ACCURATE TRACKING OF OBJECTS USING LEVEL SETS

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Masters
Computer Engineering

---

by
Nalin Pradeep Senthamil
August 2009

Accepted by:
Dr. Stan Birchfield, Committee Chair
Dr. Adam Hoover
Dr. Brian Dean

Abstract

Our current work presents an approach to tackle the challenging task of tracking objects in Internet videos taken from large web repositories such as YouTube. Such videos more often than not, are captured by users using their personal hand-held cameras and cellphones and hence suffer from problems such as poor quality, camera jitter and unconstrained lighting and environmental settings. Also, it has been observed that events being recorded by such videos usually contain objects moving in an unconstrained fashion. Hence, tracking objects in Internet videos is a very challenging task in the field of computer vision since there is no *a-priori* information about the types of objects we might encounter, their velocities while in motion or intrinsic camera parameters to estimate the location of object in each frame. Hence, in this setting it is clearly not possible to model objects as single homogenous distributions in feature space. The feature space itself cannot be fixed since different objects might be discriminable in different sub-spaces.

Keeping these challenges in mind, in the current proposed technique, each object is divided into multiple fragments or regions and each fragment is represented in Gaussian Mixture model (GMM) in a joint feature-spatial space. Each fragment is automatically selected from the image data by adapting to image statistics using a segmentation technique. We introduce the concept of *strength map* which represents a probability distribution of the image statistics and is used to detecting the object. We extend our goal of tracking object to tracking them with accurate boundaries thereby making the current task more challeng-

ing. We solve this problem by modeling the object using a level sets framework, which helps in preserving accurate boundaries of the object and as well in modeling the target object and background. These extracted object boundaries are learned dynamically over time, enabling object tracking even during occlusion. Our proposed algorithm performs significantly better than any of the existing object modeling techniques. Experimental results have been shown in support of this claim. Apart from tracking, the present algorithm can also be applied to different scenarios. One such application is contour-based object detection. Also, the idea of *strength map* was successfully applied to track objects such as vessels and vehicles on a wide range of videos, as a part of the summer internship program.

Dedication

I dedicate this thesis work to my parents, my wife, my family and my friends who have provided great support throughout.

Acknowledgments

I'm very grateful to my advisor Dr Stan Birchfield for his continued guidance throughout my stay at Clemson and helping me in understand computer vision better. Many thanks to Dr Brian Dean for his course on Advanced Data Structures that helped me in understanding its techniques to apply in computer vision and image processing. Also, thanks to Dr Adam Hoover for being part of my committee and for helpful suggestions.

Thanks to Prakash for his help during my visits to Princeton. Finally, special thanks to my wife Manika for her love and complete support during my masters.

Table of Contents

List of Figures

Chapter 1

Introduction

In the computer vision world, locating an object of interest in every frame of a video is a big challenge. An object can be located in two ways: either by first, segmenting each frame into regions and consequently identifying the object or second, by tracking the object using cues from the previous frame. Continuous tracking of an object in a video with cues from previous frames suffers from difficulties such as when the scene undergoes a change or the object changes appearance, non-rigid objects suffer abrupt motion, cases of object occlusion as well as arbitrary camera motion. Numerous techniques have been proposed in literature to overcome these difficulties and persistently track the objects. Persistent object tracking has been shown to be successful in different scenarios such as traffic monitoring, video surveillance, gesture recognition, object tagging, vehicle navigation and many more. Depending on the particular application scenario, certain constraints are added to object tracking in order to efficiently solve parameters of object motion, feature representation and background scene models. Recent research has targeted on making object tracking more generalized as opposed to specific application scenarios.

With more than 65,000 videos being uploaded by users on YouTube alone everyday, there is a massive repository of videos lying around in the Internet. Statistics reveal that a

large majority of these videos are personally shot by individual users using devices such as their cellphones and digital cameras. These videos usually are captured at poor resolutions and have unconstrained camera motion, environment settings and object characteristics. Due to the abundance of this type of data on the Internet, there has been a considerable shift in the focus of computer vision community to analyze these videos for applications such as tracking, fast image and video retrieval, video categorization and object recognition. On the same lines, in the present work, the focus is mainly on development of research that can be applied to internet videos or videos recorded by hand-held cameras and are subjected to lot of noise and disturbance. With this objective, we have employed a contour-based method for accurately tracking non-rigid deforming objects by representing them as a set of multiple Gaussians.

In the recent research, object tracking has been formulated as a classification problem in which probability of each pixel belonging to the target in the current frame is computed. Based on the probability estimates obtained for each pixel, a corresponding object region is identified and evolved using contours. Recently proposed techniques [2, 19, 20, 13] perform significantly well, but have certain limitations as elaborated below.

- Each object is modeled as being unimodal.
- Based on probability estimates, shape information of object is not incorporated.
- Spatial information capturing joint probability of features is ignored.
- Complete object occlusion is not addressed in detail.

In this work, part of which is published in ICCV [33], each object model is split automatically into different regions or fragments using a simple segmentation technique. By splitting the object into multiple regions enables the method to preserve the spatial relationship of pixels in each fragment with respect to each other. This way, each fragment

region after the split, can be represented as a single mode of distribution. Once the fragments are selected, they are represented in a level-set framework that enables tracking to be formulated in a Bayesian manner. The level-set framework used is similar to Chan-Vese [12] model, where gradients are not necessary in order to identify object boundaries. The object boundaries are localized instead, using segmentation of the object from its background. In this work, it is referred to as the *Strength Image* which is obtained based on the pixel probability measure. Further, usage of level-sets enables handling of topological changes in the object fragments such as splitting and merging.

The contour obtained through level-sets are evolved and used as a prior for evolving the contour in subsequent frames. For objects where motion is drastic between successive frames, global Joint KLT [7] is used to employ a global smoothness term to calculate sparse motion vectors for each fragment. Based on these motion vectors, for arbitrary moving objects where contours do not overlap between subsequent frames, contour evolution is re-initialized and helps in keeping track of object more closely. In scenarios where object is completely occluded, dynamically all previously learnt object shapes are retrieved and hallucinated during occlusion periods. The proposed approach has been tested on many challenging videos, as demonstrated in the later chapters, and found the performance of the proposed tracker better than existing contour based trackers that model object appearance.

The rest of the text is organized as follows. Chapter 2 gives a brief description on overview of object tracking. In Chapter 3, the proposed tracking approach is explained detailing about object representation using Gaussian Mixture Model, Strength Image computation, and level-set formulation on the strength image along with need for modeling fragment motion. Experimental results, complete occlusion handling on video sequences and comparison chart with other algorithms are presented in Chapter 4. In Chapter 5, application of object tracking for a specific problem is explained and final conclusions are presented in Chapter 6.

Chapter 2

Object Tracking - Overview

The purpose of an object tracker is to generate the trajectory of an object over time by locating its position in every frame of the video. An object tracker may return this information as a pointer to a region in the image which is being wholly or partially encompassed by the object of interest. The tracking process employs all known occurrences of the object in the previous frames in order to determine it's current location in the present frame. Hence, for successful object tracking the bounding region around the object must be continuously updated both in terms of its size as well as location. The model that is used to represent the object best determines the type of motion or deformation it can undergo. Rectangular shapes only allow translational movement, elliptical shapes allow translational and rotational changes on rigid objects, while a contour representation allows non-rigid objects to undergo complete deformations and still provide a descriptive information.

Based on the literature, object tracking can broadly be classified into three major categories: (a) Point Tracking, (b) Kernel Tracking, and (c) Contour Tracking.

Point tracking is formulated as a method that establishes correspondences between detections of an object in multiple frames using points. Point tracking becomes complicated during the presence of occlusion and merging of objects where noisy observations

are induced. Point tracking is primarily carried out either deterministically or as a statistical approach. In the deterministic approach, motion-based constraints are addressed in terms of common motion or uniform proximity [37]. While in the statistical method, noise is explicitly handled by taking model uncertainties. Since noise parameters are usually unknown, this method makes certain assumptions such as the noise being Gaussian, Kalman filters provide optimal solutions [10, 5]. In scenarios where noise is not Gaussian, particle filter techniques have been introduced to help in solving estimation problem of tracking [28].

Kernel tracking generally refers to object's shape, motion and appearance. As said earlier, objects can be modeled as rectangular or elliptical and their associated kernel density histograms are computed to track them across frames. The motion undergone can be translation, rotation or in general, affine. This tracking methodology is generally referred as appearance model-based tracking. In this technique, both online appearance building and offline appearance modeling are done. Usage of histograms, templates and covariances can help in online object modeling and learning [14, 13, 30], whereas multiple view based appearances can be modeled offline using Principal Component Analysis (PCA) and Support Vector Machines (SVM) [1, 8].

Contour based tracking is applied primarily to objects having complex shapes like human body, monkey, hand and more. This technique provides accurate shape description of the object and can be applied for shape matching and as well tracking. Contour tracking iteratively evolves from previous initial contour frame to a new position in the current frame. Tracking using contours was initially carried out with the help of state space models like particle filters [28] and more recently using contour energy functionals such as level sets and snakes [38, 15]. Some contour trackers only use boundary information whereas others use complete regions inside contours for updating the appearance information of the object.

Chapter 3

Tracking Approach

The goal of the approach being presented is to track continuously hand marked non-rigid objects from videos by fitting a contour. To accurately fit the contour, multi-modality of the object and background is utilized. Each pixel from an image frame contributes both its spatial and RGB element vector, using which the object is adaptively fragmented and modeled as mixture of Gaussians. Although other features like texture bank or edge gradients could be used, we restricted to only spatial and color features in this approach. We classify each individual pixel to either belong to foreground or background based on probability measure computed with respect to fragment models of object and background. We introduce this probability measure as *strength image*. Further to accurately fit tight boundary around the object, we utilized implicit representation of contours using level sets based on Chan-Vese approach [12] for their numeric stability and accuracy in continuously tracking the object. Also, during continuous tracking due to drastic motion of these non-rigid objects, there is a possibility of tracker to drift. To address this problem, we incorporate joint feature tracking approach proposed by Birchfield et al.[7] to align the coordinate systems of model fragments and target object. The next few Sections introduce this approach in detail.

3.1 Adaptive Fragmentation

We first do an automatic adaptive fragmentation of both the foreground (object) and background region based on the user provided mask in the first frame. We do the adaptive fragmentation to identify the different Gaussian models present in both the object and background. On the contrary, we could have also modeled in simpler terms by representing both the object and background as two different Gaussian ellipsoids rather than fragmenting, but this approach does not solve the subtle complexities present in multi-modal object. Hence, we fragment both object and background into multiple mixture of Gaussians combining spatial and feature information.

3.1.1 Region Growing mechanism

In order to adaptively fragment the object and background into multiple fragments, we employ a simple region growing mechanism. Through this mechanism, the fragments get determined automatically and divides the object and background region into multiple fragments based on image statistical data like mean and covariance. Let $\mu_1^+, \ldots, \mu_{p_+}^+$, $\Sigma_1^+, \ldots, \Sigma_{p_+}^+$, $\mu_1^-, \ldots, \mu_{q_-}^-$, $\Sigma_1^-, \ldots, \Sigma_{q_-}^-$ denote the mean and covariance for $\{+\}$ foreground and $\{-\}$ background with p and q fragments.

