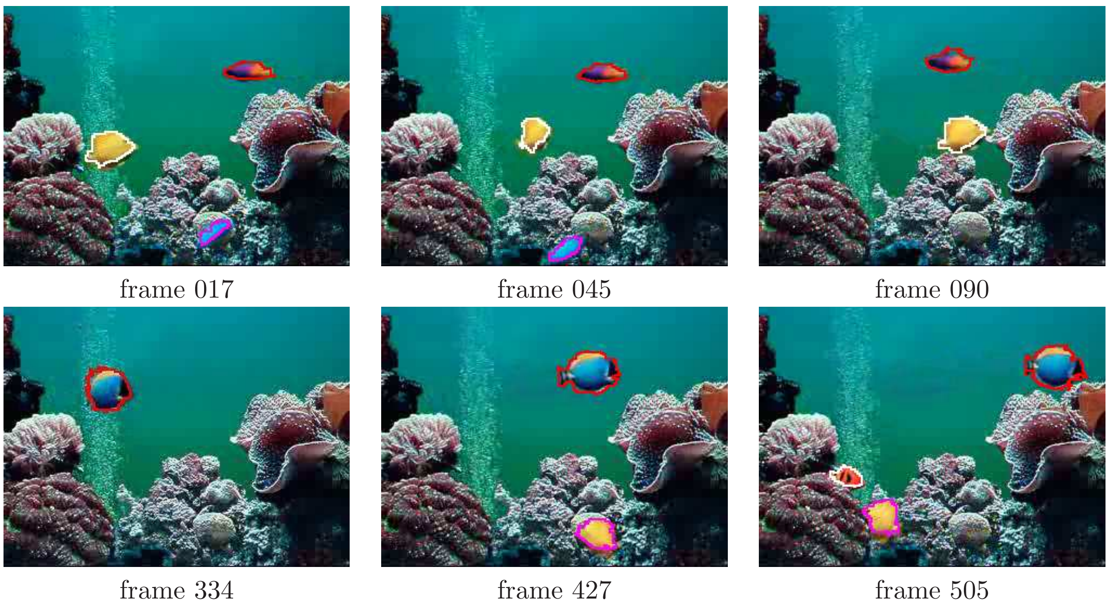


Figure 6.4: Results to demonstrate the effectiveness of the tracker to track multiple objects in a textured background and low quality video. The multi-colored fish are accurately tracked inspite of the complex textured background.

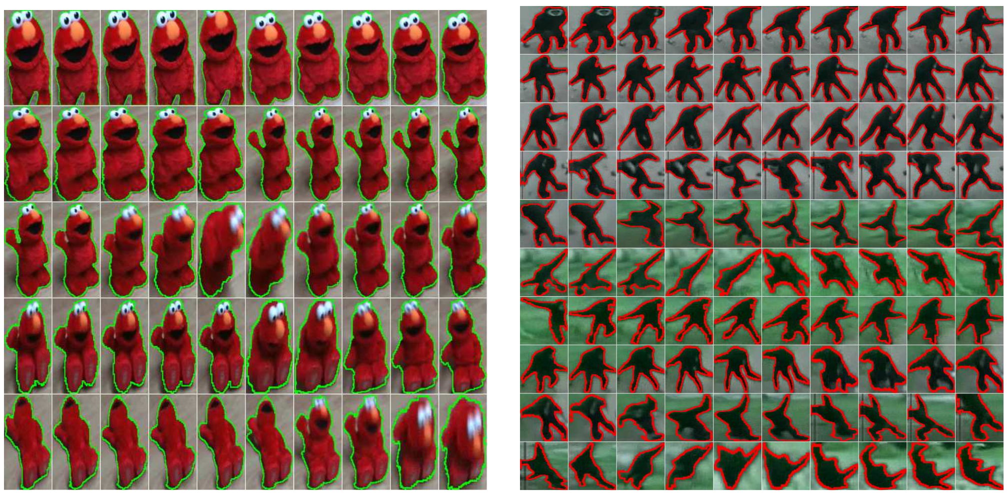


Figure 6.5: Mosaic of resulting contours from Elmo (left) and Monkey (right) sequences.

