Region Tracking Over an Image Sequence

Jeff Calder, Dana Awamleh, Allan MacAulay

Supervisor: Professor Abdol-Reza Mansouri

April 2008

Many thanks to Professor Abdol-Reza Mansouri for his guidance and support throughout the course of this project.

Abstract

The goal of this project was to design, implement on computer, and test on real images a variety of computer vision algorithms for region tracking over sequences of images. The region to be tracked is specified only in the first frame, and the algorithm proceeds to track the region through subsequent frames. Two different types of algorithms were tested using certain image characteristics. The first, local tracking, is concerned with comparing local structures of the region between two frames. The second, global tracking, is concerned with comparing the distributions of certain image characteristics in the region between two frames. Some very promising theoretical results have been demonstrated by viewing the image a 3-D surface and using functions of the principal curvatures as image characteristics for tracking. When comparing local and global tracking, we conclude that in general, they have complementary properties and properly combining the two may yield superior tracking to either one alone. This, however, has not been experimentally verified.

Contents

List of Figures

1 Introduction

The problem of region tracking over an image sequence falls under the umbrella of computer vision. The ultimate goal of computer vision is to have a computer ”understand” what it sees, allowing the computer to interpret image data for the application intended.

There are many existing applications of computer vision, including but not limited to areas such as optical character recognition, medical image processing, automated traffic monitoring, automated video editing, video surveillance, and of course military applications.

The specific goal of region tracking is to track a region of interest over a sequence of images. In order to do this, we must ask and answer the fundamental question:

What remains invariant...?

- Shape?
- Intensity Boundaries?
- Motion?
- Smoothness/Stationarity of the Background?

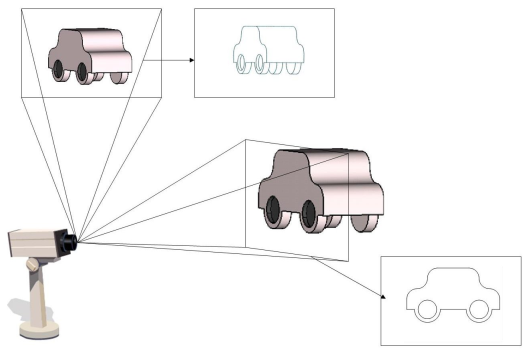

Figure 1: A camera filming a car as it passes by

If we consider the situation in Figure 1, we find that infact none of these factors can be assumed to remain invariant in an image sequence. For example, the shape of the car when projected into the 2-dimensional image space completely changes shape as the camera pans and follows the car. Intensity boundaries do not remain invariant, as the lighting, background, and other factors may change throughout the sequence. The car can slow down, speed up, or even stop, so clearly motion does not remain invariant. Also, we clearly cannot assume anything about the smoothness or stationarity of the background which is changing as the camera pans to follow the car.

There are many existing region tracking algorithms which make explicit assumptions about the above images characteristics, however, they fail on any image sequences for which these assumptions do not hold. The reduced applicability of these algorithms is a major drawback as they only work in very specialized cases.

The goal of this project is to design, implement, and test a general purpose tracking algorithm that does not make any explicit assumptions other than that the region of interest is distinguishable in some way between frames. Such a general purpose algorithm will, for example, be able to track equally well a hockey player, a car, or a boxing glove.

2 Existing Algorithms

Region tracking is a difficult problem which has been approached in many different ways. Some depend on region information, others utilize boundary information and some use both region and boundary information.

Some region based algorithms depend on feature points and edge segments of the region being tracked. Once region information is used, the motion field can be computed over the region and used to compute the position of the region in the next frame. Using this method involves making strong assumptions on the shape of the region and the motion of the points. This algorithm is optimal if the motion field is translational and the shape of the region is constant throughout the sequence, but this is not the case for most tracking problems. The main downside to this algorithm is that boundaries are not accurately computed [1].

Other approaches use boundary information through an active contours approach. Here, no assumptions are made about the shape of the region being tracked, as the algorithm only observes the boundaries and not the region itself. It was assumed, however, that there is a high contrast between the region being tracked and the background, which is not always the case [2].

Thus combined region and boundary based algorithms were developed, which are topology independent, and allow the region being tracked to merge with other regions or split into several. These algorithms can only be applied to a motion pattern where the displacement of the region being tracked is small, depending on the quality of the motion field and given parameters [1]. Also, the object is assumed to be moving while the background is stationary.

In this project we propose an algorithm that performs region tracking without any of the a priori assumptions mentioned above. No assumptions are made about smoothness, shape, contrast, intensity or motion of the region. Also, we do not assume a fixed uniform background. Consequently, computations of motion fields, parameters, and features points for motion will not be needed, and the motion will not be restricted to small displacements.

3 Problem Formulation

The problem formulation described as follows is taken from a problem description provided by Professor Abdol-Reza Mansouri in September of 2007[3]. The aim of this project is to implement and test a region tracking algorithm based on a variational formulation. Consider then two images $I_0, I_1 : \Omega \mapsto \mathbb{R}$ (where $\Omega \subset \mathbb{R}^2$ is the image domain) corresponding to two frames of an image sequence and assume we are given some region $R_0 \subset \Omega$. Typically R_0 would correspond to some region of interest in image I_0. We would like to "track" R_0 in the next frame, i.e. find the corresponding region in frame I_1. Furthermore, we wish to do so without making any constraining assumptions regarding the shape of R_0, its motion, or its contrast with the background. To make the problem tractable, some assumptions must be made. We will assume that $R_1 \subset \Omega$ "corresponds" to R_0 if certain characteristics of the function $I_0|_{R_0}$ agree with those of $I_1|_{R_1}$.

Denote by $\mathbf{v}(\mathbf{x}, I) \in \mathbb{R}^n$ a vector of characteristics (or features) of I at the point $x \in \Omega$. $\mathbf{v}(\mathbf{x}, I)$ could be a simple as $I(\mathbf{x})$ (hence a one-dimensional vector), or a more complicated function of $\mathbf{x}$ and I obtained by applying various operators to I in the neighborhood of $\mathbf{x}$. In this way, $\mathbf{v}(\mathbf{x}, I)$ could contain information about the image not only at $\mathbf{x}$ but also in a neighbourhood of $\mathbf{x}$. In section 5.6 we will describe the operators that we use for $\mathbf{v}(\mathbf{x}, I)$.

Consider the following functional:

$$\begin{aligned}
R \mapsto E[R] = {} & \lambda_1 \int_R \min_{\tau \in B(r):\mathbf{x}+\tau \in R_0} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \, d\mathbf{x} \\
& + \lambda_1 \int_{R^c} \min_{\tau \in B(r):\mathbf{x}+\tau \in R_0^c} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \, d\mathbf{x} \\
& + \lambda_2 D(f_{I_0,R_0} || f_{I_1,R}) \\
& + \lambda_3 D(f_{I_0,R_0^c} || f_{I_1,R^c}) \\
& + \lambda_4 \oint_{\partial R} ds
\end{aligned}$$

where R is a region in frame I^1, $B(r)$ denotes the closed ball of radius r centered at 0, $|| \cdot ||$ is some appropriate vector norm, $f_{I,R}$ denotes the probability density function of $\mathbf{v}$ in region R of image I, and $D(\cdot||\cdot)$ denotes the Kullback-Leibler distance measure. λ_1, λ_2, λ_3, and λ_4 are parameters that are determined experimentally. Estimating R_1 can now be expressed as finding a region $R*$ that minimizes the functional E. This will be expressed as the solution to a gradient descent partial differential equation. Before we derive the PDE, we need to review some background mathematics.

4 Background Mathematics

We will first dive into the calculus of variations as it will give us the tools required to properly discuss our solution. An excellent reference on this subject is Fomine and Gelfand[4].

4.1 Calculus of Variations

The calculus of variations is an integral tool to our problem formulation. It will allow us to devise an algorithm to minimize our tracking functional. Before we begin with the theory, we will introduce some notation that will be used throughout this paper. Let $\zeta([a,b];\alpha,\beta)$ be the set of all C^∞ functions from $[a,b]$ to $\mathbb{R}$ such that $f(a)=\alpha$ and $f(b)=\beta$. That is,

$$\zeta\left([a,b];\alpha,\beta\right) = \{f : [a,b] \mapsto \mathbb{R} | f \in C^\infty, f(a)=\alpha, f(b)=\beta\} \tag{4.1}$$

Now, for $L : \mathbb{R}^3 \mapsto \mathbb{R}$, $L \in C^\infty$ we define the functional
$A_L : \zeta([a,b];\alpha,\beta) \mapsto \mathbb{R}$ by

$$A_L[f] = \int_a^b L(x, f(x), f'(x))\mathrm{d}x \tag{4.2}$$

where $f \in \zeta([a,b];\alpha,\beta)$. Hence, A_L is a function of f and its first derivative. As A_L takes values on the real number line we can consider the problem of minimizing A_L over all $f \in \zeta([a,b];\alpha,\beta)$. This fixed endpoint problem can be thought of as trying to find a function that goes from $f(a)=\alpha$ to $f(b)=\beta$ while minimizing the cost function defined by (4.2). This leads us naturally to the Euler-Lagrange necessary conditions[4] which are proved in the following theorem.