Based on the user provided mask, a pixel position is chosen randomly that belong to the object. From this start pixel position, additional neighboring pixels are added to it to form a fragment. Additional neighboring pixels are added only if they are within certain τ standard deviation of Gaussian Model of current fragment. Also, as every new neighboring pixel gets added to the current fragment, its mean and covariance parameters are updated accordingly such that running values are maintained. If there are no more neighboring pixels to add to the current fragment, then new fragment model is initialized from this pixel as starting point. This process is repeated until entire object is flooded with multiple frag-

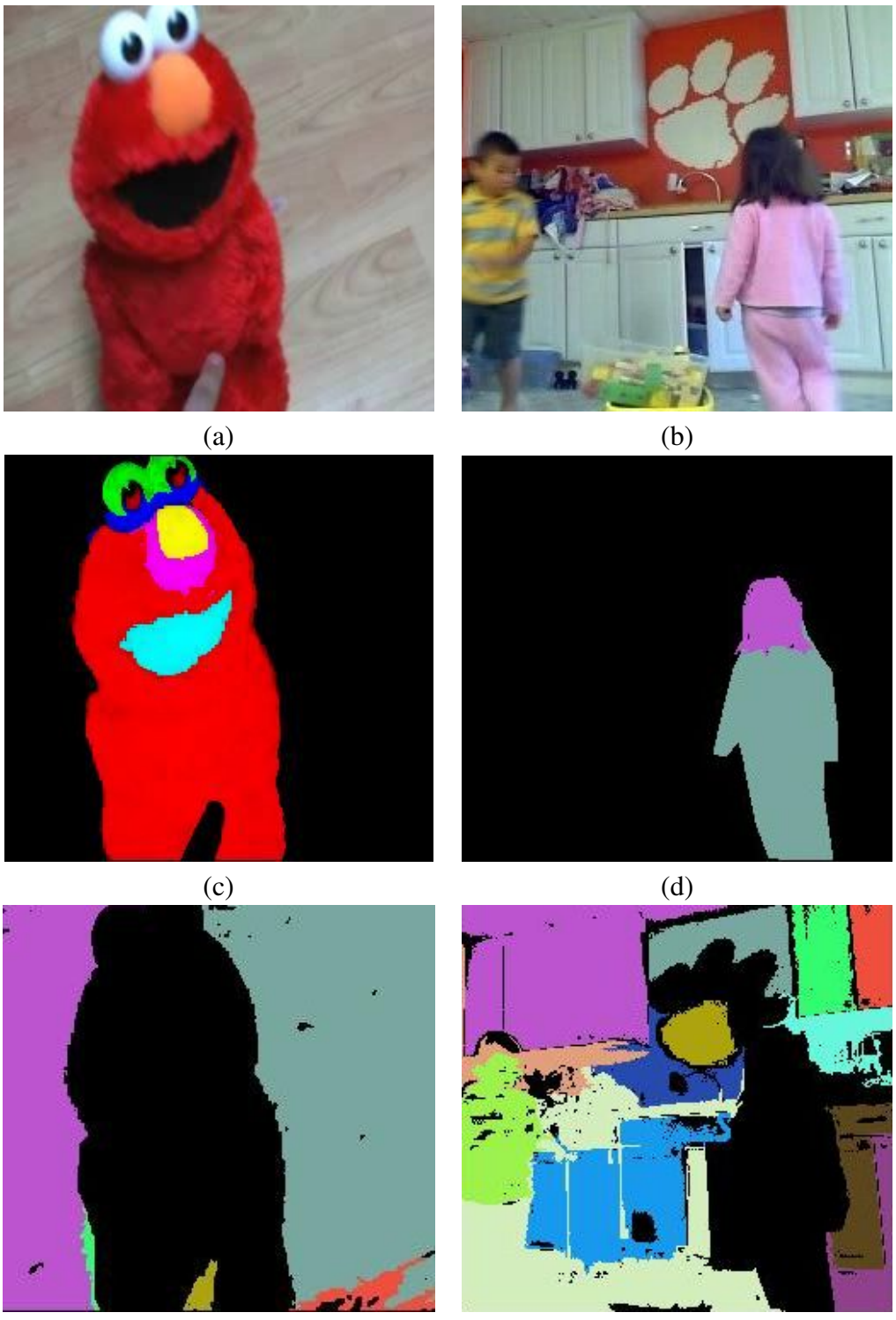


Figure 3.1: (a)-(b) Original images of Elmo and Girl, (c)-(d) Foreground regions, and (e)-(f) background fragmented regions obtained by region growing mechanism.

ments. This same procedure is repeated for background as well, and it gets modeled into multiple fragments. The fragments belonging to the object form foreground and the others form background. The number of fragments estimated for foreground and background would be in varying number and in varying size. An example output of foreground and background segmented fragments using this region growing procedure is shown for Elmo Doll and Girl frame in Figure 3.1. This region growing mechanism is inspired by the work on Spatially Variant Finite Mixture Models [34, 35].

3.1.2 Strength Image Computation

The fragments determined from the above explained procedure is modeled using the parameters mean and covariance. Each fragment is represented in separate Gaussian ellipsoid in joint feature-spatial space similar to [21].

Let $I_t : \mathbf{x} \rightarrow \Re^m$ be the image at time t that maps a pixel $\mathbf{x} = [\, x \quad y \,]^T \in \Re^2$ to a three element RGB value (m=3). Let each individual pixel $\mathbf{y} = [\, \mathbf{x}^T \quad I(\mathbf{x})^T \,]^T$ be represented as vector containing the pixel coordinates and its RGB image measurements. Let the closed curve bounding the target be represented as Γ at time t. The likelihood of this individual pixel is then given by a Gaussian mixture model (GMM):

$$p_\star(\mathbf{y}|\Gamma_t) = \sum_{j=1}^{k_\star} \pi_j p_\star(\mathbf{y}|\Gamma_t, j), \tag{3.1}$$

where $\star \in \{-, +\}$ and p_+ captures the likelihood of pixels inside Γ_t and p_- captures likelihood of pixels outside Γ_t. Also, let $\pi_j = p(j|\Gamma_t)$ is the probability that the pixel was drawn from the jth fragment, $k_\star$ is the number of fragments in the target or background, $\sum_{j=1}^{k_\star} \pi_j = 1$. Hence,

$$p_\star(\mathbf{y}|\Gamma_t, j) = \eta \exp\left\{-\frac{1}{2}(\mathbf{y}-\mu_j^\star)^T \left(\Sigma_j^\star\right)^{-1}(\mathbf{y}-\mu_j^\star)\right\}, \tag{3.2}$$

where $\mu_j^\star \in \Re^n$ is the mean and $\Sigma_j^\star$ the $n \times n$ covariance matrix of the jth fragment in the target or background model (depending upon $\star$), and η is the Gaussian normalization constant. The $n \times n$ covariance matrix of each fragment is constructed taking the pixel spatial and RGB features.

Based upon the likelihood values calculated for each pixel, a probability map termed as *strength image* is computed. It is computed by considering the problem as binary classification problem between target object and the background. Similar approach is followed by few recent works [2, 19, 20] as well. Strength value at each pixel location in the image is computed by taking log ratio of probabilities of the foreground and the background region. Positive values obtained by the *strength image* calculation indicate the pixel belonging to foreground region and negative values indicate it belonging to background region.

$$S(\mathbf{x}) = \log\left(\frac{p_+(\mathbf{x})}{p_-(\mathbf{x})}\right) = \Psi^-(\mathbf{x}) - \Psi^+(\mathbf{x}), \tag{3.3}$$

where $\Psi^\star(\mathbf{x}) = -\log p_\star(\mathbf{x})$.

An example *strength image* obtained for Elmo doll and Girl is shown in Figure 3.2. The strength image for it is calculated by considering the independent fragment models obtained from region growing mechanism. Our goal is to estimate the contour and in the next Section 3.2 we introduce the formulation of level sets on the computed *strength image*.

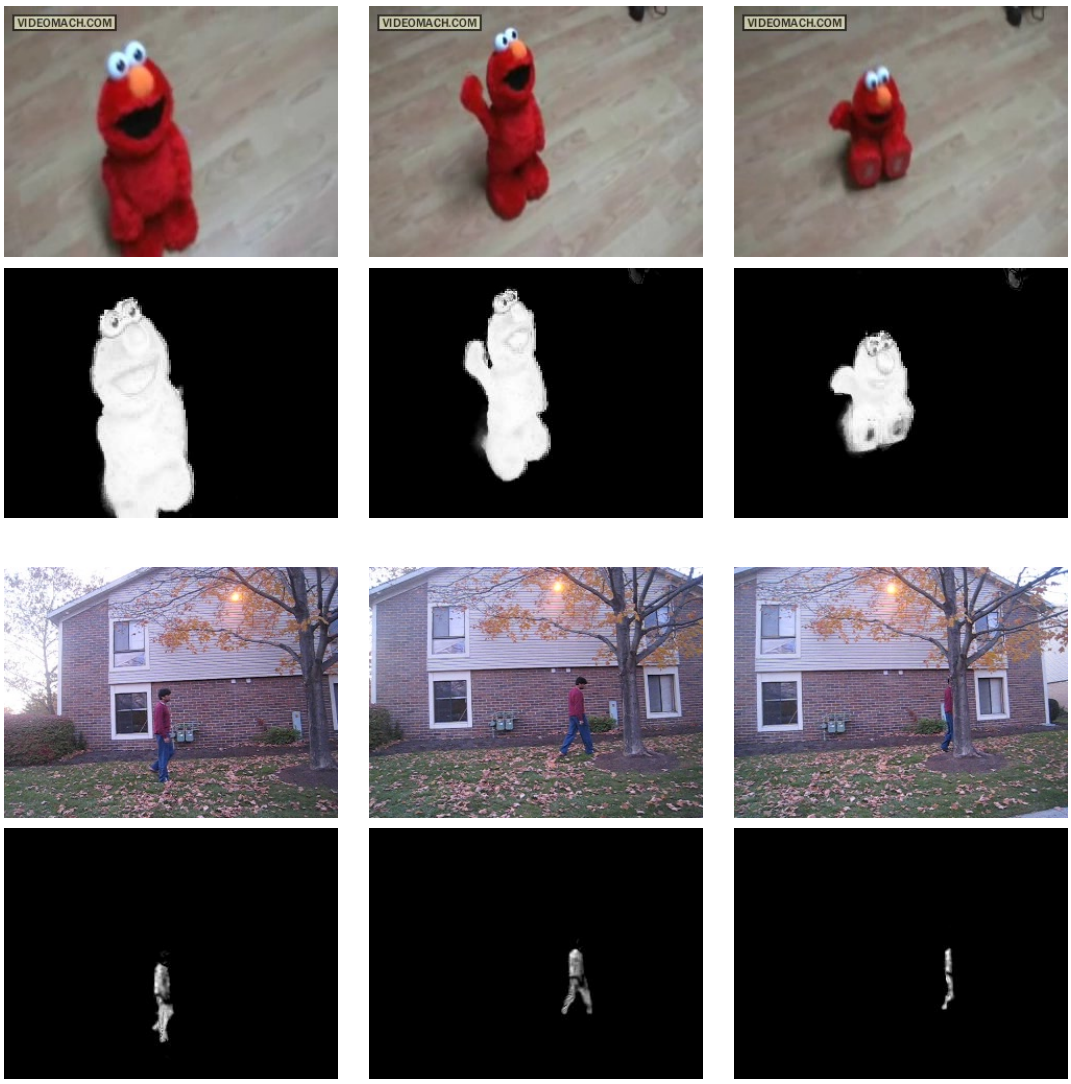


Figure 3.2: The first and third row shows original images of Elmo and Walking person. The second and fourth row shows the corresponding *strength map* obtained.

3.2 Contour Framework

In an active contour framework, segmentation or tracking of an object is achieved by evolving a closed contour to the object boundary, such that the contour tightly encloses the object region. Evolution of the contour is governed by an energy functional which defines the fitness of contour to the object region. This energy functional primarily includes image based energy term which is defined either locally or globally using gradients, color and texture.