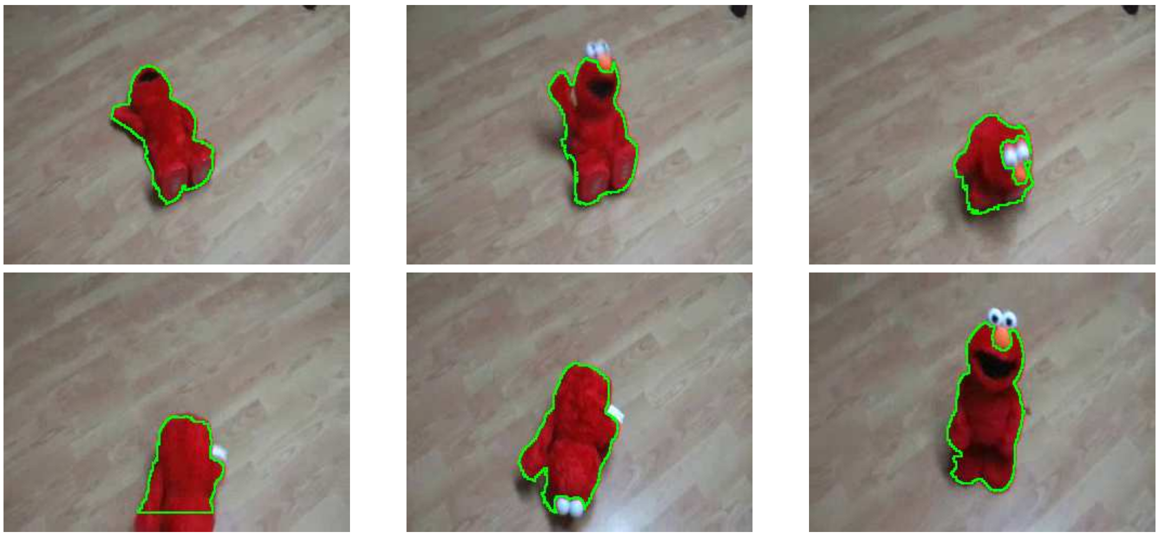
in Figure 6.6. We also tested on another sequence where a person's head is tracked successfully through an out-of-plane rotation as shown in Figure 6.7.

6.3 Comparison with other approaches

To provide quantitative evaluation of our approach, ground truth for the experiments were generated by manually labeling the object pixels in some of the intermediate frames (every 5 frames for the monkey and person sequences, and every 10 frames for Elmo). The error of each algorithm on an image of the sequence was computed as the number of pixels in the image misclassified as foreground or background, normalized by the image size.

The proposed algorithm was compared with two other approaches. In one, the strength image was computed using the linear RGB histogram representation of Collins *et al.* [16]. In the other, the strength image was computed using a standard color histogram, similar to [60, 62, 33, 50]. In both cases the contours were extracted using the wall follow method, but the fragment motion to pre-correct the spatial model was not used as both the approaches were non-parametric. For a fair comparison, we also ran our algorithm without the fragment motion.

Figure 6.8 plots the average normalized error for the three sequences. Our algorithm performs better than the two alternatives on all the sequences. Figure 6.9 shows the extracted contours of all three approaches, overlaid on the object, in some of the intermediate frames. In the case of Elmo, the other two methods have only marginal errors but with the person sequence, the standard color histogram based approach fails to distinguish the target from the background. The linear RGB histogram method performs well but even that approach loses the target when the person is walking behind the car. The importance of incorporating the spatial information and the updation of spatial statistics is prominent in the last few frames of the monkey sequence where the monkey undergoes large frame-to-frame motion. None of the approaches, except our algorithm with the motion, could lock on to the target. We have also compared our technique against a color-based version of FragTrack [1] which also loses the monkey at the end of the sequence.