Theorem 1. *(Euler-Lagrange Necessary Conditions.) Let $L : \mathbb{R}^3 \mapsto \mathbb{R}$ be a C^∞ function. Then, if $f \in \zeta([a,b];\alpha,\beta)$ is a local minimum of A_L, it must satisfy the following equation*

$$\frac{\partial L}{\partial f} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial L}{\partial f'}\right) = 0 \tag{4.3}$$

Remark 1. *$\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial f}$ and $\frac{\partial L}{\partial f'}$ denote the partial derivatives of L with respect to each of its three variables*

Proof. Assuming that f minimizes A_L, let $\eta \in \zeta([a,b];0,0)$ (ie: $\eta(a) = \eta(b) = 0$). Hence, $\forall t \in \mathbb{R}$, $f + t\eta \in \zeta([a,b];\alpha,\beta)$. Now, consider the function $g_\eta : (-\epsilon,\epsilon) \mapsto \mathbb{R}$ defined by $t \longmapsto A_L[f+t\eta]$. Since f is a minimum of A_L, $t=0$ is a minimum of g_η. From classical calculus, this implies that

$g'_\eta(0) = 0$. Therefore, we can write

$$\begin{aligned}
\left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} g_\eta(t) &= \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} A_L[f+t\eta] \\
&= \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} \left(\int_a^b L(x,(f+t\eta)(x),(f+t\eta)'(x))\mathrm{d}x\right) \\
&= \int_a^b \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} \big(L(x,(f+t\eta)(x),(f+t\eta)'(x))\big)\,\mathrm{d}x \\
&= \int_a^b \frac{\partial L}{\partial f}\eta(x) + \frac{\partial L}{\partial f'}\eta'(x)\mathrm{d}x \\
&= \int_a^b \frac{\partial L}{\partial f}\eta(x)dx + \int_a^b \frac{\partial L}{\partial f'}\eta'(x)\mathrm{d}x \\
&= \int_a^b \frac{\partial L}{\partial f}\eta(x)dx + \left[\frac{\partial L}{\partial f'}\eta(x)\right]_a^b - \int_a^b \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial L}{\partial f'}\right)\eta(x)\mathrm{d}x \\
&= \int_a^b \left(\frac{\partial L}{\partial f} - \frac{d}{dx}\left(\frac{\partial L}{\partial f'}\right)\right)\eta(x)\mathrm{d}x = 0
\end{aligned}$$

There is a slight abuse of notation throughout the proof. All the partial derivatives are assumed to be evaluated at $(x, f(x), f'(x))$ (ie: $t = 0$). The second to last line is obtained through integration by parts and the middle term is zero by the assumption that $\eta(a) = \eta(b) = 0$. The proof is completed by noting that the final equation must hold for any $\eta \in \zeta([a,b];0,0)$. □

Although the Euler-Lagrange equation (4.3) is not a sufficient condition for minimality, it is often useful in finding critical points of functionals of the form in equation (4.2). We are now equipped to talk about gradient descent in function space. This is best understood by first considering gradient descent in classical calculus. Let $F : \mathbb{R}^n \mapsto \mathbb{R}$ be some function we wish to minimize and consider the following differential equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = -\nabla F(\mathbf{x}(t)) \tag{4.4}$$

If $\mathbf{x}(t)$ is a solution to equation 4.4 then we see that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}(F(\mathbf{x}(t)) &= \left\langle \nabla F(\mathbf{x}(t)), \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right\rangle \\
&= \langle \nabla F(\mathbf{x}(t)), -\nabla F(\mathbf{x}(t))\rangle \\
&= -\left\|\nabla F(\mathbf{x}(t))\right\|^2 \leq 0
\end{aligned}$$

Therefore, $F(\mathbf{x}(t))$ is a monotonically non-increasing function of t. Now, depending on the initial condition $\mathbf{x}(0)$, as $t \to \infty$ the function $F(\mathbf{x}(t))$ will either be divergent, or converge to a local minimum or saddle point of

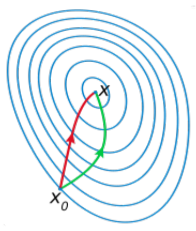

Figure 2: Gradient Descent Illustration

F. This is nicely illustrated in Figure 2. Starting at x_0, the curve $\mathbf{x}(t)$ will converge as $t \longrightarrow \infty$ to the local minimum x. By combining this idea with the Euler-Lagrange equation, we naturally arrive at the idea of gradient descent in function space[4], which is the subject of the following proposition.

Proposition 1. *(Euler-Lagrange Descent Equation) Let $L : \mathbb{R}^3 \mapsto \mathbb{R}$ be a C^∞ function. Now, take a family of functions, f, in $\zeta([a,b],\alpha,\beta)$ parameterized by t (ie: $f(\cdot,t) \in \zeta([a,b];\alpha,\beta)$ $\forall t \in \mathbb{R}^+$). If f satisfies*

$$\frac{\partial f}{\partial t} = -\left(\frac{\partial L}{\partial f} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial L}{\partial f'}\right)\right) \tag{4.5}$$

then $\forall t \in \mathbb{R}^+$,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(A_L[f(\cdot,t)]\right) \leq 0$$

Proof. Given that f is a solution to 4.5, we have

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(A_L[f(x,t)]\right) &= \frac{\mathrm{d}}{\mathrm{d}t}\int_a^b L(x,f(x,t),f'(x,t))\mathrm{d}x \\
&= \int_a^b \frac{\mathrm{d}}{\mathrm{d}t}L(x,f(x,t),f'(x,t))\mathrm{d}x \\
&= \int_a^b \left(\frac{\partial L}{\partial f}\frac{\partial f}{\partial t} + \frac{\partial L}{\partial f'}\frac{\partial f'}{\partial t}\right)\mathrm{d}x \\
&= \int_a^b \left(\frac{\partial L}{\partial f}\frac{\partial f}{\partial t} + \frac{\partial L}{\partial f'}\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial f}{\partial t}\right)\right)\mathrm{d}x \\
&= \int_a^b \frac{\partial L}{\partial f}\frac{\partial f}{\partial t}\mathrm{d}x + \left[\frac{\partial L}{\partial f'}\frac{\partial f}{\partial t}\right]_a^b - \int_a^b \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial L}{\partial f'}\right)\frac{\partial f}{\partial t}\mathrm{d}x \\
&= \int_a^b \left(\frac{\partial L}{\partial f} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial L}{\partial f'}\right)\right)\frac{\partial f}{\partial t}\mathrm{d}x \\
&= \int_a^b -\left(\frac{\partial L}{\partial f} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\partial L}{\partial f'}\right)\right)^2 \mathrm{d}x \leq 0
\end{aligned}$$

□

We will now see some applications of these theorems.

4.1.1 Minimizing Euclidean Length of a Closed Curve

Let $C = \{\vec{\gamma} : [0,1] \mapsto \mathbb{R}^2 \;\; | \;\; \vec{\gamma}(0) = \vec{\gamma}(1)\}$ and consider the functional $E : C \mapsto \mathbb{R}$ defined by $E[\vec{\gamma}] = \oint_{\vec{\gamma}} \mathrm{d}p$. So, to find the curve $\vec{\gamma}^*$ that minimizes E, we embed $\vec{\gamma}$ in a one-parameter family $(\vec{\gamma}(\cdot,t))_{t\geq 0}$ such that as $t \to \infty$, $\vec{\gamma}(\cdot,t)$ converges to $\vec{\gamma}^*$. Such a family is obtained by calculating the Euler-Lagrange descent equations of Proposition 1. In our development thus far we have only considered scalar valued functions but the generalization to vector valued functions, such as $\vec{\gamma}$, is simple once we note that we can express $\vec{\gamma}(s)$ as $\vec{\gamma}(s) = (x(s), y(s))$ where $x(s), y(s) \in \zeta([0,1]; 0, 0)$. So in writing the Euler-Lagrange descent equation for $\vec{\gamma}(\cdot,t)$ we will obtain a system of coupled partial differential equations for $x(s)$ and $y(s)$. First, note that the functional E can be expressed as

$$E[\vec{\gamma}] = \oint_{\vec{\gamma}} \mathrm{d}p = \int_0^1 \left\| \dot{\vec{\gamma}}(s) \right\| \mathrm{d}s \;\; = \int_0^1 \sqrt{\dot{x}^2(s) + \dot{y}^2(s)}\mathrm{d}s$$

For simplicity we will often omit t, but one should keep in mind that $\vec{\gamma}, x$, and y are families of functions parameterized by t. The Euler-Lagrange descent equation gives us the coupled partial differential equation

$$\frac{\partial x}{\partial t} = -\left(\frac{\partial L}{\partial x} - \frac{\partial}{\partial s}\left(\frac{\partial L}{\partial \dot{x}}\right)\right)$$
$$\frac{\partial y}{\partial t} = -\left(\frac{\partial L}{\partial y} - \frac{\partial}{\partial s}\left(\frac{\partial L}{\partial \dot{y}}\right)\right)$$

where L is given by the integrand of the functional. This is now a simple calculation. Note that the dot notation for derivative indicates differentiation in the s variable. We have

$$\begin{aligned} \frac{\partial x}{\partial t} &= \frac{\partial}{\partial s}\left(\frac{\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}}\right) \\ &= \frac{\ddot{x}\left(\dot{x}^2 + \dot{y}^2\right) - \dot{x}\left(\dot{x}\ddot{x} + \dot{y}\ddot{y}\right)}{\left(\dot{x}^2 + \dot{y}^2\right)^{3/2}} \\ &= \dot{y}\frac{\ddot{x}\dot{y} - \dot{x}\ddot{y}}{\left(\dot{x}^2 + \dot{y}^2\right)^{3/2}} \end{aligned}$$