Kass et al. [29] used local information in the form of image gradient to define the image energy term for contour evolution. However, usage of image gradients provide local information and are sensitive to local minima during the contour evolution. Further, in these approaches, the contour is initialized typically outside the object and shrunk until its boundary is encountered. In order to overcome these problems, researchers introduced region-based image energy terms [12, 17] where the contour can grow inwards or outwards. Apart from identifying the contour evolution mechanism, there is also challenge in selecting the right contour representation, either explicitly using snakes [29, 9] or implicitly using level sets [38, 17]. The most important advantage to model implicitly using level sets is its flexibility in allowing topological changes (split and merge).

To solve our purpose of tracking non-rigid objects where each fragment dynamically deforms, we utilized region-based approach defined by the *strength image* from mixture of fragment Gaussians implicitly using level sets. We adopted the Chan-Vese [12] approach in formulating the level set model. The next section introduces the formulation.

3.2.1 Level set formulation

In general Level Set Method (LSM), a 2D deformable contours in a 2D image are iterated over time to minimize a energy functional. They can be better viewed as 3D level

set function $\phi(x, y, t)$, where t represents the iterations of the level set contour. The zeroth level set which is $\phi(x, y, t) = 0$ represents a 2D contour Γ on the 2D image at any given time. Figure 3.3 shows an example relationship of boundary of an object in an image, level set function ϕ, and contour Γ. The initial value of ϕ is chosen arbitrarily surrounding the target.

The level set function ϕ has the property of $\phi > 0$ for regions R^- inside the curve Γ and $\phi < 0$ for regions R^+ outside the curve Γ. And, the evolution of the level set function ϕ from initial contour is defined to minimize the following energy functional:

$$E(\phi) = \int_{R^+} \Psi^+(\mathbf{x}) d\mathbf{x} + \int_{R^-} \Psi^-(\mathbf{x}) d\mathbf{x} + \mu \ell(\Gamma), \tag{3.4}$$

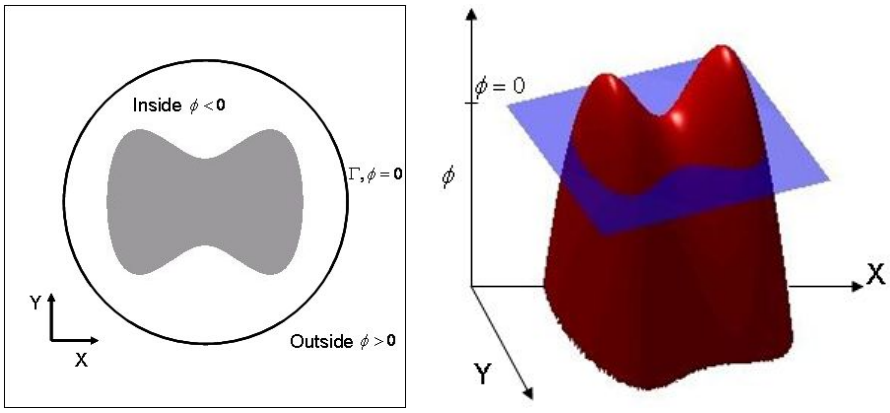
where μ is a scalar that weights the relative importance of the shape term, which is assumed for the moment to consist only in measuring $\ell(\Gamma)$, the length of the curve. $\Psi^+(\mathbf{x})$ and $\Psi^-(\mathbf{x})$ are the strength (probability) of each pixel to belong to target or background as explained earlier. At this point, another term is introduced - regularized Heaviside function $H(z) = \frac{1}{1+e^{-z}}$ as a differentiable threshold operator to rewrite the above as

$$E(\phi) = \int_{\Omega} H(\phi)\Psi^+(\mathbf{x}) + (1 - H(\phi))\Psi^-(\mathbf{x}) + \mu |\nabla H(\phi)| d\mathbf{x}, \tag{3.5}$$

where $\ell(\Gamma) = \int_{\Omega} |\nabla H(\phi)| d\mathbf{x}$, and $\Omega = R^+ \cup R^-$ is the image domain. With $E = \int_{\Omega} F(x, y, \phi, \phi_x, \phi_y) d\mathbf{x}$, the associated Euler-Lagrange equation is given by

$$\begin{aligned} 0 &= \frac{\partial F}{\partial \phi} - \frac{\partial}{\partial x}\left[\frac{\partial F}{\partial \phi_x}\right] - \frac{\partial}{\partial y}\left[\frac{\partial F}{\partial \phi_y}\right] \\ &= h(\phi)\left(\Psi^+(\mathbf{x}) - \Psi^-(\mathbf{x}) - \mu \text{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right)\right), \end{aligned}$$

where $\phi_x = \partial\phi/\partial x$, $\phi_y = \partial\phi/\partial y$, $h(\phi) = \partial H/\partial\phi$, $\nabla\phi = [\, \phi_x \quad \phi_y \,]^T$ is the gradient of ϕ,



(a) (b)

Figure 3.3: (a) A relationship among the boundary of an object in a image, level set function ϕ and contour Γ. (b) A sample shape of level set function ϕ whose zero level set corresponds to contour Γ.

and div is the divergence operator. To avoid the difficulty of solving this PDE explicitly for ϕ, we instead take the value on the right-hand side as an indication of the error, and apply gradient descent iterations with

$$\phi^{(k+1)} = \phi^{(k)} + |\nabla\phi| \left(\Psi^{-}(\mathbf{x}) - \Psi^{+}(\mathbf{x}) + \mu \text{div} \left(\frac{\nabla\phi}{|\nabla\phi|} \right) \right), \tag{3.6}$$

where k is the iteration number, and we have used the approximation $h(\phi) \approx |\nabla\phi|$, which is accurate as long as the level set function is smooth away from the boundary. The sign in the equation comes from the convention that $\phi > 0$ inside the boundary.

This method of modeling the foreground and background regions explicitly like [12] results in a large basin of attraction, so that the iterations above will converge to the target from a wide variety of initial curves, without being significantly distracted by local noise in the data. Also, since the curve evolution follows region-based image energy, the initial curve may be inside the target, outside the target, or combination of the two. Figure 3.4 shows for the Elmo doll, different stages of curve evolution modeled by the level set function in the first frame.

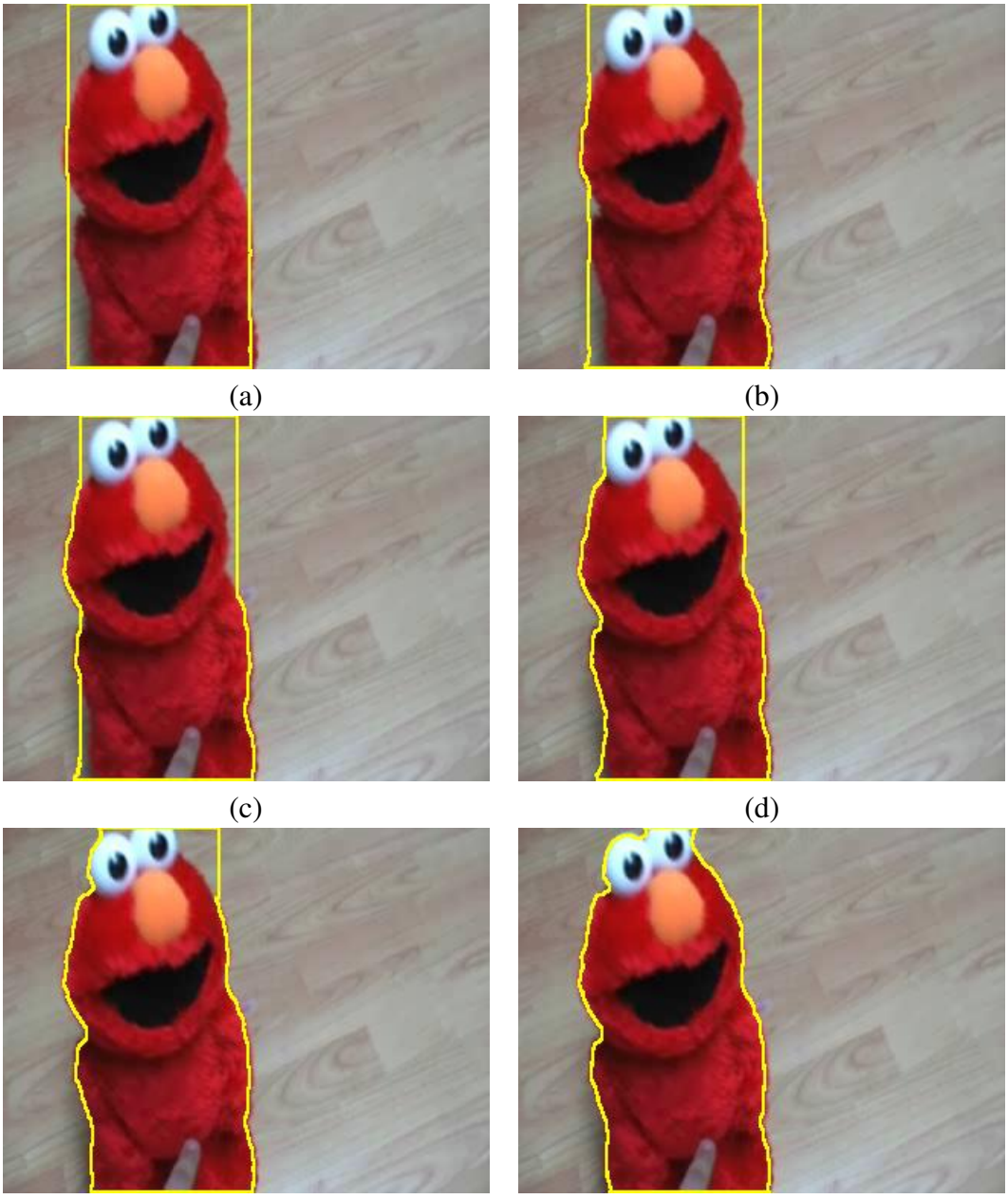


Figure 3.4: Level Set Evolution for Elmo Doll. Starting from top left in lexicographic order different time stamps of convergence of contour Γ is shown. The frames are extracted at time stamps $t = 0, 10, 22, 35, 56, 90$.

3.3 Fragment Motion

While the minimization explained in Section 3.2 is not extremely sensitive to the initial contour, nevertheless it is beneficial for the coordinate systems of the target and the model fragments to be approximately aligned. Such alignment increases the accuracy of the strength image and avoids tracker from drift under drastic motion, due to the use of spatial information in the joint spatial-feature vectors. As a result it creates a need to identify, *prior* to computing the strength image, approximate motion vectors between the previous and current image frame for each fragment: $\mathbf{u}_i^\star = (u_i^\star, v_i^\star), i = 1, \ldots, k^\star$.

We utilize the recent joint feature tracking approach proposed by Birchfield et al.[7] for identifying the motion vectors rather than the traditional motion algorithms like Horn-Schunck (dense motion) or Lucas-Kanade (sparse motion). The traditional 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 thesis work. Moreover, dense motion computation wastes precious resources for this application, since we only need approximate alignment between the fragments. In a similar manner, sparse feature tracking algorithms are not suitable for recovering the motions of the individual fragments due to their sparse nature. These traditional techniques handle features independently and often yield some percentage of unreliable estimates. In order to overcome this dilemma, this approach of joint feature tracking [7] as explained below is utilized. An example output is shown in Figure 3.5.

3.3.1 Joint-KLT Motion vectors

Differential methods for both dense optical flow as well as sparse feature tracking are based on the assumption that the intensity values of the projection of scene points do

not change over time, as explained by Bruhn et al [11].

$$I(x+u, y+v, t+1) = I(x, y, t), \tag{3.7}$$

where $I(x, y, t)$ is the intensity of pixel $\mathbf{x} = (x, y)^T$ in frame t, and $\mathbf{u} = (u, v)^T$ is the displacement of the pixel between consecutive frames t and $t+1$. For small displacements, a linearized Taylor series expansion yields the well-known *optic flow constraint equation*:

$$f(u, v; I) = I_x u + I_y v + I_t = 0, \tag{3.8}$$

where the subscripts denote partial derivatives. The well-known *aperture problem* arises because this single equation is insufficient to recover the two unknowns u and v.