(a) Without self-occlusion module

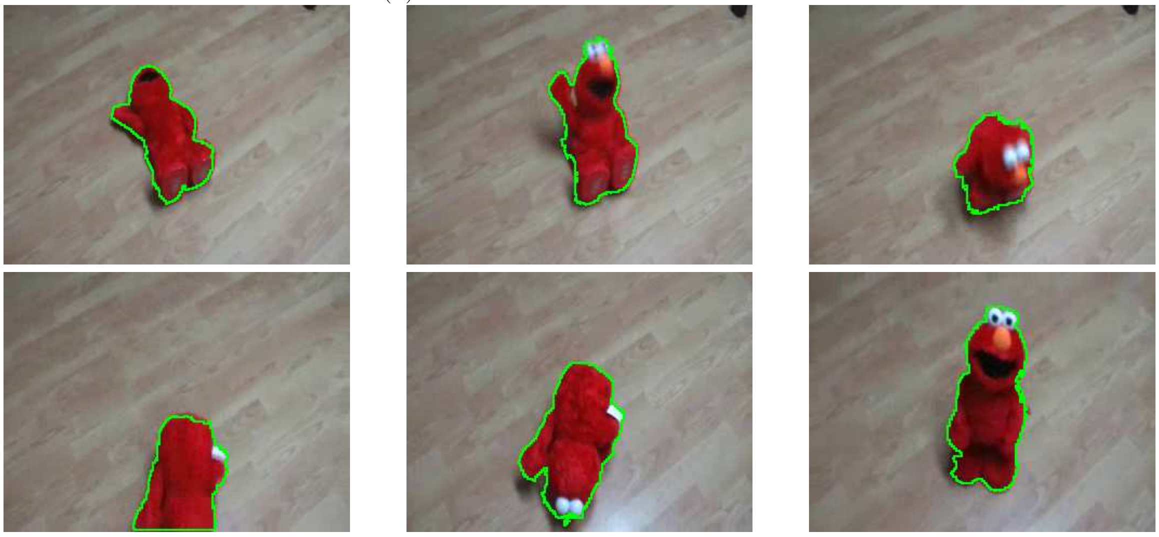
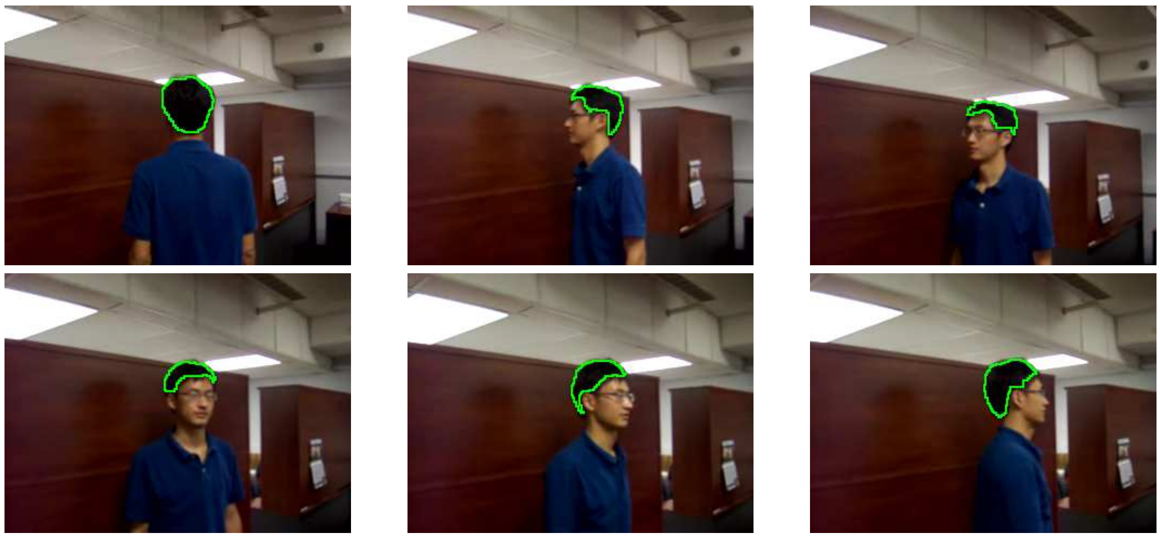
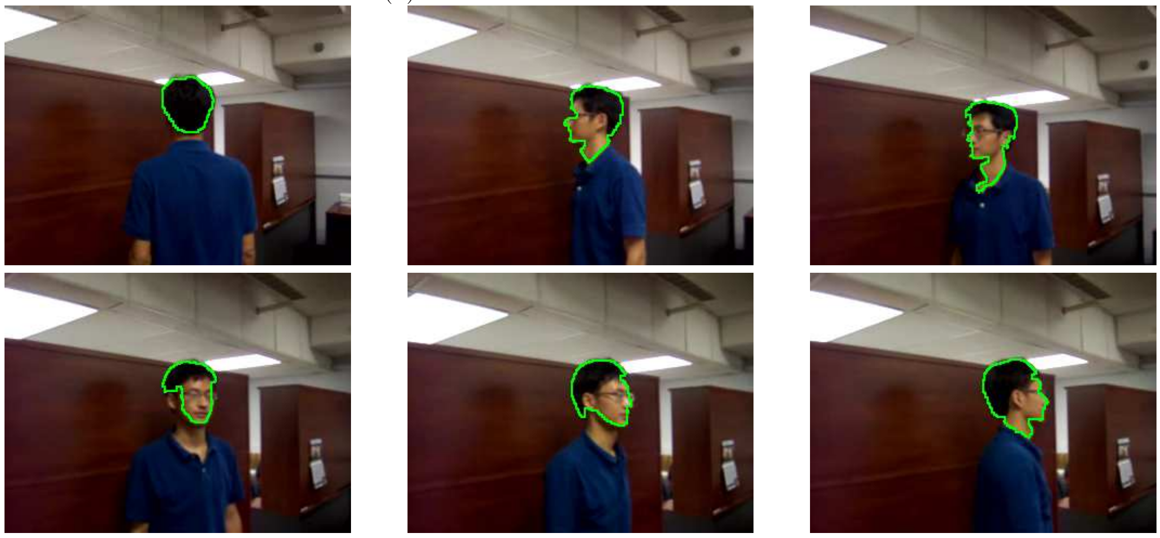

(b) With self-occlusion module

Figure 6.6: Results of handling self-occlusion on the Elmo sequence. The top left frame is the initial frame on which the object is marked. (a) shows results without handling self-occlusion and (b) shows results with the self-occlusion module turned on while tracking. As the Elmo stands up, the new modes (eyes and nose) move along with the object and they are updated into the foreground model. Hence the tracker is able to get an accurate contour of the Elmo in all the frames.