Similarly, for $y(s,t)$ we have

$$\frac{\partial y}{\partial t} = \dot{x}\frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\left(\dot{x}^2 + \dot{y}^2\right)^{3/2}}$$

Putting everything together we have

$$\frac{\partial \vec{\gamma}}{\partial t} = \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}\right) = \kappa\left(-\dot{y}, \dot{x}\right)$$

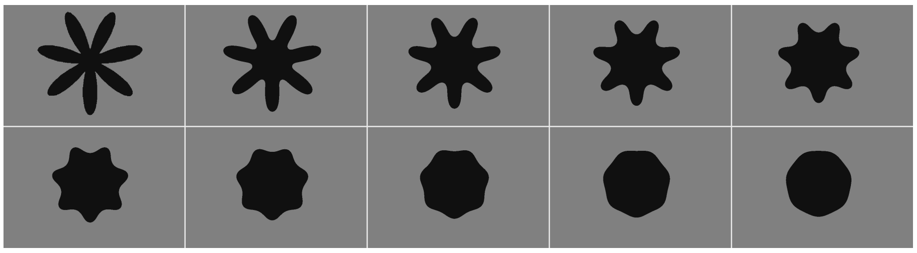

Figure 3: Mean Curvature Flow

where κ is the curvature of $\vec{\gamma}$ given by

$$\kappa = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}$$

Note that the vector $(-\dot{y}, \dot{x})$ is orthogonal to the tangent vector $(\dot{x}, \dot{y})$. If we extend these vectors into $\mathbb{R}^3$, we see that the cross product $(\dot{x}, \dot{y}, 0) \times (-\dot{y}, \dot{x}, 0) = (0, 0, \dot{x}^2 + \dot{y}^2)$ is oriented in the positive z direction. If we take the positive curve orientation and use the right-hand rule, we see that $(-\dot{y}, \dot{x})$ is the inward pointing normal to the curve. By letting $\vec{N} = (-\dot{y}, \dot{x})$, we can write the Euler-Lagrange descent equation as

$$\frac{\partial \vec{\gamma}}{\partial t} = \kappa \vec{N}$$

This is commonly termed mean curvature flow or "curve shrinking", and is illustrated in Figure 3.

4.1.2 Minimizing Surface Integrals

We know how to use the Euler-Lagrange equations to minimize contour integrals, but many important problems are posed with surface integrals, so we will now show how to apply the Euler-Lagrange equations to surface integrals. The main idea is that we use Green's Theorem to convert the surface integral to a contour integral and write the Euler-Lagrange equations for the contour integral. For some $F : \mathbb{R}^2 \mapsto \mathbb{R}$, we will consider functionals A_L of the form

$$A_L[\vec{\gamma}] = \int_{R_{\vec{\gamma}}} F(x, y) \mathrm{d}A \tag{4.6}$$

For some $P : \mathbb{R}^2 \mapsto \mathbb{R}$ and $Q : \mathbb{R}^2 \mapsto \mathbb{R}$ recall Green's Theorem relating surface integrals and contour integrals

$$\oint_{\vec{\gamma}} P \mathrm{d}x + Q \mathrm{d}y = \int_{R_{\vec{\gamma}}} \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) \mathrm{d}x \mathrm{d}y \tag{4.7}$$

Now, let $(x_0, y_0) \in \mathbb{R}^2$ and consider the functions $P : \mathbb{R}^2 \mapsto \mathbb{R}$ and $Q : \mathbb{R}^2 \mapsto \mathbb{R}$ defined by

$$\begin{aligned} P(x,y) &= -\frac{1}{2}\int_{y_0}^{y} F(x,z)\mathrm{d}z \\ Q(x,y) &= \frac{1}{2}\int_{x_0}^{x} F(z,y)\mathrm{d}z \end{aligned}$$

Note that this construction lets us express F as

$$F(x,y) = \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}$$

Now, using Green's Theorem (4.7), we see that

$$\begin{aligned} A_L[\vec{\gamma}] &= \int_{R_{\vec{\gamma}}} F(x,y)\mathrm{d}A \\ &= \int_{R_{\vec{\gamma}}} \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right)\mathrm{d}x\mathrm{d}y \\ &= \oint_{\vec{\gamma}} P\mathrm{d}x + Q\mathrm{d}y \\ &= \int_0^1 \Big(P(x(s),y(s))\dot{x}(s) + Q(x(s),y(s))\dot{y}(s)\Big)\mathrm{d}s \end{aligned}$$

We can now apply the Euler-Lagrange equations noting that $L = P(x(s),y(s))\dot{x}(s) + Q(x(s),y(s))\dot{y}(s)$. We have

$$\begin{aligned} \frac{\partial x}{\partial t} &= -\left(\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{\partial L}{\partial \dot{x}}\right)\right) \\ &= -\left(\frac{\partial P}{\partial x}\dot{x} + \frac{\partial Q}{\partial x}\dot{y} - \frac{\mathrm{d}}{\mathrm{d}s}(P)\right) \\ &= -\left(\frac{\partial P}{\partial x}\dot{x} + \frac{\partial Q}{\partial x}\dot{y} - \frac{\partial P}{\partial x}\dot{x} - \frac{\partial P}{\partial y}\dot{y}\right) \\ &= -\left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right)\dot{y} \\ &= -F\dot{y} \end{aligned}$$

Similarly, for $\frac{\partial y}{\partial t}$ we have

$$\frac{\partial y}{\partial t} = F\dot{x}$$

This gives us the gradient descent curve evolution equation

$$\frac{\partial \vec{\gamma}}{\partial t} = F\vec{N} \tag{4.8}$$

4.2 Level Set Method

The mathematics developed so far allows us to write the Euler-Lagrange descent equations as the equations of motion of a propagating curve. In order to implement this curve evolution we will follow the methods of Sethian [5]. Recall the curve evolution equation from (4.8)

$$\frac{\partial \vec{\gamma}}{\partial t} = F\vec{n} \tag{4.9}$$

where $\vec{n}$ is the normalized inward unit normal and $\vec{\gamma} : [0,1] \mapsto \mathbb{R}^2$. Instead of using an explicit representation of $\vec{\gamma}$, we consider an implicit representation as the zero level set of some function $u : \mathbb{R}^2 \mapsto \mathbb{R}$. We will now consider the curve evolution equations for $\vec{\gamma}$ in this new representation. Recall that we are considering $\vec{\gamma}$ as a family of curves parameterized by t. Likewise, we take $(u(\cdot,\cdot,t))_{t\in\mathbb{R}^+}$ to be a family of surfaces parameterized by t. As $\vec{\gamma}$ is the zero level set of u, we require that

$$u(\vec{\gamma}(s,t),t) = 0 \quad \forall s \in [0,1], t \in \mathbb{R}^+ \tag{4.10}$$

We will use the convention that $u > 0$ on the interior of $\vec{\gamma}$ and $u < 0$ on the exterior of $\vec{\gamma}$. To simplify notation the rest of the calculations will assume everything is evaluated at $\vec{\gamma}$, the zero-level set of u, and that $t \in \mathbb{R}^+$. By differentiating (4.10) with respect to t we get

$$\frac{\partial u}{\partial t} + \vec{\nabla} u \cdot \frac{\partial \vec{\gamma}}{\partial t} = 0 \tag{4.11}$$

Now, substituting (4.9) into (4.11) we get

$$\frac{\partial u}{\partial t} = -\vec{\nabla} u \cdot \frac{\partial \vec{\gamma}}{\partial t} = -F\left(\vec{\nabla} u \cdot \vec{n}\right) \tag{4.12}$$

Now, because $\vec{\gamma}$ is a level set of u we have that $\vec{\nabla} u$ is perpendicular to the curve $\vec{\gamma}$. Also, since we have chosen u to be positive on the interior of $\vec{\gamma}$ and negative on the exterior, $\vec{\nabla} u$ is a non-negative scalar multiple of the unit normal vector $\vec{n}$. Therefore, we can write equation (4.12) as

$$\frac{\partial u}{\partial t} = -F\left\|\vec{\nabla} u\right\| \tag{4.13}$$

Equation (4.13) is the curve evolution equation for $\vec{\gamma}$ embedded in u.