Two approaches have been developed to overcome this which are traditional motion estimation techniques : Lucas-Kanade (sparse feature based) and Horn-Schunck (dense motion based).

The Lucas-Kanade [30] approach to overcoming the aperture problem assumes that the unknown displacement **u** of a pixel is constant within some neighborhood. As a result, the displacement can be computed by minimizing

$$E_{LK}(u, v) = K_\rho * \left((f(u, v; I))^2 \right), \tag{3.9}$$

where $K_\rho * (\cdot)$ denotes convolution with an integration window of size ρ. Differentiating with respect to u and v, and setting the partial derivatives to zero, yields the linear system

$$\begin{bmatrix} K_\rho * (I_x^2) & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & K_\rho * (I_y^2) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} K_\rho * (I_x I_t) \\ K_\rho * (I_y I_t) \end{bmatrix} \tag{3.10}$$

which is solved iteratively to minimize E_{LK}.

Alternatively, the Horn-Schunck [24] approach regularizes the under-constrained optic flow constraint equation by imposing a global smoothness term. While Lucas-Kanade finds the displacement of a small window around a single pixel, Horn-Schunck computes the global displacement functions $u(x, y)$ and $v(x, y)$ by minimizing

$$E_{HS}(u, v) = \int_{\Omega} \left(f(u, v; I)\right)^2 + \lambda \left(|\nabla u|^2 + |\nabla v|^2\right) dx\, dy, \tag{3.11}$$

where λ is the regularization parameter and Ω is the domain of the image. The minimum of this functional is found by solving the corresponding Euler-Lagrange equations, leading to

$$\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \lambda \nabla^2 u - I_x I_t \\ \lambda \nabla^2 v - I_y I_t \end{bmatrix}, \tag{3.12}$$

where $\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ and $\nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}$ are the Laplacian of u and v, respectively. Solving this equation for u and v and using the approximation that $\nabla^2 u \approx h(\bar{u} - u)$, where $\bar{u}$ is the average of the values of u among the neighbors of the pixel, and h is a constant scale factor, we get

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \bar{u} \\ \bar{v} \end{bmatrix} - \frac{I_x \bar{u} + I_y \bar{v} + I_t}{h\lambda + I_x^2 + I_y^2} \begin{bmatrix} I_x \\ I_y \end{bmatrix}. \tag{3.13}$$

Thus, the sparse linear system can be solved using the Jacobi method with iterations for pixel $(i, j)^T$ of the form:

$$\begin{aligned} u_{ij}^{(k+1)} &= \bar{u}_{ij}^{(k)} - \gamma I_x \\ v_{ij}^{(k+1)} &= \bar{v}_{ij}^{(k)} - \gamma I_y, \end{aligned} \tag{3.14}$$

where

$$\gamma = \frac{I_x \bar{u}_{ij}^{(k)} + I_y \bar{v}_{ij}^{(k)} + I_t}{h\lambda + I_x^2 + I_y^2}. \tag{3.15}$$

It is important to note that, although the derivation of Eq. (3.12) assumes a con-

tinuous formulation, the final result in Eqs. (3.14)–(3.15) corresponds to a discrete energy functional, due to the discrete approximation of the Laplacian. This observation led to combining the Lucas-Kanade and Horn-Schunck approaches as explained by [7] in Eqs. (3.9) and (3.11) into the following functional to be minimized:

$$E_{JLK} = \sum_{i=1}^{N} (E_D(i) + \lambda_i E_S(i)), \tag{3.16}$$

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

$$E_D(i) = K_\rho * \left((f(u_i, v_i; I))^2 \right) \tag{3.17}$$

$$E_S(i) = \left((u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2 \right). \tag{3.18}$$

In these equations, the energy of feature i is determined by how well its displacement $(u_i, v_i)^T$ matches the local image data, as well as how far the displacement deviates from the expected displacement $(\hat{u}_i, \hat{v}_i)^T$.

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

$$Z_i \mathbf{u}_i = \mathbf{e}_i, \tag{3.19}$$

where

$$Z_i = \begin{bmatrix} \lambda_i + K_\rho * (I_x I_x) & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & \lambda_i + K_\rho * (I_y I_y) \end{bmatrix}$$

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

(a) (b)

Figure 3.5: (a) Elmo image shown with Lucas-Kanade colored vectors and (b) shows image with Joint-Lucas-Kanade motion vectors. The vectors are colored differently based on each fragment.It can be seen from the figure that Joint-Lucas-Kanade outputs produce smoother motion vectors.

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

$$\tilde{u}_i^{(k+1)} = \hat{u}_i^{(k)} - \frac{J_{xx}\hat{u}_i^{(k)} + J_{xy}\hat{v}_i^{(k)} + J_{xt}}{\lambda_i + J_{xx} + J_{yy}} \tag{3.20}$$

$$\tilde{v}_i^{(k+1)} = \hat{v}_i^{(k)} - \frac{J_{xy}\hat{u}_i^{(k)} + J_{yy}\hat{v}_i^{(k)} + J_{yt}}{\lambda_i + J_{xx} + J_{yy}}, \tag{3.21}$$

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 $J_{yt} = K_\rho * (I_y I_t)$.

Usage of Gauss-Seidel iterations leads to increased convergence so that $\hat{u}_i^{(k)}$ and $\hat{v}_i^{(k)}$ are actually computed using a mixture of values from the kth and $(k+1)$th iterations (depending upon the order in which the values are updated), and by performing a weighted average of the most recent estimate and the new estimate.

Once the N features have been tracked, the motion vector of each fragment $\mathbf{u}_i^\star$ is computed by averaging the motions of the features within the fragment. Based on the motion vector identified for each fragment, the computation of *strength image* location in next frame is aligned with fragment models. 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.

3.4 Shape Matching

Shape matching is performed similar to tracking based on template matching, where an object silhouette (contour) and its associated model is matched in the current frame. The search is performed by computing the similarity of the object with shapes learnt online in previous frames or with the template model from the database. In [26] Huttenlocher et al. performed shape matching using an edge-based representation. The authors use the Hausdorff distance to construct a correlation surface from which the minimum is selected as the new object position. On similar lines, in this proposed approach, Hausdorff distance metric is used as a measure to identify the best object shapes matching the 2D contour points. The Hausdorff metric is a mathematical measure for comparing two sets of points $A = a1, a2, , an$ and $B = b1, b2, , bm$ in terms of the least similar members [23]:

$$H(A, B) = \max_{a \in A}(\min_{b \in B}(d(a, b))) \tag{3.22}$$

where a and b are points of set A and B respectively, and $d(a, b)$ is distance metric between the two points. This shape matching technique is utilized in this proposed approach during complete object occlusion and as well to dynamically recognize objects. Section 4.2 explains the application of shape matching technique during complete object occlusion and Section 4.4 explains its application for recognizing objects.

Chapter 4

Experimental Results

The proposed algorithm for accurate object tracking with contours is implemented in Microsoft Visual C++. The algorithm runs at 6-10 frames per second on Intel Duo-core processor with 1GB RAM and 1.8 GHz processor speed. The variation in running time occurs due to size of image and object being tracked. Four sequences were chosen to demonstrate and evaluate the tracker where objects undergo significant scale changes, lot of deformations, and unpredictable fast motion. These sequence were either obtained from internet video sites or captured using hand held camera. The contours were accurately tracked most of time, except when object occludes where dynamically learnt prior shape is hallucinated. In all the sequences, the object was handmarked at first frame to start tracking.

4.1 Single Object Tracking

The first sequence is shown for Tickle Me Elmo doll in Figure 4.1. It can be seen in these outputs the benefit of using a multi-modal framework. The multi-colored Elmo is accurately tracked with contours (green outlines) being computed despite the complexity in both the target and background as Elmo stands tall, falls down, and sits up.

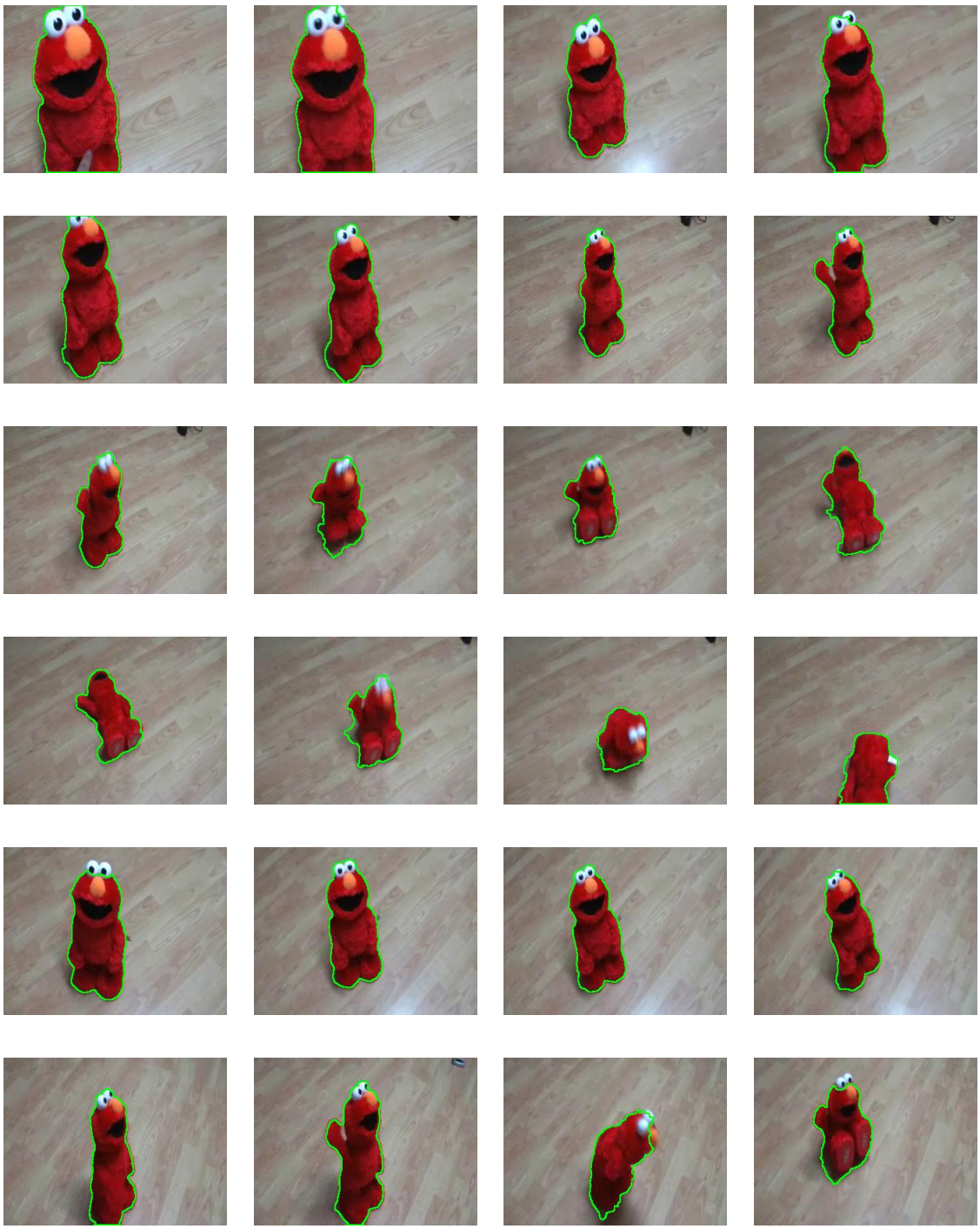


Figure 4.1: The sequence shows Tickle Me Elmo Doll undergoing a variety of deformations.