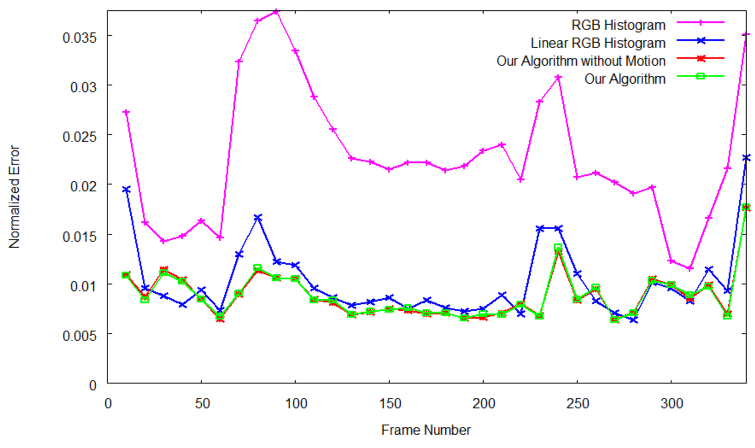


(a) Without self-occlusion module

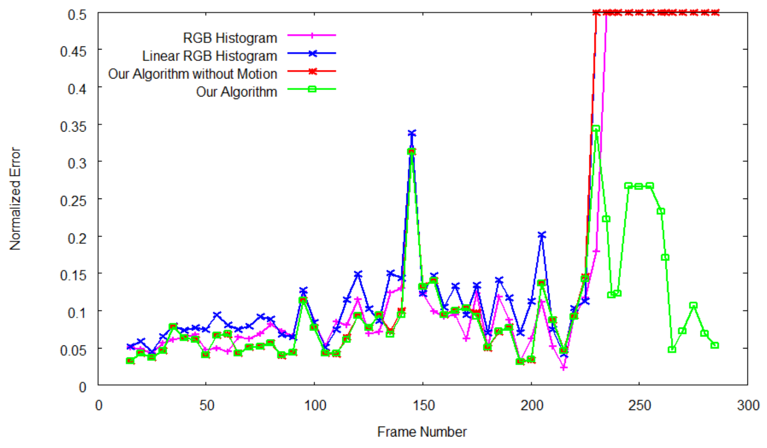


(b) With self-occlusion module

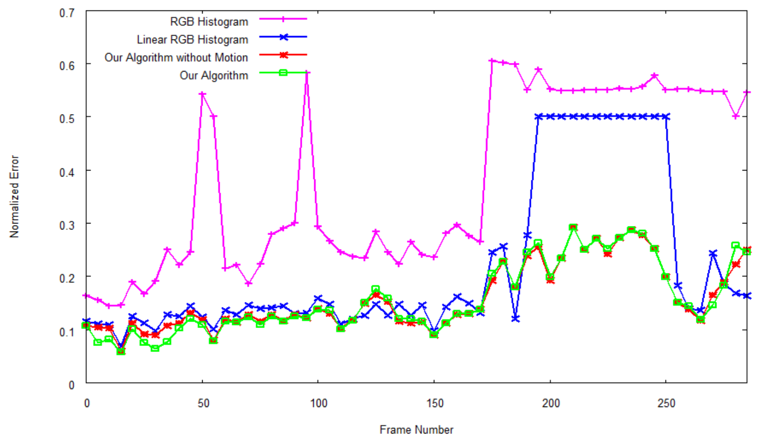
Figure 6.7: Results on another sequence where the person's face has out-of-plane rotation. The initial frame has only the hair color as part of the object model. When the person's skin color appears, the new fragment is added to the object model and hence the face of the person is also tracked.