4.2.1 Curvature in the Level Set Formulation

In the previous formulation, the function F was assumed to be independent of $\vec{\gamma}(s) = (x(s), y(s))$. In the case of mean curvature flow, we have the curve evolution equations

$$\frac{\partial \vec{\gamma}}{\partial t} = \kappa \vec{N}$$

where κ is given by

$$\kappa = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\left(\dot{x}^2 + \dot{y}^2\right)^{3/2}}$$

Clearly, κ is dependent on the curve $\vec{\gamma}$. In order to use the level set method, we must express κ in terms of the level set function u. As $\vec{\nabla}u$ is perpendicular to $\vec{\gamma}$ we can write

$$\vec{\nabla}u \cdot \dot{\vec{\gamma}} = \frac{\partial u}{\partial x}\dot{x} + \frac{\partial u}{\partial y}\dot{y} = 0$$

Differentiating with respect to s, we can express this relation as

$$\left(\frac{\partial^2 u}{\partial x^2}\dot{x} + \frac{\partial^2 u}{\partial x \partial y}\dot{y}\right)\dot{x} + \frac{\partial u}{\partial x}\ddot{x} + \left(\frac{\partial^2 u}{\partial x \partial y}\dot{x} + \frac{\partial^2 u}{\partial y^2}\dot{y}\right)\dot{y} + \frac{\partial u}{\partial y}\ddot{y} = 0$$

By re-arranging terms in the above expression, we obtain

$$\vec{\nabla}u \cdot \ddot{\vec{\gamma}} = -\dot{\vec{\gamma}}^T H_u \dot{\vec{\gamma}} \tag{4.14}$$

where H_u is the Hessian of u defined by

$$H_u = \begin{pmatrix} \frac{\partial^2 u}{\partial x^2} & \frac{\partial^2 u}{\partial x \partial y} \\ \frac{\partial^2 u}{\partial x \partial y} & \frac{\partial^2 u}{\partial y^2} \end{pmatrix}$$

If we assume $\vec{\gamma}$ is parameterized at unit speed then by definition we have $\ddot{\vec{\gamma}} = \kappa \vec{n}$. In addition, we have already shown that $\vec{\nabla}u \cdot \vec{n} = \left\|\vec{\nabla}u\right\|$ so we have

$$\vec{\nabla}u \cdot \ddot{\vec{\gamma}} = \kappa \vec{\nabla}u \cdot \vec{n} = \kappa \left\|\vec{\nabla}u\right\| \tag{4.15}$$

Now, let $\vec{\nabla}u^{\perp} = (-u_y, u_x)$ so $\vec{\nabla}u^{\perp}$ is perpendicular to $\vec{\nabla}u$, and thus a scalar multiple of $\dot{\vec{\gamma}}$. Since we assumed $\vec{\gamma}$ is parameterized with unit speed, we have

$$\dot{\vec{\gamma}} = \frac{\pm\vec{\nabla}u^{\perp}}{\left\|\vec{\nabla}u\right\|} \tag{4.16}$$

where the relative directions of $\vec{\gamma}$ and $\vec{\nabla}^{\perp}$ will be irrelevant in the following steps. Substituting (4.16) and (4.15) into (4.14) and re-arranging we get

$$\kappa = -\frac{\vec{\nabla}u^{\perp T} H_u \vec{\nabla}u^{\perp}}{\left\|\vec{\nabla}u\right\|^3} \tag{4.17}$$

Multiplying out the expression above gives

$$\kappa = \frac{u_{xx}u_y^2 - 2u_x u_y u_{xy} + u_{yy}u_x^2}{\left(u_x^2 + u_y^2\right)^{\frac{3}{2}}} \tag{4.18}$$

This is the expression that will be used in our implementation.

5 Design and Implementation

Now that we have introduced the background mathematics, we are equipped to discuss our design. We first derive the Euler-Lagrange descent equations for each term in our tracking functional.

5.1 Local Tracking Descent Equations

The first two terms in our tracking functional (4.2) can be written as

$$\begin{aligned} E[R_{\vec{\gamma}}] = \int_{R_{\vec{\gamma}}} \Bigg(&\min_{\tau \in B(r): \mathbf{x}+\tau \in R_0} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \\ &- \min_{\tau \in B(r): \mathbf{x}+\tau \in R_0^c} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \Bigg) \mathrm{d}\mathbf{x} \\ &+ \int_{\Omega} \min_{\tau \in B(r): \mathbf{x}+\tau \in R_0^c} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \, \mathrm{d}\mathbf{x} \end{aligned}$$

where Ω is the domain of the image. Clearly the last term will not affect the Euler-Lagrange descent equations as it is independent of $\vec{\gamma}$. Using equation (4.8) we can write down the Euler-Lagrange descent equations for the local tracking term as

$$\begin{aligned} \frac{\partial \vec{\gamma}}{\partial t} = -\Bigg(&\min_{\tau \in B(r): \mathbf{x}+\tau \in R_0^c} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \\ &- \min_{\tau \in B(r): \mathbf{x}+\tau \in R_0} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \Bigg) \vec{N} \end{aligned} \tag{5.1}$$

5.2 Kullback-Leibler Descent Equations

We now consider the descent equations for the Kullback-Leibler distance terms in our functional. This derivation borrows much of the notation and methods from [6]. Let I be some image with domain Ω and range Z. Let q be some probability density function over Z, and let $p(\cdot, \vec{\gamma})$ be the probability density function associated with the region $R_{\vec{\gamma}}$. We first need to express p in terms of the curve $\vec{\gamma}$. For $\mathbf{x} \in \Omega$ and $z \in Z$, we can express the cumulative density function as

$$P(I(\mathbf{x}) \leq z) = \frac{\int_{R_{\vec{\gamma}}} u(z - I(\mathbf{x})) \mathrm{d}\mathbf{x}}{\int_{R_{\vec{\gamma}}} \mathrm{d}\mathbf{x}}$$

By differentiating with respect to z we obtain the probability density function

$$\begin{aligned} p(z;\vec{\gamma}) &= \frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{\int_{R_{\vec{\gamma}}} u\left(z - I(\mathbf{x})\right)\mathrm{d}\mathbf{x}}{\int_{R_{\vec{\gamma}}} \mathrm{d}\mathbf{x}}\right) \\ &= \frac{\int_{R_{\vec{\gamma}}} \delta\left(z - I(\mathbf{x})\right)\mathrm{d}\mathbf{x}}{\int_{R_{\vec{\gamma}}} \mathrm{d}\mathbf{x}} \\ &= \frac{N(R_{\vec{\gamma}}, z)}{A(R_{\vec{\gamma}})} \end{aligned}$$

where u is the unit step function, δ is the dirac distribution, and N and A are defined as

$$N(R, z) = \int_R \delta(z - I(\mathbf{x}))\mathrm{d}\mathbf{x}$$

$$A(R) = \int_R \mathrm{d}\mathbf{x}$$

Now, the Kullback-Leibler distance, as a functional over the space of closed curves, can be written as

$$E[\vec{\gamma}] = D(q(\cdot)||p(\cdot, \vec{\gamma})) = \int_Z q(z)\ln\frac{q(z)}{p(z;\vec{\gamma})}dz = -h(q) - \int_Z q(z)\ln p(z;\vec{\gamma})dz$$

where $h(\cdot)$ is differential entropy. Since the differential entropy of q does not depend on $\vec{\gamma}$ we can restrict ourselves to the functional

$$\begin{aligned} K[\vec{\gamma}] &= -\int_Z q(z)\ln(p(z;\vec{\gamma}))dz \\ &= \ln(A(R_{\vec{\gamma}}))\int_Z q(z)dz - \int_Z q(z)\ln(N(R_{\vec{\gamma}}, z))dz \\ &= \ln(A(R_{\vec{\gamma}})) - \int_Z q(z)\ln(N(R_{\vec{\gamma}}, z))dz \end{aligned}$$

Now, we have

$$\frac{\delta K}{\delta\vec{\gamma}} = \frac{1}{A(R_{\vec{\gamma}})}\frac{\delta A(R_{\vec{\gamma}})}{\delta\vec{\gamma}} - \int_Z q(z)\left[\frac{1}{N(R_{\vec{\gamma}}, z)}\frac{\delta N(R_{\vec{\gamma}}, z)}{\delta\vec{\gamma}}\right]dz$$

The next step can be understood by examining the Euler-Lagrange descent equations for $A(R_{\vec{\gamma}})$ and $N(R_{\vec{\gamma}}, z)$. Alternatively, there is a proposition that yields the same result in [6]. What we get is

$$\begin{aligned} A(R_{\vec{\gamma}}) &= \int_{R_{\vec{\gamma}}} \mathrm{d}\mathbf{x} \Rightarrow \frac{\delta A(R_{\vec{\gamma}})}{\delta\vec{\gamma}} = \vec{N} \\ N(R_{\vec{\gamma}}, z) &= \int_{R_{\vec{\gamma}}} \delta(z - I(\mathbf{x}))\mathrm{d}\mathbf{x} \Rightarrow \frac{\delta N(R_{\vec{\gamma}}, z)}{\delta\vec{\gamma}} = \delta(z - I(\mathbf{x}))\vec{N} \end{aligned}$$

By substituting these expressions, we get

$$\begin{aligned}\frac{\delta K}{\delta \vec{\gamma}} &= \left[\frac{1}{A(R_{\vec{\gamma}})} - \int_Z q(z)\frac{\delta(z - I(\mathbf{x}))}{N(R_{\vec{\gamma}}, z)}\mathrm{d}\mathbf{x}\right]\vec{N} \\ &= \left[\frac{1}{A(R_1)} - \frac{q(I(\mathbf{x}))}{N(R_{\vec{\gamma}}, I(\mathbf{x}))}\right]\vec{N} \\ &= \frac{p(I(\mathbf{x})) - q(I(\mathbf{x}))}{N(R_{\vec{\gamma}}, I(\mathbf{x}))}\vec{N}\end{aligned}$$

Finally, the gradient descent equations for the Kullback-Leibler distance are

$$\frac{\partial \vec{\gamma}}{\partial t} = \frac{q(I(\mathbf{x})) - p(I(\mathbf{x}))}{N(R_{\vec{\gamma}}, I(\mathbf{x}))}\vec{N} \tag{5.2}$$

where N is the inward normal to the curve $\vec{\gamma}$.