Figure 4.2: The sequence shows hand contour undergoing partial occlusion and non-rigid deformations in scale and orientation.

The second sequence shows a hand image in Figure 4.2 undergoing considerable deformations and scale changes. It demonstrates how well the contour fit the hand running through fingers. It can be noted from these images as the hand grows bigger, closes fist or orients at different angles, the contour still remains a tight fit around it.

4.2 Occlusion Object Sequences

Occlusion handling can be broadly classified to fall in either of the three categories: self occlusion, interobject occlusion, and occlusion by the background scene structure.

Self occlusion occurs mostly when one part of the object occludes another. As an example we can consider modeling a person from behind with hair and the face remains self occluded. Handling self occlusion requires dynamically updating the number of fragment models and this part is still explored in my research. An example of this scenario is shown in Figure 4.4 where the girl's hair is modeled and not face. Every time the girl turns around the contour fitting does not model the face.

Interobject occlusion occurs when two objects being tracked occlude each other. These scenarios occur when two moving people cross each other. For instance, in the same example as cited above, the girl continuously occludes the boy when they are running in circular fashion. This example demonstrates both self occlusion and interobject occlusion where the latter is handled.

Finally, occlusion by the background occurs when a structure in the background occludes the tracked objects. The Figure 4.3 shown puts an example of full occlusion occurring as the person walks behind the tree.

A common approach to handle occlusion during tracking typically followed is to model the object motion by linear dynamic models or by nonlinear dynamics and, in the case of occlusion, to keep on predicting the object location until the object reappears. For

example, a linear velocity model is used in Beymer and Konolige [6] and a Kalman filter is used for estimating the location and motion of objects. A nonlinear dynamic model is used in Isard and MacCormick [27] and a particle filter employed for state estimation. Researchers have also utilized other features to resolve occlusion, for example, silhouette projections [22] (to locate persons heads during partial occlusion), and optical flow [18] (assuming that two objects move in opposite directions). Yilmaz et al. [38] build online shape priors using a mixture model based on the level set contour representation.

In this proposed approach, occlusion is handled by following the linear dynamic object motion model. Occlusion is detected by changing shape size of the object being tracked. As the object size reduces drastically across frames and no model information is available, then the object is identified to be in occluded state. We learn the shape priors obtained by level-set modeling at runtime and best shape is hallucinated during occlusion. We utilize the shape priors being learnt before occlusion to predict the dynamic motion of the object during complete occlusion. Figures 4.3 and 4.4 shows these examples of human motion being predicted during occlusion. With the wealth of prior contour information, we could predict the way person evolves under occlusion and this is unlike prediction using fixed size object representation. The contour prediction is done by performing a search from learned database of shapes and the closest shape match is obtained using Hausdorff distance. The subsequent frames from learned database are replicated during the occlusion until the object comes back from occlusion.

Figures 4.3 and 4.4 show couple of examples of complete occlusions and hallucinations of contour during occluded time. This approach of predicting contours during complete occlusion helps in preventing tracker failures, thereby avoiding re-initialization when the object comes back. Figure 4.3 shows an example occlusion scenario where person walks behind the tree. Figure 4.4 shows a more complex scenario where the girl occludes the boy very frequently running in circular fashion at high speed (almost every 25 frames).

The proposed approach is able to handle this difficult scenario well and able to accurately fit the contour(red color). It can be noted that face of the girl is not fitted due to self occlusion at the time of fragment modeling.

4.3 Comparison with other results

In order to provide quantitative comparison of the proposed approach, we generate ground-truth for the experiments by manually labeling the object pixels in intermediate frames (every 10 frames for Elmo sequence, 5 frames for walk sequence, and 4 frames for girl sequenece). We computed the error of each algorithm on an image of the sequence as number of pixels in the image misclassified as foreground or background, normalized by image size.

The proposed approach was compared with two approaches. One was with Collins [13] method where strength was computed using using linear RGB histogram representation, and the other was using standard color histogram [38, 39, 36]. Figure 4.5 shows the comparison in terms of normalized pixel classification error. In both these approaches contours were extracted using level set framework, but fragment motion was not included for comparison. In order to make the comparison fair, the proposed approach was run without motion as well. It can be seen from the Figure 4.5 that without motion too the proposed approach of adaptive fragments performed well. Also, the proposed approach was completely automatic whereas the other two approached had to be manually re-started for every occlusion.

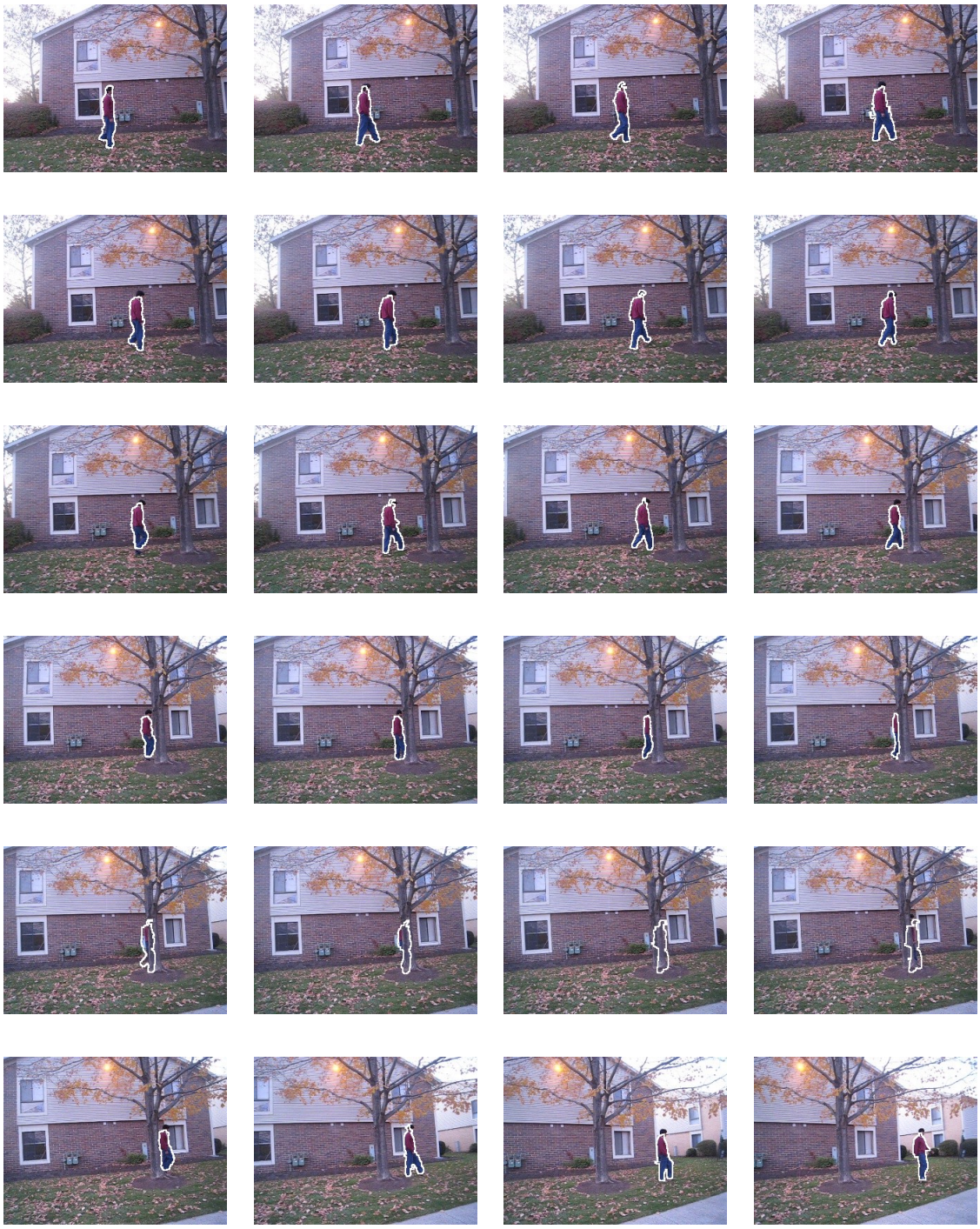


Figure 4.3: The sequence shows occlusion scenario as person walks behind tree at different frames from the video. Note the complete occlusion in fifth row.

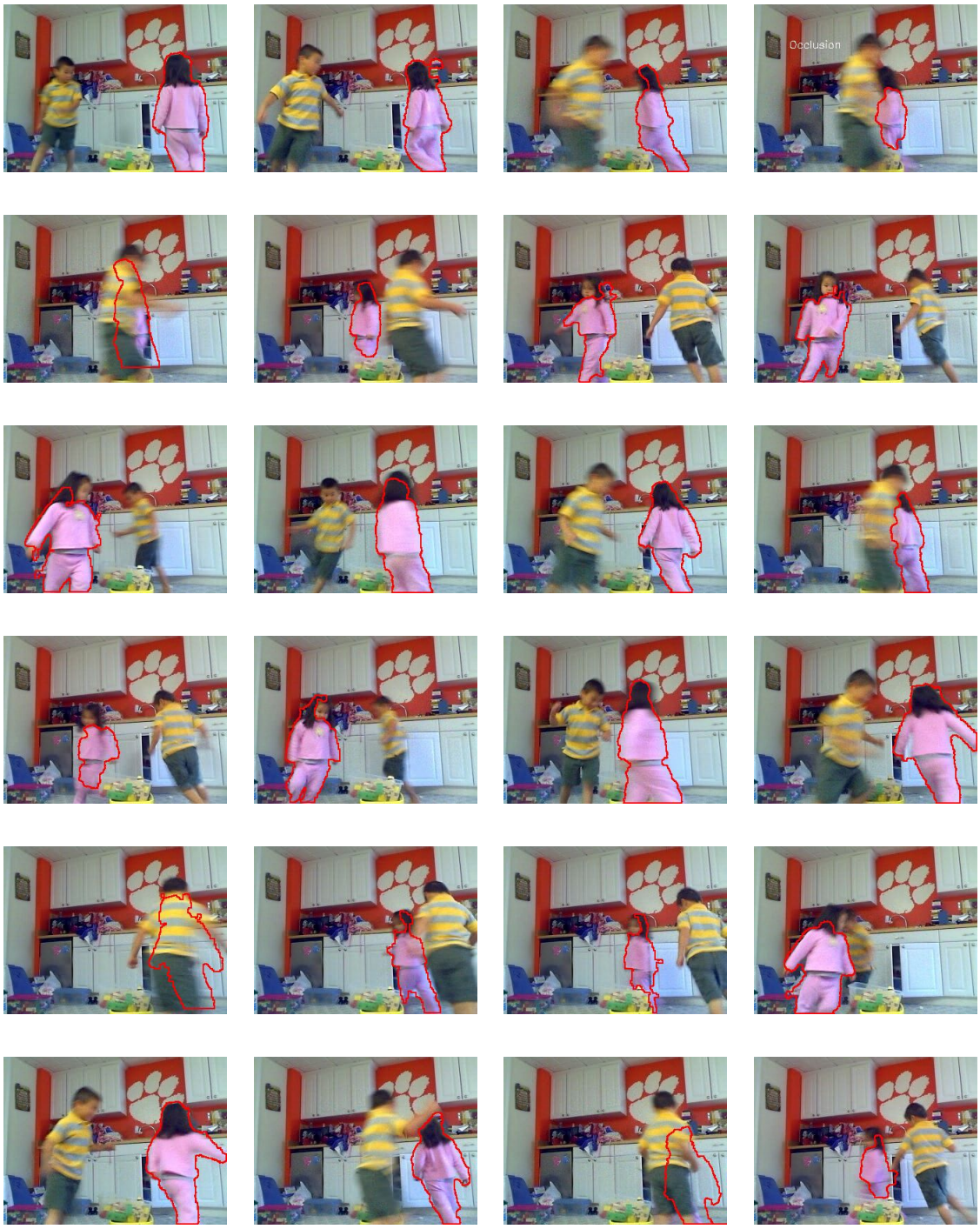
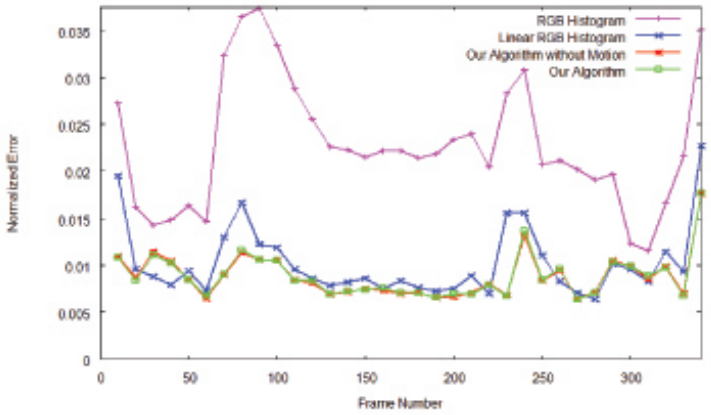
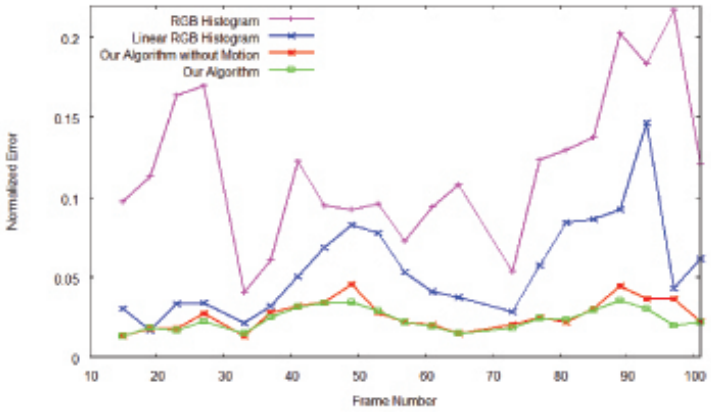