(a) Elmo

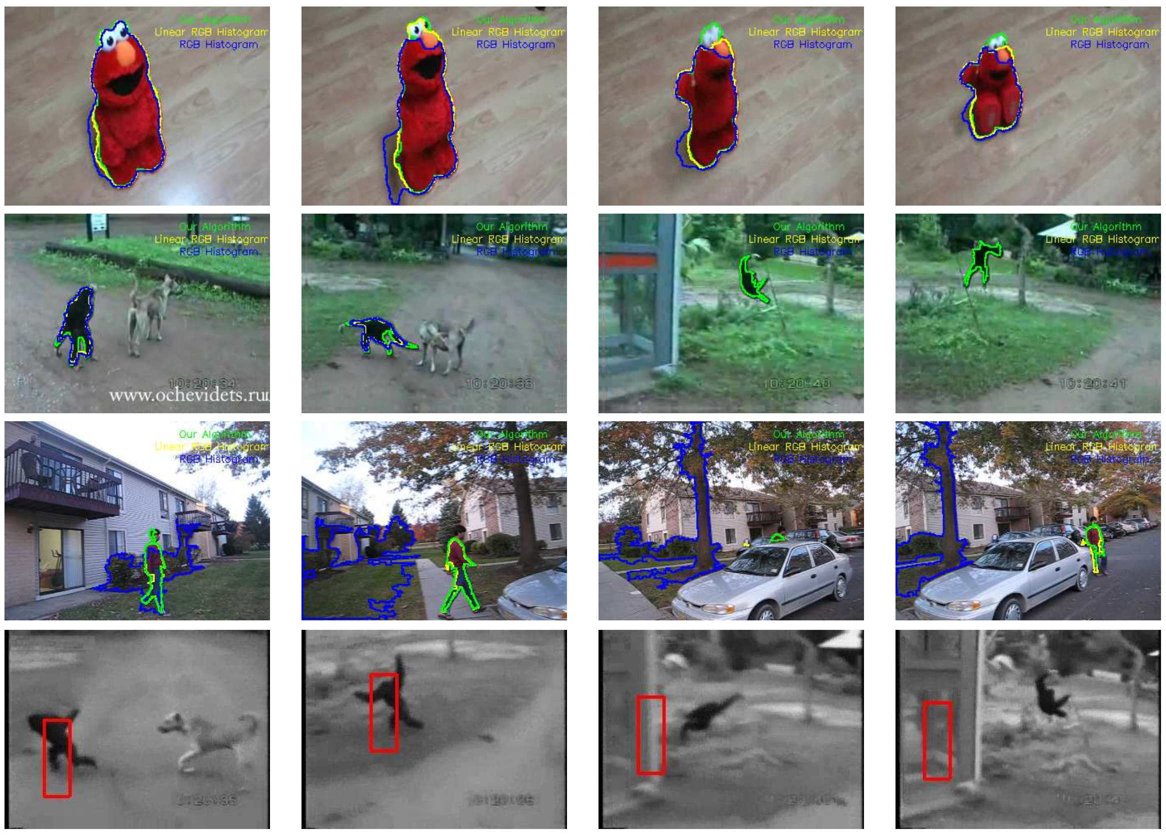


(b) Monkey



(c) Person

Figure 6.8: Normalized pixel classification error for (a) Elmo, (b) Monkey, and (c) Person sequence. Our algorithm outperforms implementations based upon Linear RGB Histogram [16] and color histogram [60, 62, 33, 50], showing the importance of spatial information for capturing accurate target representation. Motion marginally assists our algorithm, except when the drastic movement of the target (Monkey) causes the tracker to fail without it.



Frame 1 Frame 2 Frame 3 Frame 4

Figure 6.9: Contours obtained from our algorithm and the implementation based on Linear RGB Histogram [16] and standard RGB histogram [60, 62, 33, 50] are overlaid on the object. The blue contour represents the RGB histogram, the yellow one corresponds to the Linear RGB Histogram and the contour obtained from our algorithm is displayed in green color. The last row shows the results of FragTrack [1] on the monkey sequence.

Chapter 7

Conclusion and Discussion

A tracking algorithm based upon modeling the foreground and background regions with a mixture of Gaussians is presented. A simple and efficient region growing procedure to initialize the models is proposed, and comparisons with state-of-the-art segmentation algorithms are also presented. The segmentation algorithm is fast but it was found to be slightly sensitive to pre-defined thresholds. The GMMs are used to compute a strength image indicating the probability of any given pixel belonging to the foreground. Since GMMs are employed for object modeling, they capture all modes of the object and the scene, producing a better strength image than the ones obtained using linear classifiers. This strength image computation is embedded into the region growing formulation to evolve the contour of the object from the its position in the previous frame. The region growing model serves as a discrete implementation of level sets to evolve the contour. Since the formulation is based on only considering the neighbors of the evolving frontier, it is extremely faster than a traditional level set formulation. The appearance information of the foreground and background models are finally updated using the past and present tracked outputs to handle appearance changes in the object and also cases of self-occlusion. It can be noted that the spatial information is also part of the feature vector and hence it is essential to align the spatial coordinates before the strength image computation to avoid tracker failure in cases of large frame-to-frame motion. To achieve this purpose, joint feature tracking is used to update or pre-correct the spatial information. The joint feature tracking approach produces more reliable and smooth motion vectors than Lucas-Kanade method even in untextured areas. Extensive experimental results show that the resulting algorithm is able to compute accurate boundaries of multi-colored objects undergoing drastic shape changes and unpredictable motions on complex backgrounds. One of the main drawbacks of the approach is its heavy dependency on the performance of the segmentation algorithm in the initialization phase. An offline or online evaluation mechanism of the computed models for its separability of the given object from the background in the initial frame will alleviate such failures.