5.3 Euler-Lagrange Descent Equations

We are now in a position to write down the Euler-Lagrange descent equations for our tracking functional which is reproduced here.

$$\begin{aligned}R_{\vec{\gamma}} \mapsto E[R_{\vec{\gamma}}] = \lambda_1 &\int_{R_{\vec{\gamma}}} \min_{\tau \in B(r): \mathbf{x}+\tau \in R_0} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\|\, d\mathbf{x} \\ &+ \lambda_1 \int_{R^c_{\vec{\gamma}}} \min_{\tau \in B(r): \mathbf{x}+\tau \in R^c_0} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\|\, d\mathbf{x} \\ &+ \lambda_2 D(f_{I_0,R_0} || f_{I_1,R_{\vec{\gamma}}}) \\ &+ \lambda_3 D(f_{I_0,R^c_0} || f_{I_1,R^c_{\vec{\gamma}}}) \\ &+ \lambda_4 \oint_{\vec{\gamma}} ds\end{aligned}$$

The Euler-Lagrange Descent Equations are given by

$$\begin{aligned}\frac{\partial \vec{\gamma}}{\partial t} = -\Bigg(&\lambda_1 \min_{\tau \in B(r): \mathbf{x}+\tau \in R^c_0} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \\ &-\lambda_1 \min_{\tau \in B(r): \mathbf{x}+\tau \in R_0} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \\ &+\lambda_2 \frac{f_{I_0,R_0}(I_1(\mathbf{x})) - f_{I_1,R_{\vec{\gamma}}}(I_1(\mathbf{x}))}{N(R_{\vec{\gamma}}, I_1(\mathbf{x}))} \\ &-\lambda_3 \frac{f_{I_0,R^c_0}(I_1(\mathbf{x})) - f_{I_1,R^c_{\vec{\gamma}}}(I_1(\mathbf{x}))}{N(R^c_{\vec{\gamma}}, I_1(\mathbf{x}))} - \lambda_4 \kappa(\mathbf{x})\Bigg)\vec{N}\end{aligned} \tag{5.3}$$

where $\vec{N}$ is the inward pointing normal vector to $\vec{\gamma}$.

5.4 Level Set Descent Equations

The level set gradient descent equations for our tracking functional can be written directly from equations (5.3) and (4.13)

$$\begin{aligned}\frac{\partial u}{\partial t} = \Bigg(&\lambda_1 \min_{\tau \in B(r):\mathbf{x}+\tau \in R_0^c} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \\ &-\lambda_1 \min_{\tau \in B(r):\mathbf{x}+\tau \in R_0} \|\mathbf{v}(\mathbf{x}, I_1) - \mathbf{v}(\mathbf{x}+\tau, I_0)\| \\ &+\lambda_2 \frac{f_{I_0,R_0}(I_1(\mathbf{x})) - f_{I_1,R_{\vec{\gamma}}}(I_1(\mathbf{x}))}{N(R_{\vec{\gamma}}, I_1(\mathbf{x}))} \\ &-\lambda_3 \frac{f_{I_0,R_0^c}(I_1(\mathbf{x})) - f_{I_1,R_{\vec{\gamma}}^c}(I_1(\mathbf{x}))}{N(R_{\vec{\gamma}}^c, I_1(\mathbf{x}))} - \lambda_4 \kappa(\mathbf{x})\Bigg) \left\|\vec{\nabla} u\right\|\end{aligned} \tag{5.4}$$

where the curvature κ is given by equation (4.18).

5.5 Discretization

Recall equation (4.18) for the curvature of $\vec{\gamma}$ in the level set formulation

$$\kappa = \frac{u_{xx}u_y^2 - 2u_x u_y u_{xy} + u_{yy}u_x^2}{(u_x^2 + u_y^2)^{\frac{3}{2}}} \tag{5.5}$$

To discretize equation (4.18), we use the central difference approximation of the partial derivatives which are shown here.

$$\begin{aligned} u_x &= \frac{u_{i+1,j} - u_{i-1,j}}{2} \\ u_y &= \frac{u_{i,j+1} - u_{i,j-1}}{2} \\ u_{xx} &= u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \\ u_{yy} &= u_{i,j+1} - 2u_{i,j} + u_{i,j-1} \\ u_{xy} &= \frac{u_{i+,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4} \end{aligned}$$

For the discretization of the gradient of u, we use the methods of Sethian in [5]. In order to have the numerical domain of dependence align with the mathematical domain of dependence, the following numerical scheme from [5] is used. If our functional at a point (i, j) is given by $F_{i,j}$, then we write the gradient at that point as

$$\left\|\vec{\nabla} u\right\| = \begin{cases} \nabla^+, & \text{if } \; F_{i,j} > 0 \\ \nabla^-, & \text{if } \; F_{i,j} \leq 0 \end{cases}$$

where ∇^+ uses the forward difference approximation when the derivative is negative and the backwards difference approximation when the derivative is positive. ∇^- is defined exactly opposite of ∇^+. ∇^+ and ∇^- are easily computed in terms of the level set function u. The reader is referred to [5] for more details on this topic.

5.6 Characteristic Vector

The characteristic vector holds invariant image characteristics. Tracking performance can be improved if the characteristic vector is expanded to include more invariant image characteristics, allowing for a more accurate matching of regions between frames.

During our initial testing phase the characteristic vector simply consisted of 0^{th} order terms, namely the RGB pixel intensity values. Although tracking with only these 0^{th} order invariants was often quite successful, in order to improve tracking performace, higher order invariant image characteristics are desired. During a project meeting in January 2008, Professor Mansouri suggested viewing the image as a 3-dimensional surface [7], and including the principal cirvatures and directions of the surface in the characteristic vector. This idea is based on the assumption that the region of interest undergoes (locally) Euclidean transformations between subsequent frames, and hence the principal curvature values will be preserved.

5.6.1 Principal Curvatures

In order to calculate principal directions and principal curvatures, the image must first be converted into a 3-dimensional surface. This can be done via a least squares fitting on the graph of the image. This was done by first converting the image from RGB to YCbCr, considering only the Y intensity values, and locally performing least squares fittings on the graph of these intensities. Using these surfaces the corresponding principal curvatures and directions can be calculated for each pixel in the image. As the principal directions at any given point on a surface are always orthogonal[8], we now include three additional invariants in the characteristic vector: principal direction, maximum principal curvature, and minimum principal curvature. Note that it is not assumed in general that the principal directions remain invariant, however under certain circumstances this measure can be useful.

Although the mathematics behind principal curvature is summarized in Section 5.6.4, it is beneficial to first provide a visual explanation of the concept. Consider the image in Figure 4.a. Zoom in on the image and focus on the specified 3-by-3 window. Let the graph in Figure 4.b correspond to the Y intensities in this window. Finally, let the 3-dimensional surface in Figure 4.c correspond to a least squares fitting of the graph in Figure 4.b. The principal directions for the middle pixel are shown on the surface in Figure 4.c. The directions are necessarily orthogonal[8], and correspond to the directions of maximum and minimum curvature. For each pixel in the

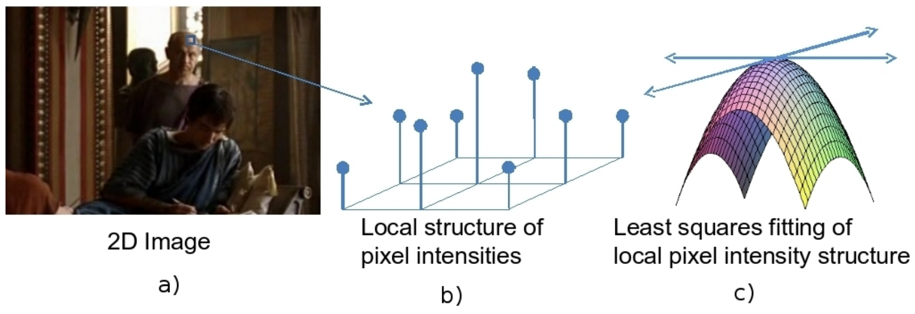
2D Image
a)
Local structure of pixel intensities
b)
Least squares fitting of local pixel intensity structure
c)

Figure 4: a) Input Image. b) Local structure of pixel intensities. c) Least squares surface approximation.

image we will calculate and store one principal direction (due to orthogonality), and the maximum and minimum curvatures. These values can then be used in the characteristic vector for tracking purposes. Figure 5 displays the an image together with a gray-scale image of the maximum curvature value at each pixel. By inspection, it is clear that much of the image structure is maintained in the maximum curvature output. Hence, such an invariant measure may be used to supplement the 0^{th} order measures. Note that we

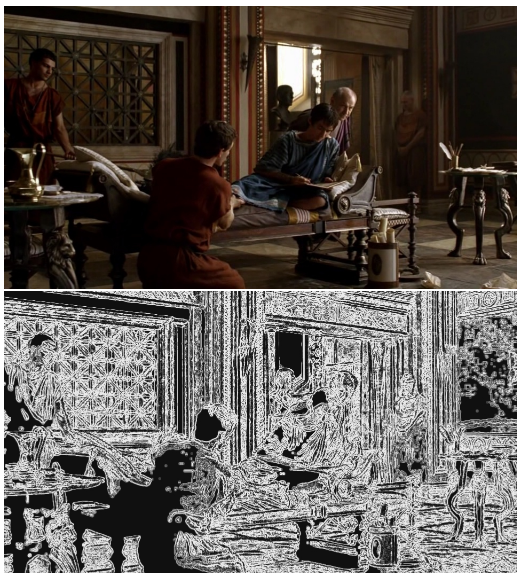

Figure 5: Gray-scale of maximum principal value at each pixel

can view the principal direction as a 1^{st} order image characteristic, and the principal curvature values as 2^{nd} order image characteristics. Before we discuss the mathematics, we will introduce least squares theory and how we have implemented the least squares approximation.