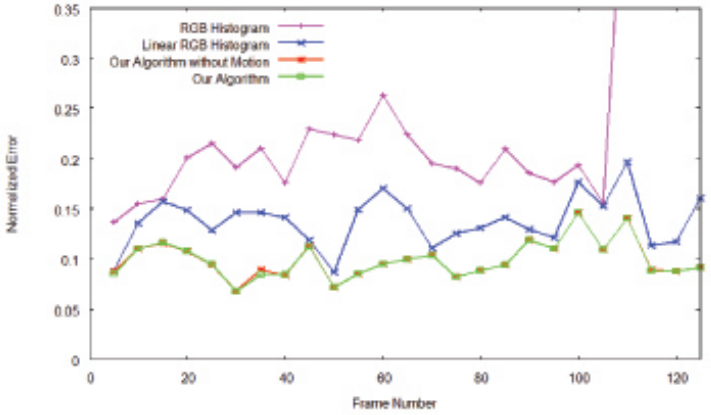
Figure 4.4: The sequence shows several severe occlusion scenario of the girl running in circular fashion. It can also be noted extreme non-rigid deformations as the girl runs around.



(a) Elmo sequence



(b) Girl sequence



(c) Walk behind tree sequence

Figure 4.5: Normalized pixel classification error results shown for elmo, girl and walk sequences. Comparison is made against linear RGB representation of Collins et al. [13] and also with [38, 39, 36] using standard color histogram. It can be seen that the proposed method outperforms other approaches in all the three sequences. Motion cue using Joint-KLT provides little help in getting accurate *Strength map* in Girl sequence as the motion is faster.

4.4 Application of Tracked Shapes - Object Recognition

As explained in earlier Section 3.4, target shapes being learnt were used to inject shapes during occlusion by learning and matching them online. These shapes of target being tracked can also be applied to recognize simple rigid objects. In order to demonstrate that we had collected few rigid objects and stored their corresponding template shapes by tracking in database. The objects considered were very simple like mug, disc and stapler with each having different shapes.

In order to recognize these objects, when a query video with one of these above mentioned objects are tracked then a search is made in database to identify the closest matching shape and henceforth recognizing the object. The matching is done using Hausdorff distance measure as explained in Section 3.4. This is similar to hallucinating with best matched prior shapes during occlusion, where the shape of tracked rigid object is matched with database of shapes instead of online shape learning. In the example Figure 4.6 shown, mug and circular lid are identified as the respective objects by matching with template shapes stored in database. This simple object recognition application demonstrates the idea of robustly tracking an object by fitting contour and thereby applying for object recognition in videos. With more sophisticated and robust shape matching techniques like [4, 31], complicated objects both rigid and non-rigid could be recognized.

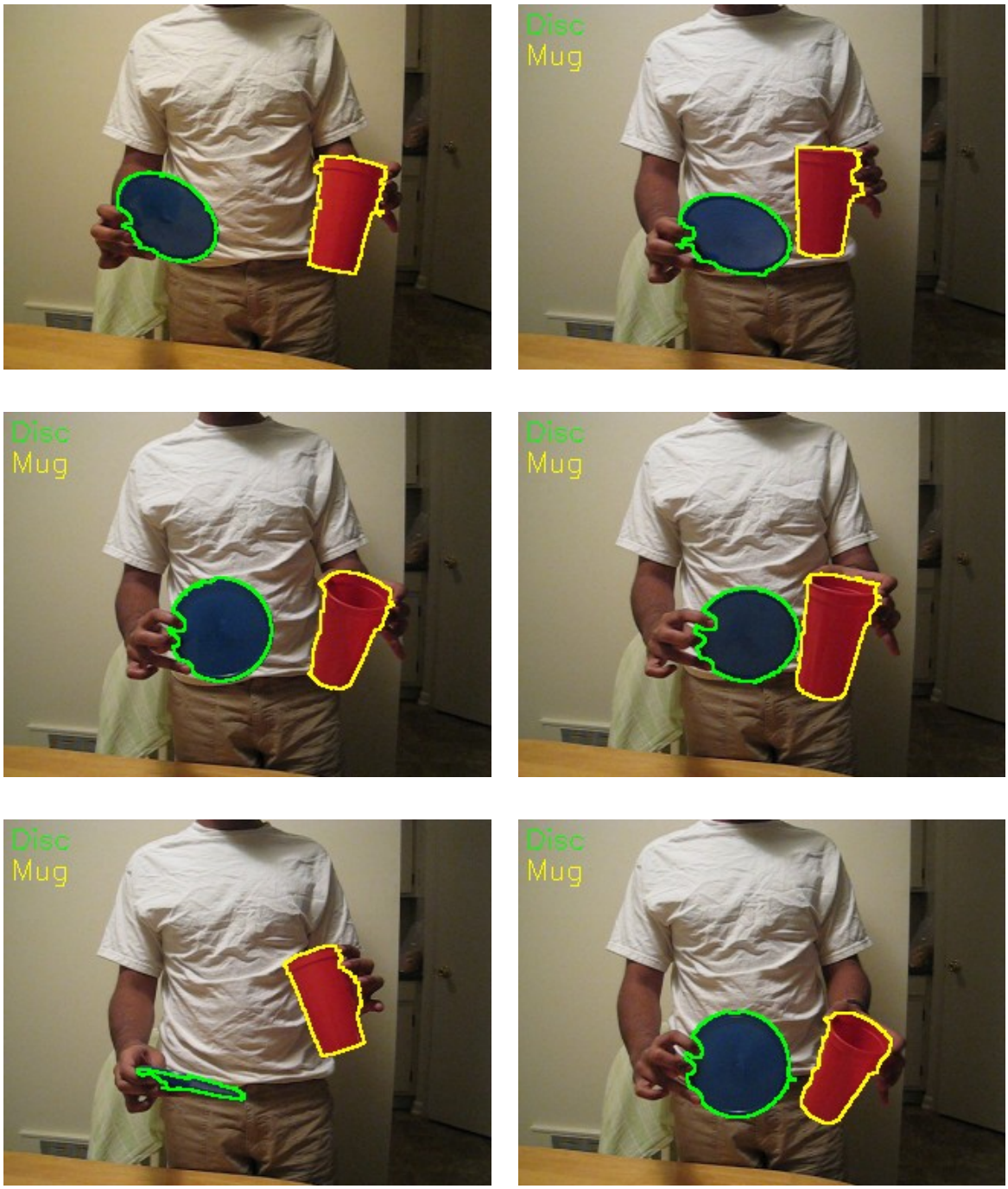


Figure 4.6: Objects being recognized as *disc* and *mug* in the images by shape matching with template in database.

Chapter 5

Tracking - Application

This Chapter presents an alternate framework for real-time detection and tracking of objects such as moving boats, vehicles and people in video datasets that were collected specifically for the particular application. In typical setting, there were multiple PTZ cameras installed at the site such that they have different non-overlapping view points and each camera tracked objects in its own view. Then, tracklets from each camera view were merged together as single track i.e had to be uniquely feature matched. The following section elaborates on the technical aspects of the algorithm used for detection and tracking along with a few sample images to illustrate the concept.

5.1 Object Detection

From the datasets gathered for this work, we observed many challenges for successfully tracking the objects. Firstly, the objects usually moved quite slowly across the frame sequence hence allowing little scope of using any motion cue for tracking them. Another reason that made this task difficult was the constantly moving background undergoing ran-

[0]This work was performed while the author was in summer internship. Due to company proprietary information, original images are not published.

dom motion like water. Secondly, these objects are often small in size since they were captured using PTZ cameras from a distance and hence it eliminated usage of sophisticated spatial features for detection.

It was however observed that all the frames in the video sequence contained background with uni-modal spectral characteristics. In the spectral domain, the background can be separated from other dynamic objects in the scene using a concept called as saliency which was initially proposed by [25]. In this method, log spectrum of the image is first computed and then analyzed to mask out frequencies corresponding to background. The spectral residual is then converted to the spatial domain to give the *saliency map*. The algorithm for this entire process is simple and straightforward and is explained in the following steps:

- Let the image under consideration be denoted as $I(x)$
- Compute the real part of the Fourier spectrum of the image as $A(f) = \Re(F[I(x)])$, where F is Fourier Transform and $\Re$ corresponds to the real part.
- Compute the phase spectrum of the image as $P(f) = \Im(F[I(x)])$, where F is Fourier Transform and $\Im$ corresponds to the imaginary part.
- Compute the log spectrum of the image as $L(f) = \log(A(f))$
- Compute the spectral residual of the image as $R(f) = L(f) - h_n(f) * L(f)$, where $h_n(f)$ is $n \times n$ matrix defined as smoothing operator
- Compute the *saliency map* of the image as $S(x) = g(x) * F^{-1}[\exp(R(f)) + j * P(f)]^2$, where $g(x)$ is a Gaussian filter, F^{-1} is Inverse Fourier Transform and $j = sqrt(-1)$

From the *saliency map*($S(x)$) obtained, a simple thresholding is done to segment the objects from the background. Figure 5.1 shows an example image with its corresponding *saliency map*.