Possible future directions using the existing tracking framework are:

- Incorporating shape priors. From equation (2.1), we have that the probability of the contour at current frame involves the previously seen target information, background information and shape information.
- Utilizing the extracted shapes to learn more robust priors. Such learning mechanism will avoid tracker failure when the object comes in contact with background regions of similar appearance.
- Inducting an offline or online evaluation mechanism of the curved decision boundaries to get an optimal separating hypercurve that provides maximum separability for the detected object from the background. Such a sophisticated evaluation would reduce the over-dependency of the tracker on some fixed parameters to produce best results.
- Adding global information into the region segmentation process to reduce the sensitivity of the algorithm to pre-defined thresholds.
- Automating the object detection and initialization process.
- Handling cases of partial and complete occlusion using the extracted contours to avoid tracker failure during such scenarios.

Appendices

Appendix A Proof of main equation

Proof of (2.1) is as follows. From standard probability definitions, for any random variables a, b, and c:

$$\begin{aligned} p(a,b,c) &= p(a,b,c) && (1)\\ p(a|b,c)p(b,c) &= p(a,c|b)p(b) && (2)\\ p(a|b,c) &= \frac{p(a,c|b)p(b)}{p(b,c)} && (3) \end{aligned}$$

Using the above definition, we get

$$\begin{aligned} p(\Gamma_t|\Gamma_{0:t-1}, I_{0:t}) &= \frac{p(\Gamma_t, I_{0:t}|\Gamma_{0:t-1})p(\Gamma_{0:t-1})}{p(\Gamma_{0:t-1}, I_{0:t})} && (4)\\ &= \frac{p(\Gamma_t, I_{0:t}|\Gamma_{0:t-1})p(\Gamma_{0:t-1})}{p(I_{0:t}|\Gamma_{0:t-1})p(\Gamma_{0:t-1})} && (5)\\ &= \frac{p(\Gamma_t, I_{0:t}|\Gamma_{0:t-1})}{p(I_{0:t}|\Gamma_{0:t-1})} && (6)\\ &= \frac{p(I_{0:t}|\Gamma_{0:t})p(\Gamma_t|\Gamma_{0:t-1})}{p(I_{0:t}|\Gamma_{0:t-1})}, && (7) \end{aligned}$$

where the last equation is obtained using the definition:

$$p(a,b|c) = p(a|b,c)p(b|c) \qquad (8)$$

The assumptions are:

$$\begin{aligned} p(\Gamma_t|\Gamma_{0:t-1}) &= p(\Gamma_t|\Gamma_{t-1}) \qquad \text{(Markov)} && (9)\\ p(I_{0:t}|\Gamma_{0:t}) &= \prod_{i=1}^{t} p(I_i|\Gamma_i) \qquad \text{(independence)} && (10) \end{aligned}$$

Now, from (8), we have

$$\begin{aligned} p(I_{0:t}|\Gamma_{0:t-1}) &= p(I_t|I_{0:t-1}, \Gamma_{0:t-1})p(I_{0:t-1}|\Gamma_{0:t-1}) && (11)\\ &= p(I_t|I_{0:t-1}, \Gamma_{0:t-1})\prod_{i=1}^{t-1} p(I_i|\Gamma_i) \qquad \text{(using (10))} && (12) \end{aligned}$$

Plugging (9), (10) and (12) in (7), we get

$$\begin{aligned}
p(\Gamma_t|\Gamma_{0:t-1}, I_{0:t}) &= \frac{\prod_{i=1}^{t} p(I_i|\Gamma_i)p(\Gamma_t|\Gamma_{t-1})}{p(I_t|I_{0:t-1},\Gamma_{0:t-1})\prod_{i=1}^{t-1} p(I_i|\Gamma_i)} && (13)\\
&= \frac{p(I_t|\Gamma_t)p(\Gamma_t|\Gamma_{t-1})}{p(I_t|I_{0:t-1},\Gamma_{0:t-1})} && (14)\\
&\propto p(I_t|\Gamma_t)p(\Gamma_t|\Gamma_{t-1}) && (15)\\
&\propto p(I_t^+, I_t^-|\Gamma_t)p(\Gamma_t|\Gamma_{t-1}) && (16)\\
&\propto p(I_t^+|\Gamma_t)p(I_t^-|\Gamma_t)p(\Gamma_t|\Gamma_{t-1}) \qquad \text{(independence)} && (17)
\end{aligned}$$

where I_t^+ and I_t^- are the pixels inside and outside the contour respectively and they together constitute the image at time t.