5.6.2 Least Squares Theory

Viewing an image as a 3-dimensional surface, where the height at each point is a function of the brightness at that point, is a useful technique to extcan-canract local properties from the image. The idea of viewing an image as a surface was first discussed in [7]. The theory of least squares will enable us to quickly and efficiently fit a polynomial surface locally around a given pixel. From this point, we can use the coefficients of this fitting to extract meaningful local properties from the image that may improve our tracking algorithm. This section provides a brief overview of some relevant elements of least squares theory.

Let $\Omega \subset \mathbb{R}^2$ and $f : \Omega \mapsto \mathbb{R}$ be a function with domain Ω. Let $D = \{(x_k, y_k)\}_{k=0}^N$ be a finite set of points contained in Ω. We are interested in approximating f as

$$f(x,y) \approx \sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij}x^i y^j$$

while minimizing the sum of the squared error of all points in D. Thus we are trying to minimize

$$R = \sum_{k=0}^{N}\left(f(x_k,y_k) - \sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij}x_k^i y_k^j\right)^2$$

over all sets of coefficients $\{a_{ij}\}_{0\le i,j\le n}$. Since R is a continuously differentiable function of the coefficients, any set of minimal coefficients, $\{a_{ij}^*\}_{0\le i,j\le n}$, must satisfy

$$\begin{aligned}\frac{\partial R}{\partial a_{rs}} &= \frac{\partial}{\partial a_{rs}}\sum_{k=0}^{N}\left(f(x_k,y_k) - \sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij}x_k^i y_k^j\right)^2 \\ &= \sum_{k=0}^{N}\left(-2f(x_k,y_k)x_k^r y_k^s + 2\left(\sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij}^* x_k^i y_k^j\right)x_k^r y_k^s\right) = 0\end{aligned}$$

for every coefficient a_{rs}. Rearranging this expression gives us

$$\sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij}^* \sum_{k=0}^{N} x_k^{r+i} y_k^{s+j} = \sum_{k=0}^{N} f(x_k,y_k)x_k^r y_k^s$$

which is a linear equation over the set of coefficients $\{a_{ij}^*\}_{0\le i,j\le n}$. Since this equation must hold for $0 \le r,s \le n$, we have a linear system of $(n+1)^2$ equations and $(n+1)^2$ unknowns. Let $M(D,n)$ be the matrix associated with this linear system. The least squares approximation is performed by solving this linear system for the coefficients.

5.6.3 Least Squares Implementation

As $M(D, n)$ does not depend on the values f takes on, we have pre-computed $M(D, n)$ and $M^{-1}(D, n)$ for all values of D and n that are deemed reasonable. Then, only the right side of equation 5.6.2 needs to be computed at run-time which greatly improves efficiency. As we are trying to approximate images, we have taken the set of sample points D to be the grid of pixels centered at the pixel of interest. Thus for a pixel at position (x, y), D is given by

$$D = \{(x + i, y + j) \mid i, j \in \{-N, -N + 1, \cdots, N - 1, N\}\}$$

Thus, the window of pixels that are considered in the least squares fitting is $2N \times 2N$ centered at the pixel of interest. We have pre-computed $M^{-1}(D, n)$ for $2 \leq n \leq 5$ and $n \leq 2N \leq 15$.

5.6.4 Principal Curvature Theory

For a parameterized surface $\mathbf{x} = (x_1(u, v), x_2(u, v), x_3(u, v))$ it can be shown [8] that independent of parameterization , the *normal curvature* at a point on the surface, κ, in direction dv/du, is given by the equation

$$\kappa = \frac{II}{I} = \frac{(\mathbf{x}_u \cdot \vec{N}_u)du^2 + (\mathbf{x}_u \cdot \vec{N}_v + \mathbf{x}_v \cdot \vec{N}_u)dudv + (\mathbf{x}_v \cdot \mathbf{N}_v)dv^2}{(\mathbf{x}_u \cdot \mathbf{x}_u)du^2 + 2(\mathbf{x}_u \cdot \mathbf{x}_v)dudv + (\mathbf{x}_v \cdot \mathbf{x}_v)dv^2}$$

where I and II are the *first* and *second fundamental forms* of the surface, respectively [8]. Now, consider the surface given by the equation

$$f(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{n} a_{ij} u^i v^j$$

which corresponds to a least squares fitting of an image. For surfaces of this form $\mathbf{x} = (u, v, f(u, v))$, we can express the *normal curvature* in direction dv/du as

$$\kappa = \frac{edu^2 + 2fdudv + gdv^2}{Edu^2 + 2Fdudv + Gdv^2}$$

where

$$e = \frac{1}{\rho} f_{uu} \qquad f = \frac{1}{\rho} f_{uv} \quad g = \frac{1}{\rho} f_{vv}$$
$$E = 1 + f_u^2 \quad F = f_u f_v \quad G = 1 + f_v^2$$
$$\rho = \sqrt{f_u^2 + f_v^2 + 1}$$

Now, by letting $\lambda = dv/du$, we can rewrite κ as

$$\kappa = \kappa(\lambda) = \frac{e + 2f\lambda + g\lambda^2}{E + 2F\lambda + G\lambda^2}$$

The extreme values of κ can be characterized by $\frac{d\kappa}{d\lambda} = 0$ [8]. Using properties of κ we find the following quadratic equation in λ with real roots

$$(eF - fE) + (eG - gE)\lambda + (fG - gF)\lambda^2 = 0$$

This equation determines the two directions dv/du in which κ obtains its extrema. Unless II vanishes or II and I are proportional, one value must be a minimum and the other a maximum. Thus, the roots λ_1, λ_2 of the quadratic equation are the *directions of principal curvature.*

The normal curvatures in the directions of principal curvature are called the *principal curvatures*, denoted κ_1 and κ_2, and are found by the equations

$$\begin{aligned} \kappa_1 &= \frac{e + 2f\lambda_1 + g\lambda_1^2}{E + 2F\lambda_1 + G\lambda_1^2} \\ \kappa_2 &= \frac{e + 2f\lambda_2 + g\lambda_2^2}{E + 2F\lambda_2 + G\lambda_2^2} \end{aligned} \tag{5.6}$$

5.6.5 FIR Filters

A very natural way to extract local structure from an image is to apply a finite impulse response (FIR) filter to the pixel samples. We have implemented within our code support for appending non-separable FIR filters outputs to our characteristic vector $\mathbf{v}$. The width and height of the filtering window is limited to 5 by 5. Thus the output of the filter for pixel (x, y) is given by,

$$F(x, y) = \sum_{i=1}^{5} \sum_{j=1}^{5} c_{ij} I(x + i - 3, y + j - 3)$$

where the c_{ij}'s are the filter coefficients and $I(x, y)$ is the pixel intensity at position (x, y). Depending on the processing mode, the filter outputs are calculated for either the luminance component only, or all three RGB values. Clearly the challenging problem here is to decide what properties the filter should have in order to provide a good invariant measure of local image structure. Although this has been implemented, it has not been thouroughly experimented with.

5.6.6 Vector Quantization

In our code, the probability density functions are implemented as histograms of the characteristic vector $\mathbf{v}$. Since we cannot assume anything about the dependence between the random variables corresponding to the entries in $\mathbf{v}$, we need one bin in the histogram for every combination of values that $\mathbf{v}$ can take on. Clearly, the number of bins required grows exponentially with the length of $\mathbf{v}$. Given that each entry in $\mathbf{v}$ is represented by an 8-bit integer, we will require 256^n bins for a vector $\mathbf{v}$ of length n. This quickly

becomes ridiculous if $n > 3$ or 4, which leads us to consider some form of quantization.

Let $\Delta = 2^q$ be the step size of our quantizer, where q is a user-controllable parameter in our code. Let the components of $\mathbf{v}$ be given by $(v_1, v_2, \cdots, v_n)$. We will quantize the vector $\mathbf{v}$ by performing uniform scalar quantization on each element,

$$Q(\mathbf{v}) = (Q(v_1), Q(v_2), \cdots, Q(v_3))$$

where the scalar quantizer $Q(\cdot)$ is defined by

$$Q(v_j) = \left\lfloor \frac{v_j}{2^{8-q}} \right\rfloor$$

The idea here is that now only $8 - q$ bits are required to represent each element of $\mathbf{v}$ thereby reducing the number of bins required. So, for example, if $q = 5$, then we are essentially representing each element of $\mathbf{v}$ by 3 bits.

5.7 Choice of Programming Environment

The two logical choices for our software implementation were C and MATLAB. We have chosen C because of its performance advantage over MATLAB. Since the algorithm is very processing intensive it is able to run much faster in C. The main advantage MATLAB has over C is easy image I/O, however we have acquired open-source C libraries which read and write TIFF and BMP images so image I/O is not a problem. We have chosen to implement our algorithm using floating-point arithmetic.

We setup multiple Linux server environments which we used as a shared programming environment, code repository, and algorithm test environment. The Linux environment has provided many advantages, such as centralized code management, improved efficiency, and ease of scripting useful tools to help run, test, and debug code. The server was remotely accessible via a VNC session. Each group member has their own dedicated user account with the ability to work concurrently in the environment.