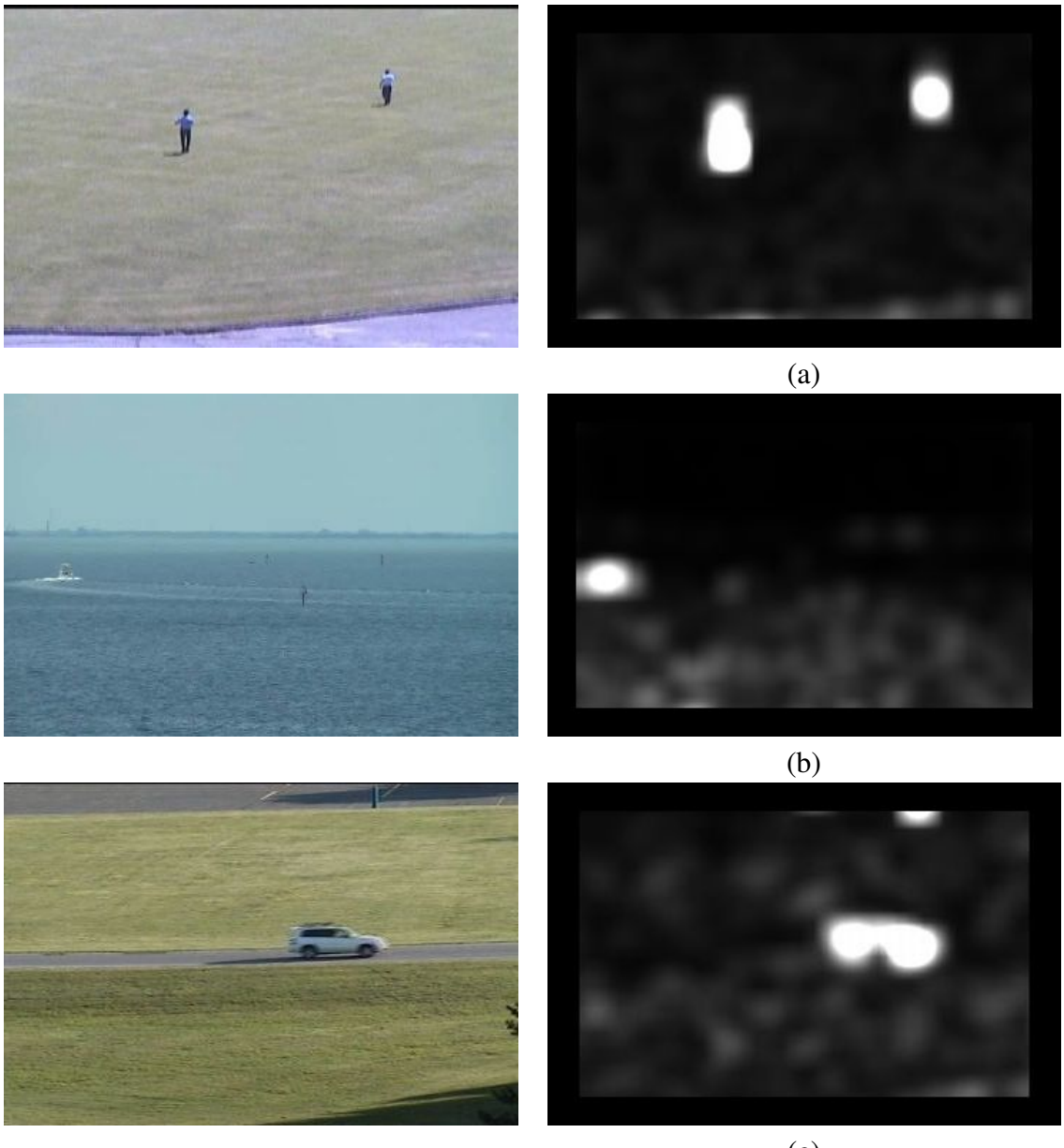


Figure 5.1: The figure (a), (b) and (c) shows an example image of a person, boat and vechicle along with their respective *saliency maps*.

5.2 Object Tracking

The objects detected using the above technique are then tracked using the proposed tracking algorithm by fitting rectangular regions of interest (ROI) to the objects rather than extracting their contours as per requirements of the system. Since the nature of objects under consideration allowed use of a uni-modal color distribution for modeling them, the present task did not require to represent them using multiple fragments as discussed in Chapter 3. The approach followed in this case is similar to the one proposed by Collins et al. [13] which first computes the likelihood map and later uses an EM-like algorithm for tracking. The next two subsections explain the algorithm in detail.

5.2.1 Strength Model Computation

For an image frame, all objects are detected in it using the procedure explained in the previous Section 5.1. Once the objects are detected, they are segmented from the background by an online process of selecting color features (RGB) in linear space as proposed by [13]. The most promising linear combination of RGB feature is selected that best discriminates the object from the background class. Based on the appearance model of objects being detected, as described in Section 5.1, a linear distribution of features is computed for both the object and its background. The most discriminative 1D color subspace needs to be chosen from an initial superset of linear combinations of RGB features. In the present case, this superset of features from which the best candidate is selected, is shown by F_1 in the equation below and is generated by taking linear combinations of integer coefficients ranging from -2 to 2.

$$F_1 = w_1 R + w_2 G + w_3 B \tag{5.1}$$

where $w_* \in (-2, -1, 0, 1, 2)$. The total number of combinations of these features comes to 125, out of which only 49 features are non-redundant. There are two major advantages of using these set of features, (a) they are easy and efficient to compute and (b) they represent the 3D RGB space as a uniform set of 1D subspaces. These features are normalized in the range 0 to 255 and discretized into histograms with 32 bins. Each feature from the set of 49 combinations, are evaluated to determine the one that best separates the object from its background. Here, the term background refers to the immediate surrounding of the object. Hence, for each feature (a) estimate the distribution for both the object and background (b) compute log likelihood of these distributions and (c) apply the variance ratio measure to these likelihood values.

In mathematical terms, for a feature f, let $p(i)$ and $q(i)$ be the normalized discrete probability densities of the object and its background respectively, such that i is the index of histogram bin from 0 to 32. The log likelihood of feature value i is given by

$$L(i) = \log\left(\frac{\max(p(i), \delta)}{\max(q(i), \delta)}\right) \tag{5.2}$$

where δ is a small value (0.001). As mentioned in Section 3.1, positive values of L correspond to the object while negative values to the background. These log likelihood values are computed for each feature f and for each bin i of its histogram distribution.

Variance ratio of log likelihood function is defined as total variance over both object and background pixels $var(L; (p+q/2))$ divided by sum of within class variances of object $var(L; p)$ and background $var(L; q)$ separately. Higher the value of variance ratio for a feature f, better is the separation it offers between the object and background classes. Hence, in order to compute the best separability of object and background class from the candidate set of 49 features, variance ratio L of each feature f is computed and rank-ordered. The feature f with the highest score is then selected and employed for tracking

Figure 5.2: The figure shows an example image of a person along with the respective *strength map* obtained using linear combinations of RGB. Using such *strength maps* obtained in each frame, the objects are being tracked.

the object.

$$VR(L;p,q) = \frac{var(L;(p+q)/2)}{[var(L;p) + var(L;q)]} \tag{5.3}$$

where for any distribution $a(i)$, variance of $L(i)$ with respect to a is

$$var(L;a) = \sum_i a(i)L^2(i) - [\sum_i a(i)L(i)]^2 \tag{5.4}$$

The log likelihood function corresponding to the feature with the highest variance ratio is selected and used to compute the *strength map*,S. In subsequent frames, if RGB value of a pixel is quantized into say, bin i of the 32-bin feature histogram, then the corresponding likelihood value $L(i)$ is selected for computation of the strength map. While in the work published by [13], the authors have considered updating these features during run time to account for changes in view point of the object, in the present scenario of tracking application the object's view in one camera remains nearly constant and hence there was no need felt for doing a dynamic update, thereby saving computation cost for this real-time application. Figure 5.2 shows an example *strength map*.

5.2.2 Maximum Likelihood (ML) Framework

In objective of this tracking application was to recover a tightly bound ROI around the object in every frame. In order to get such a tight bound, where the ROI varies with respect to slight changes in the scale of the object, the best approach was to use a ML framework with an EM-like algorithm to iteratively estimate new mean $\mathbf{M}^0$ and covariance $\mathbf{V}^0$ of the ROI. The mean and covariance of the pixels in the foreground helped in determining the position and shape of the object.

To elaborate in mathematical terms, let R^+ be the ROI of the object that needs to be estimated. Given the target model Ψ^+ and background model Ψ^- respectively, the objective of the search mechanism is to find a region $\mathcal{R}$ in the new frame described by mean and covariance $(\mathbf{M}, \mathbf{V})$ that maximizes the following function :

$$J(\mathbf{M}, \mathbf{V}) = \sum_{\mathbf{x} \in \mathcal{R}} S(\mathbf{x}) L(\mathbf{x}|\mathbf{M}, \mathbf{V}) \tag{5.5}$$

where $S(x)$ is the strength map and the term

$$L(\mathbf{x}|\mathbf{M}, \mathbf{V}) \propto exp(-(\mathbf{x} - \mathbf{M})^t \mathbf{V}^{-1} (\mathbf{x} - \mathbf{M})), \tag{5.6}$$

prevents pixel locations that are farther from the original region from distracting the tracker. Note here that t indicates transpose and -1 indicates inverse of matrix. As a pixel's contribution falls off with the distance from the original region, this helps in both reducing the effect of outlier pixel on the search as well as preventing the tracker from *drifting* away from the object.

As shown in [40, 16, 32], the maximum-likelihood estimates of $\mathbf{M}$ and $\mathbf{V}$ can be obtained via an EM-like iterative procedure. Starting with an initial estimate $\mathbf{M}^0, \mathbf{V}^0$ of $\mathcal{R}$, the EM-iteration proceeds as below:

- **E-Step:** Given current estimates $\mathbf{M}^k$ and $\mathbf{V}^k$ of the mean and covariance of the region, compute hidden variables $w^k(\mathbf{x})$:

$$w^k(\mathbf{x}) = \frac{s(\mathbf{x})\mathbf{L}(\mathbf{x}|\mathbf{M^k}, \mathbf{V^k})}{\sum_{\mathbf{x}' \in \mathcal{R}} s(\mathbf{x})\mathbf{L}(\mathbf{x}'|\mathbf{M^k}, \mathbf{V^k})} \tag{5.7}$$

- **M-Step:** Using the hidden variables computed above, compute the next estimates of mean and covariance of the region, $\mathbf{M}^{k+1}$ and $\mathbf{V}^{k+1}$ of that maximize $J(.,.)$:

$$\mathbf{M}^{k+1} = \sum_{\mathbf{x} \in \mathcal{R}} w^k(\mathbf{x})\mathbf{x} \tag{5.8}$$

$$\mathbf{V}^{k+1} = \sum_{\mathbf{x} \in \mathcal{R}} w^k(\mathbf{x})(\mathbf{x} - \mathbf{M}^{k+1})(\mathbf{x} - \mathbf{M}^{k+1})^t \tag{5.9}$$

The optimal values for $\mathbf{M}$ and $\mathbf{V}$ are obtained by iterating the above steps until convergence. In practice, the method converges in about 2 to 3 iterations.

5.3 Results and Discussion

The present system has been developed and tested on Intel Duo-Core Processor with 3.4 GHz and 4GB RAM. The code has been developed using C++ in Microsoft Visual Studio .Net environment. The detection and tracking algorithm runs at around 25 - 30 frames per second depending on number of objects being tracked.