Q.E.D.

Appendix B Computing summary statistics

Suppose we have a set of values $x_1, \ldots, x_N$. The mean and variance of the first $n \leq N$ values in the set are given by

$$
\begin{aligned}
\mu_n &= \sum_{i=1}^{n} x_i \\
&= \frac{S_n}{n} \\
\sigma_n^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_n)^2 \\
&= \frac{1}{n}\left[\sum_{i=1}^{n} x_i^2 - 2\mu_n \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} \mu_n^2\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{n} x_i^2 - 2\mu_n n \frac{1}{n}\sum_{i=1}^{n} x_i + \mu_n^2 \sum_{i=1}^{n} 1\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{n} x_i^2 - 2\mu_n n \mu_n + \mu_n^2 n\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{n} x_i^2 - 2n\mu_n^2 + n\mu_n^2\right] \\
&= \frac{1}{n}\left[\sum_{i=1}^{n} x_i^2 - n\mu_n^2\right] \\
&= \frac{B_n}{n} - \mu_n^2
\end{aligned}
$$

where

$$
\begin{aligned}
S_n &= \sum_{i=1}^{n} x_i \\
B_n &= \sum_{i=1}^{n} x_i^2
\end{aligned}
$$

Therefore, it is easy to compute the statistics at $n+1$ given n, S_n, B_n, and x_{n+1}, since $S_{n+1} = S_n + x_{n+1}$ and $B_{n+1} = B_n + x_{n+1}^2$.

Suppose we want to weight the values with an exponential decay. In the most general case, the values are not collected at equally spaced time intervals, yielding

$$\mu_n = \hat{S}_n / \hat{C}_n,$$

where

$$\begin{aligned}\hat{S}_n &= \sum_{i=1}^{n} x_i e^{-(t_n - t_i)} \\ \hat{C}_n &= \sum_{i=1}^{n} e^{-(t_n - t_i)}.\end{aligned}$$

Therefore, it is easy to compute the statistics at $n+1$ given n, $\hat{S}_n$, $\hat{C}_n$, and x_{n+1}, since $\hat{S}_{n+1} = \hat{S}_n e^{-(t_{n+1}-t_n)}$ and $\hat{C}_{n+1} = \hat{C}_n e^{-(t_{n+1}-t_n)}$.

With equally spaced time intervals, the equations become even simpler. Let $\alpha = e^{-(t_i - t_{i-1})}$ be a constant. Then

$$\mu_n = \hat{S}_n / \hat{C}_n,$$

where

$$\begin{aligned}\hat{S}_n &= \sum_{i=1}^{n} x_i (n-i)\alpha \\ \hat{C}_n &= \sum_{i=1}^{n} (n-i)\alpha.\end{aligned}$$

Therefore, it is easy to compute the statistics at $n+1$ given n, $\hat{S}_n$, $\hat{C}_n$, and x_{n+1}, since $\hat{S}_{n+1} = \hat{S}_n \alpha$ and $\hat{C}_{n+1} = \hat{C}_n \alpha$.

Bibliography

[1] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2006.

[2] Arasanathan Anjulan and Nishan Canagarajah. Object based video retrieval with local region tracking. *Signal Processing: Image Communication*, 22:607–621, August 2007.

[3] S. Avidan. Subset selection for efficient svm tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 85–92, June 2003.

[4] S. Avidan. Support vector tracking. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, pages 1064–1072, August 2004.

[5] Shai Avidan. Ensemble tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2005.

[6] M. Bertalmio, G. Sapiro, and G. Randall. Region tracking on level-sets methods. *IEEE Transactions on Medical Imaging*, 18:448–451, May 1999.

[7] Stanley T. Birchfield and Shrinivas J. Pundlik. Joint tracking of features and edges. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[8] Stanley T. Birchfield and Sriram Rangarajan. Spatiograms versus histograms for region-based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1158–1163, June 2005.

[9] G. R. Bradski. Computer vision face tracking as a component of a perceptual user interface. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, pages 214–219, October 1998.

[10] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):90–99, January 1986.

[11] Thomas Brox, Andrés Bruhn, and Joachim Weickert. Variational motion segmentation with level sets. In *Proceedings of the European Conference on Computer Vision*, pages 471–483, May 2006.

[12] A. D. Bue, D. Comaniciu, V. Ramesh, and C. Regazzoni. Smart cameras with real-time video object generation. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 429–432, June 2002.

[13] Yunqiang Chen, Yong Rui, and Thomas S. Huang. JPDAF based HMM for real-time contour tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 543–550, December 2001.

[14] Prakash Chockalingam, Nalin Pradeep, and Stan Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *Proceedings of the International Conference on Computer Vision*, October 2009.