5.8 Code Description

The region tracking algorithm has been implemented via three major classes: an image class, a discretization class, and a main class. The main class

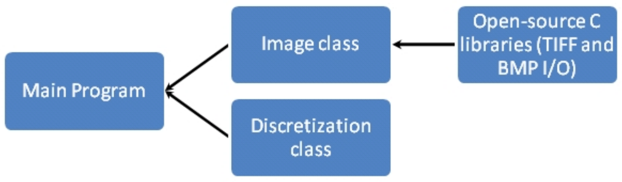

Figure 6: Block Diagram of Code Hierarchy

performs iterations in order to solve the PDE. The image and discretization classes provide the main class with the machinery required to numerically solve the PDE during the iteration process.

A hierarchical representation of the code is displayed in Figure 6. Each of the three classes have been broken down and summarized in detail below.

5.8.1 'Image' Class

The image class is essentially a library of functions required to work with the image files which are taken as input to the program. We have chosen to work exclusively with TIFF and BMP format image files. Thus, any image file taken as input to the algorithm is read via a function which stores all the image data into an image object. The image object holds the various image characteristics, such as image height and width, as well as RGB and/or YCbCr intensity values associated with each pixel. An image object can be written to an output file in either TIFF or BMP format.

On top of providing I/O functionality, the image class also contains various functions for extracting and updating the image characteristics held by the image object. The region tracking algorithm requires the RGB and YCbCr intensity values associated with each pixel in an image, as well as the ability to update these intensity values if necessary. The image class makes use of open source libraries to read and write TIFF images.

5.8.2 'Discretization' Class

The discretization class contains all functions required to calculate our functional, the level set function, and its gradient. These functions are called by the main algorithm on each iteration in order to numerically solve the PDE.

5.8.3 'Main' Class

The main class executes the region tracking algorithm. It makes use of the many functions provided by the image and discretization classes.

When the code is executed, the main program first takes care of any initializations required by the algorithm. The first image input to the program is a mask which allows the region curve to be initialized. This allows the program to initialize the level-set which discriminates between the region and the background.

Following the initialization phase, the main program performs iterations in order to numerically solve the PDE. The number of iterations is chosen large enough to ensure convergence.

The iterative process of numerically solving the level-set PDE is repeated for each frame in the image sequence. The program terminates following the iteration on the final input frame.

"Tracked" frames are output from the main program so a tracking sequence can be analyzed to determine how successful the algorithm was.

5.9 Pseudocode

The following pseudo-code outlines the general procedure which is followed by the main program when executing the region tracking algorithm.

```
Main program
{
   Initialize Input Images();
   Initialize Level Set ();
   Initialize Local Tracking ();
   Initialize Global Tracking ();

   for (each input frame in image sequence)
   {
      Reset Level Set to +1(-1) inside(outside) the region ();
      Initialize Global Tracking For Current Frame ();
      Initialize Local Tracking For Current Frame ();
      for (n iterations and every pixel)
      {
         Gradient Calculation ();
         Global Tracking Calculations ();
         Local Tracking Calculations ();
         Regularization Calculations ();
         Update Level Set (using above calculations);
      }
      Output Resulting Tracked Image Sequence ();
   }
```

6 Results and Observations

We tested our algorithm with the following real video sequences: car, SNL, hockey, face, finger, and bow. In each case, the region of interest is manually specified in the first frame and the algorithm tracks the region throughout the subsequent frames. All sequences are tested with the characteristic vector $\mathbf{v}(\mathbf{x}) = (R(\mathbf{x}), G(\mathbf{x}), B(\mathbf{x}))$ unless otherwise noted. R, G, and B are the red, green, and blue components of the image.

6.1 Car

Figure 7 shows the results with only local tracking enabled (ie: $\lambda_2 = \lambda_3 = 0$) with a search window of $\delta = 15$. The algorithm successfully tracks the car

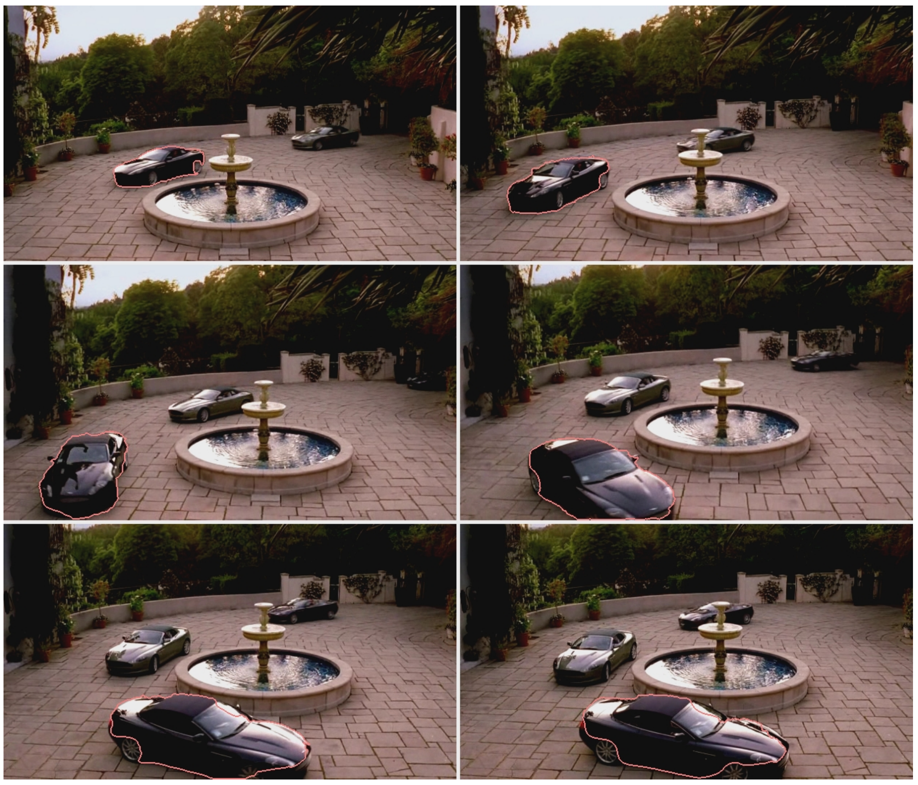

Figure 7: Tracking output of every 15^{th} frame of the car sequence (local tracking)

through the entire sequence of 75 frames. When we first experimented with this sequence, the tracking was quite poor as the back left wheel is very similar in colour to the ground. This would cause the algorithm to take the ground into the region and lose the back edge of the car. This issue was resolved by excluding the back wheel from the region in the first frame. This

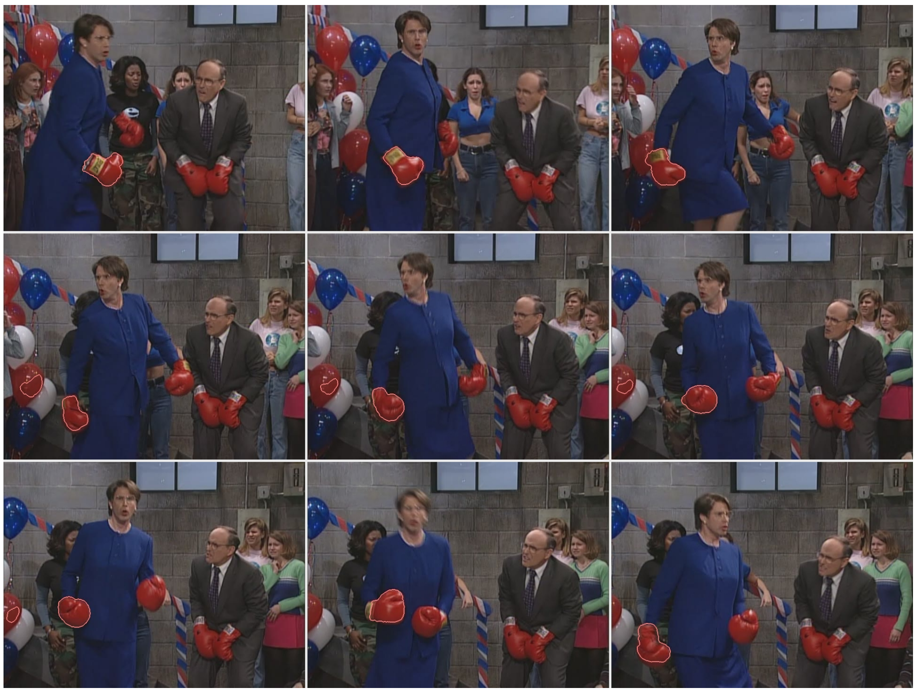

Figure 8: Tracking output of every 6th frame of the SNL sequence (local tracking)

demonstrates how the algorithm can be very sensitive to the initial position of the contour.

6.2 SNL

We have experimented with the SNL sequence using both local and global tracking separately. Figure 8 shows the tracking results for local tracking with a search window of $\delta = 40$. We can see that the region splits into two regions as the boxing glove passes by the red balloon. This illustrates the flexibility of the level set method which allows splitting and merging of regions. Our algorithm successfully tracks the glove through 90 frames until the motion of the glove exceeds the search window between two frames and the algorithm loses track. This demonstrates a limitation of the algorithm; although the motion pattern is not constrained, the distance of motion between frames is assumed to be within the defined search window δ.