In order to comply with company's policy to protect its intellectual property, I have displayed the performance of the proposed algorithm as graphical plots instead of overlaying the results on the original datasets. Figure 5.3 and 5.4 shows plots of object trajectories obtained from tracking objects in different sequences. In these plots, the y-axis denotes the distance of centroid of object from the image origin while the x-axis denotes the respective frame number. For plots 5.3(a), 5.3(b) and 5.4(c), the path in red indicates motion of the

object of interest while blue indicates false detections. In plot 5.4(d), multiple objects are being simultaneously tracked and each represented by a unique color. By analyzing these plots carefully, we arrive at the following observations:

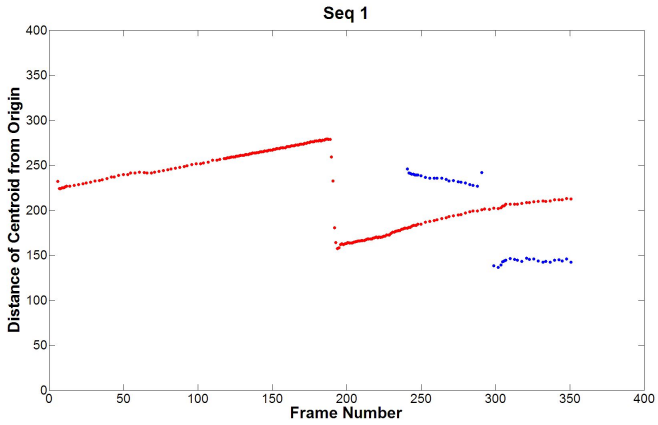
- In plot 5.3(a), there is a sudden shift in the motion of the object around frame number 200. This is due to a fast camera panning across it's view. As it can be observed, the object was still locked on by the tracker. There are also couple of false detections which the tracker encounters as highlighted in blue.

- In plot 5.3(b), there is a smooth and consistent track of the object of interest for the entire length of the video sequence.

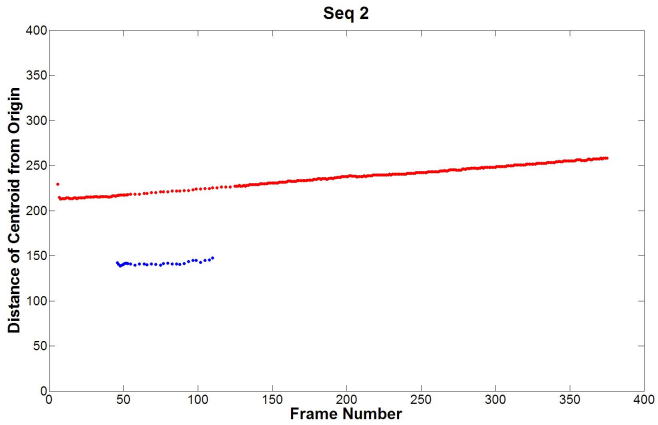
- In plot 5.4(c), the corresponding video sequence is extremely jittery due to significant camera shakes along with panning. Even in such scenarios, the proposed tracker performs well. Even though there is a slight drift in the ROI across the frame sequence, the tracker always latches back on to the object due to a good initial template strength model.

- In plot 5.4(d), multiple objects were tracked together. It can be noticed from the plot that the path of the object in green merges with the one in black around frame number 165. This happens because of two objects with very similar color characteristics that are moving close to one another.

The small patches of blue paths that we notice in plots 5.3(a), 5.3(b) and 5.4(c) are due to false saliency detections made by the object detection module. These erroneous ROIs are handed over to the tracker which tries to track them across frames using their appearance. It can be seen from the plots that these paths are most often short-lived and die off across time. Also, it should be noted here that these erroneous tracks do not pose

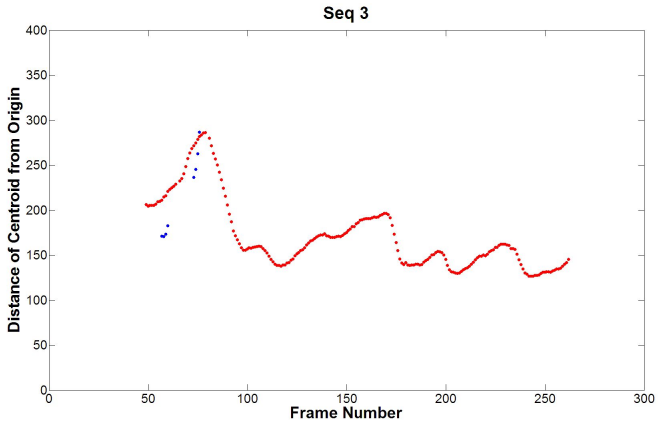


(a)

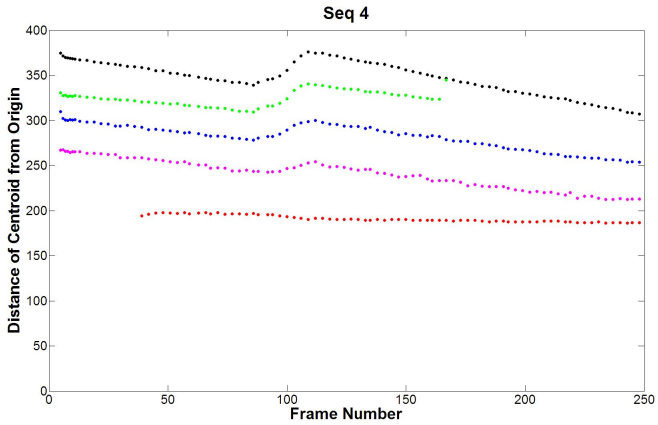


(b)

Figure 5.3: The figure (a) and (b) shows plots of object trajectories being tracked in two different sequences.



(c)



(d)

Figure 5.4: The figure (c) and (d) shows plots of object trajectories being tracked in two different sequences.

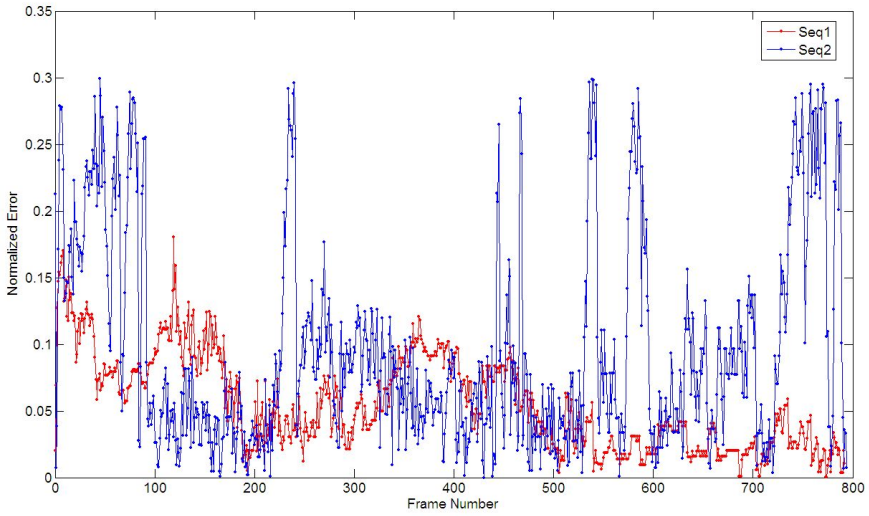


Figure 5.5: Normalized ROI intersection error shown for two sequences.

a problem to the final system since they are removed in the future stages which employ feature matching to determine association of tracks across multiple PTZ views.

Figure 5.5 shows the normalized deviation of the tracked ROI from the one obtained using ground-truth for two sequences. The normalized error to denote the deviation between the two ROI's is given by (5.10).

$$E_{norm} = 1 - \frac{R_g \cap R_t}{R_g \cup R_t} \tag{5.10}$$

where R_g and R_t are the object ROI's obtained from ground-truth and tracking respectively. The ground-truth was performed for every alternate frame in the original sequence of 1600 frames. The red sequence corresponds to a video in which the object was tracked smoothly thereby producing smaller drift errors w.r.t the ground-truth. The blue sequence on the other hand has been generated by a video that had large camera shakes and jitter.

Though there are limitations existing in terms of dependence being only on color and motion not being considered due to slow speed of objects, it can still be said that

the application performed well. Although motion cue couldn't be applied, still further robustness can be added by combining KLT based features [30, 3] on detected objects with strength map. Since there existed a clear distinction in the characteristics of objects from background at the boundary, good features that can be efficiently associated and tracked in such an environment can be developed. The potential of such features gets completely utilized only when they can be combined well with color features. Currently, this is being explored on efficient ways to combine KLT features with strength map to make tracker more robust.

Chapter 6

Conclusions

I have presented a tracking algorithm to accurately track non-rigid objects using their contour. This was achieved by modeling the foreground and background regions of the object using a mixture of Gaussians in spatial-feature space. A simple region growing segmentation was introduced to identify multiple fragments in the object and background. Using the GMM, we computed the *strength map* giving the probability distribution if a pixel belongs to foreground or background. This *strength map* was modeled in level set framework in chan-vese manner thereby allowing contours to adapt from previous contour estimations. Joint feature tracking was incorporated to improve performance of algorithm as was discussed in experimental section. Experimental results were shown for the algorithm where accurate boundaries of multi-colored objects undergoing lot of shape changes, fast motions, and complete occlusion were obtained. Also, application of object shapes was shown to recognize simple objects. Future work will involve utilizing this idea and build robust shape priors enabling object classification while tracking.

Further, as part of internship, I presented an algorithm for combining detection and tracking efficiently for an application involving different types of objects like boats or humans. These objects were detected in spectral domain where frequencies corresponding

to relatively unimodal background was masked out, and then *strength map* was applied to track the detected objects by modeling them using linear RGB histograms. A search technique for the objects was discussed using ML framework to obtain accurate and tight ROI locks on them. The application was demonstrated to perform well.

Bibliography

[1] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, August 2004.

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

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

[4] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Proc. of IEEE CVPR*, pages I: 26–33, 2005.

[5] D. Beymer and K. Konolige. Real-time tracking of multiple people using continuous detection. In *Proc. ICCV Frame-rate Workshop*, 1999.

[6] David Beymer and Kurt Konolige. Real-time tracking of multiple people using continuous detection. In *In Proceeding of International Conference on Computer Vision (ICCV)*, 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] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *International Journal of Computer Vision*, pages 329–342, 1998.

[9] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag, 1998.

[10] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):90–99, 1986.

[11] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[12] Tony F. Chan and Luminita A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, February 2001.

[13] 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.

[14] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–577, 2003.

[15] D. Cremers. Dynamical statistical shape priors for level set-based tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1262–1273, August 2006.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[17] R. Deriche and N. Paragios. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):415, April 2000.

[18] Shiloh L. Dockstader and A. Murat Tekalp. On the tracking of articulated and occluded video object motion. *Real-Time Imaging*, 7(5):416–431, 2001.

[19] 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.

[20] 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.

[21] 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.

[22] Ismail Haritaoglu, Davis Harwood, and Larry S. David. W4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):809–830, 2000.

[23] F. Hausdorff. *Set Theory*. Chelsea, NY, USA, 1962.

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

[25] Xiaodi Hou and Liqing Zhang. Saliency detection: A spectral residual approach. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[26] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge. Tracking non-rigid objects in complex scenes. In *In Proceedings of International Conference on Computer Vision (ICCV)*, volume 93, pages 93–101, 1993.

[27] M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. In *IEEE International Conference on Computer Vision*, volume 2, page 34, Los Alamitos, CA, USA, 2001.

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

[29] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[30] 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, 1981.

[31] G. Mori, S. J. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *Proc. of IEEE CVPR*, pages I:723–730, 2001.

[32] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[33] Nalin Pradeep, Prakash C., and Stan Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *In Proceedings of International Conference on Computer Vision (ICCV)*, 2009.

[34] S. Sanjay-Gopal and T. J. Hebert. Bayesian pixel classification using spatially variant finite mixtures and the generalized EM algorithm. *IEEE Transactions on Image Processing*, 7(7):1014–1028, July 1998.

[35] Girgios Sfikas, Christophoros Nikou, and Nikolaos Galatsanos. Edge preserving spatially varying mixtures for image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[36] 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, 2005.

[37] C. J. Veenman, Marcel J. T. Reinders, and Eric Backer. Resolving motion correspondence for densely moving points. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 23, pages 54–72, 2001.

[38] Alper Yilmaz, Xin Li, and Mubarak 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, 2004.

[39] 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, 2003.

[40] Z. Zivkovic and B. Krose. An EM-like algorithm for color-histogram-based object tracking. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 798–803, 2004.