[15] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. In *Proceedings of the IEEE*, volume 89, pages 1456–1477, October 2001.

[16] Robert Collins, Yanxi Liu, and Marius Leordeanu. On-line selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631 – 1643, October 2005.

[17] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[18] Daniel Cremers and Christoph Schnrr. Statistical shape knowledge in variational motion segmentation. *Image and Vision Computing*, 21:77–86, January 2003.

[19] A. P. Dempster, N. M. Laird, and D . B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. In *J. Roy. Statist. Soc*, pages 1–38, 1977.

[20] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 300–305, April 1998.

[21] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. In *Proceedings of the IEEE*, pages 1151–1163, July 2002.

[22] Wei Fan. Systematic data selection to mine concept-drifting data streams. In *International Conference on Knowledge Discovery and Data Mining*, pages 128–137, August 2004.

[23] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[24] V. Ferrari, T. Tuytelaars, and L. V. Gool. Real-time affine region tracking and coplanar grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 226–233, December 2001.

[25] Paul Fieguth and Demetri Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 21, June 1997.

[26] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 260–267, June 2006.

[27] Michael Grabner, Helmut Grabner, and Horst Bischof. Learning features for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[28] Hayit Greenspan, Jacob Goldberger, and Arnaldo Mayer. Probabilistic space-time video modeling via piecewise GMM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):384–396, March 2004.

[29] M. Greiffenhagen, D. Comaniciu, H. Niemann, and V. Ramesh. Design, analysis, and engineering of video monitoring systems: an approach and a case study. In *Proceedings of the IEEE*, volume 89, pages 1498–1517, October 2001.

[30] Jeffrey Ho, Kuang-Chih Lee, Ming-Hsuan Yang, and David Kriegman. Visual tracking using learned linear subspaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 782–789, June 2004.

[31] B. K. P. Horn and B G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(185):185–203, 1981.

[32] Michael Isard and Andrew Blake. Condensation conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, August 1998.

[33] Stéphanie Jehan-Besson, Michel Barlaud, Gilles Aubert, and Olivier Faugeras. Shape gradients for histogram segmentation using active contours. In *Proceedings of the International Conference on Computer Vision*, volume 1, pages 408–415, October 2003.

[34] Allan D. Jepson, David J. Fleet, and Thomas F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1296–1311, October 2003.

[35] Neeraj K. Kanhere, Shrinivas J. Pundlik, and Stanley T. Birchfield. Vehicle segmentation and tracking from a low-angle off-axis camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1152–1157, June 2005.

[36] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[37] Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 487–494, 2000.

[38] Maxime Lhuillier. Efficient dense matching for textured scenes using region growing. In *British Machine Vision Conference*, pages 700–709, 1998.

[39] David G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110, November 2004.

[40] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, April 1981.

[41] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[42] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):810–815, June 2004.

[43] U. Neumann and S. You. Natural feature tracking for augmented reality. In *IEEE Transactions on Multimedia*, volume 1, pages 53–64, March 1999.

[44] Stanley J. Osher and James A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, November 1988.

[45] G. P. Otto and T. K. W. Chau. A region-growing algorithm for matching of terrain images. *Image and Vision Computing*, 7(2):83–94, May 1989.

[46] Nikunj C. Oza and Stuart Russell. Online bagging and boosting. In *IEEE Transactions on Systems, Man, and Cybernetics*, pages 2340–2345, October 2005.

[47] Fatih Porikli, Oncel Tuzel, and Peter Meer. Covariance tracking using model update based on Lie algebra. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 728–735, June 2006.

[48] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[49] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[50] Y. Shi and W. C. Karl. Real-time tracking using level sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 34–41, June 2005.

[51] Kinh Tieu and Paul Viola. Boosting image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 228–235, June 2000.

[52] Andrea Vedaldi and Stefano Soatto. Local features, all grown up. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1753–1760, June 2006.

[53] N. Vlassis and A. Likas. A greedy EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, 15(1):77–87, February 2002.

[54] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, August 2003.

[55] Yichen Wei and Long Quan. Region-based progressive stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 106–113, June 2004.

[56] Gerhard Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. In *Machine Learning*, pages 69–101, April 1996.

[57] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19, pages 780–785, July 1997.

[58] Jun Wu, Xian-Sheng Hua, and Bo Zhang. Tracking concept drifting with Gaussian mixture model. *Visual Communications and Image Processing*, pages 1562–1570, July 2005.

[59] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory andits application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, November 1993.

[60] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, November 2004.

[61] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. In *IEEE Transactions on Computers*, pages 68–86, 1971.

[62] Tao Zhang and Daniel Freedman. Tracking objects using density matching and shape priors. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1056–1062, October 2003.

[63] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:884–900, September 1996.