We have also experimented with global tracking on the SNL sequence (ie: $\lambda_1 = 0$). The results are shown in Figure 9. Global tracking is less successful on this sequence. We can see that from frame to frame the left boxing glove is moving behind Janet's body. This implies that the proportion of red pixels in the background is decreasing. Since the algorithm is attempting

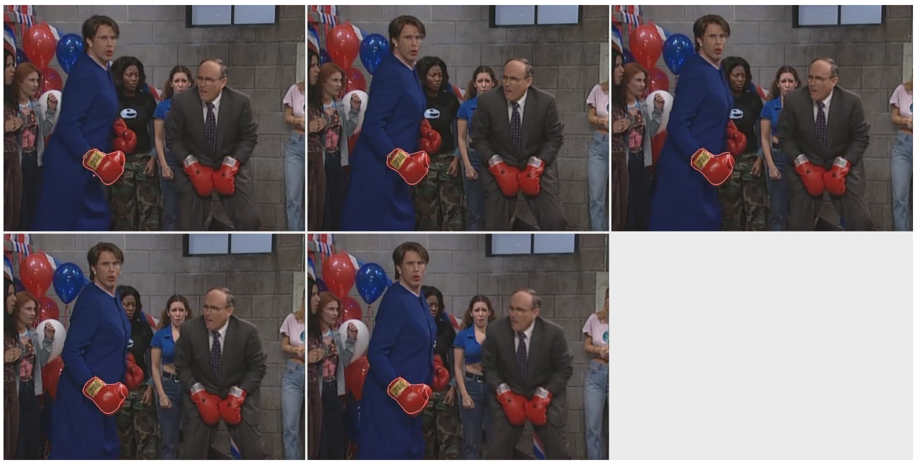

Figure 9: Tracking output of frames 4,5,6,7 and 8 from the SNL sequence (global tracking)

to match the probability density functions representing the foreground and background, it is trying to compensate for this loss by shifting red pixels from the tracked glove into the background. Note that this does not change the probability density function of the region (by very much) and therefore does not affect the foreground matching term. This is a drawback of the global tracking algorithm not present in local tracking.

6.3 Hockey

We have experimented with both local and global tracking separately on the hockey sequence. Figure 10 shows the results for local tracking. We can see that after 10 frames, the algorithm has lost part of the player's leg. This is likely because the ice is very similar in colour to the white stripe on the player's socks. Also, the curvature of the contour is high around the player's skates, so the regularization term overpowers the other terms.

Next, note that in frame 20, the algorithm includes the puck as part of the region. This is likely because the puck "emerged" from the left side of the player a few frames earlier and part of the contour split off from the main region. As the curvature is extremely high on such a small region, the contour immediately shrinks away.

Finally, in frame 90, the algorithm takes the goalie's left pad into the region. Since the local tracking algorithm has no idea of the global statistics of the region, as long as the background looks locally the same, it will be included in the region. We can conclude that local tracking is not very successful on this sequence.

The same hockey sequence was tested with global tracking yielding better results. Here, we used the characteristic vector $\mathbf{v}(\mathbf{x}) = Y(\mathbf{x})$, where $Y(\mathbf{x})$ is

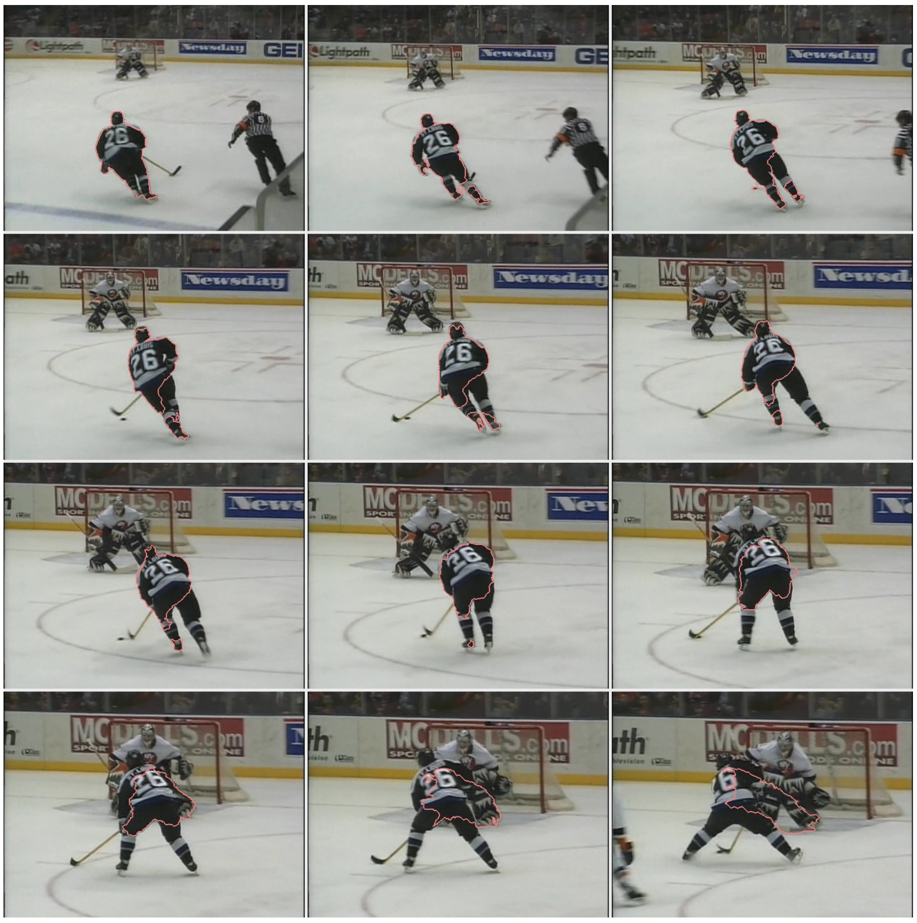
Figure 10: Tracking output of every 10^{th} frame of the hockey sequence (local tracking)

the luminance value at pixel $\mathbf{x}$. Figure 11 shows the tracking output for this sequence. Note that the algorithm has a general idea of where the player is, but does not have a "tight fit" on the player. This is a general property of the global density matching method. Due to the fact that the density of the region encapsulated by the the "loose fit" contour is approximately equal to the density of the player, the algorithm chooses the loose fit because the cost of the regularization term is minimized for this fit.

Also, the global tracking algorithm has the same problem as local tracking at the end of the sequence; the density of the goalie's pad is similar to the player and hence the algorithm cannot distinguish between the two. This problem could be resolved for both local and global tracking by including other measures in the characteristic vector (ie: FIR filter or Principal Curvature), but this was not tested.

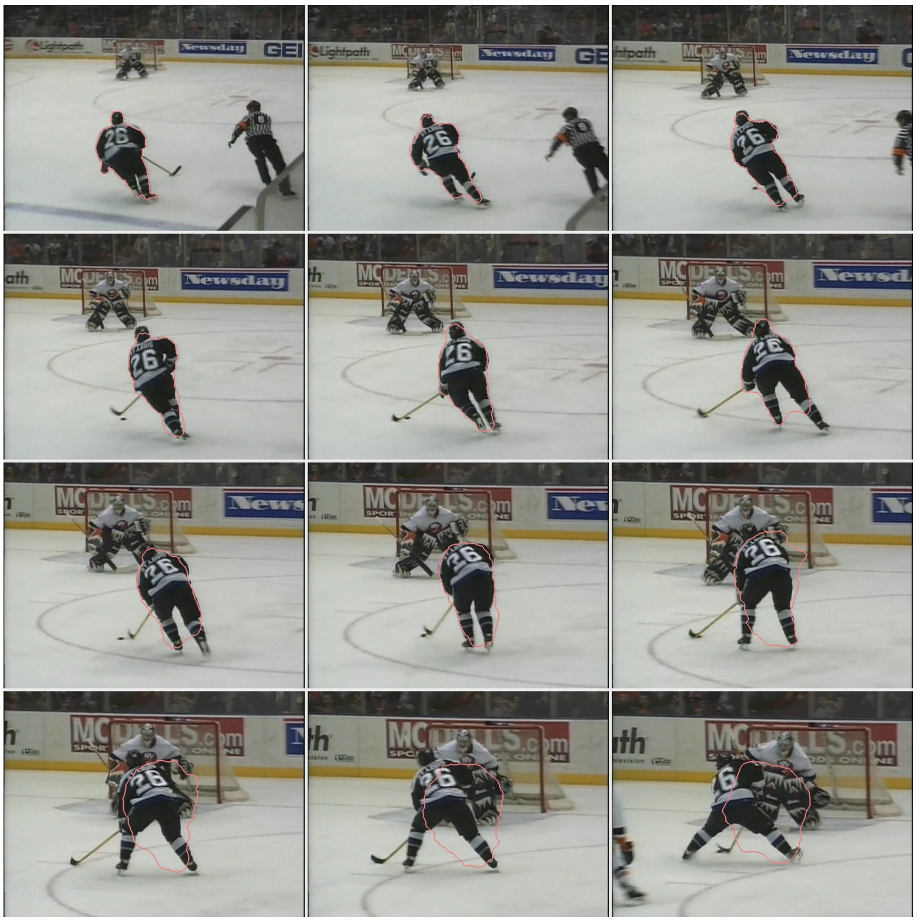

Figure 11: Tracking output of every 10^{th} frame of the hockey sequence (global tracking)

6.4 Face

We have experimented with the local tracking on the face sequence and the results are shown in Figure 12. We used a search window $\delta = 10$. We can see that although the size and intensity boundaries of the region vary throughout the sequence, our algorithm successfully tracks the face through 150 frames.. This demonstrates the general flexibility of our solution.

6.5 Finger

We experimented with local and global tracking on the finger sequence. The results for local tracking with a search window of $\delta = 15$ are shown in Figure 13. We see the same issues with colour similarities between the region and the background as we saw in the hockey sequence. This is the main reason why the algorithm cannot maintain a tight fit on the upper portion of the

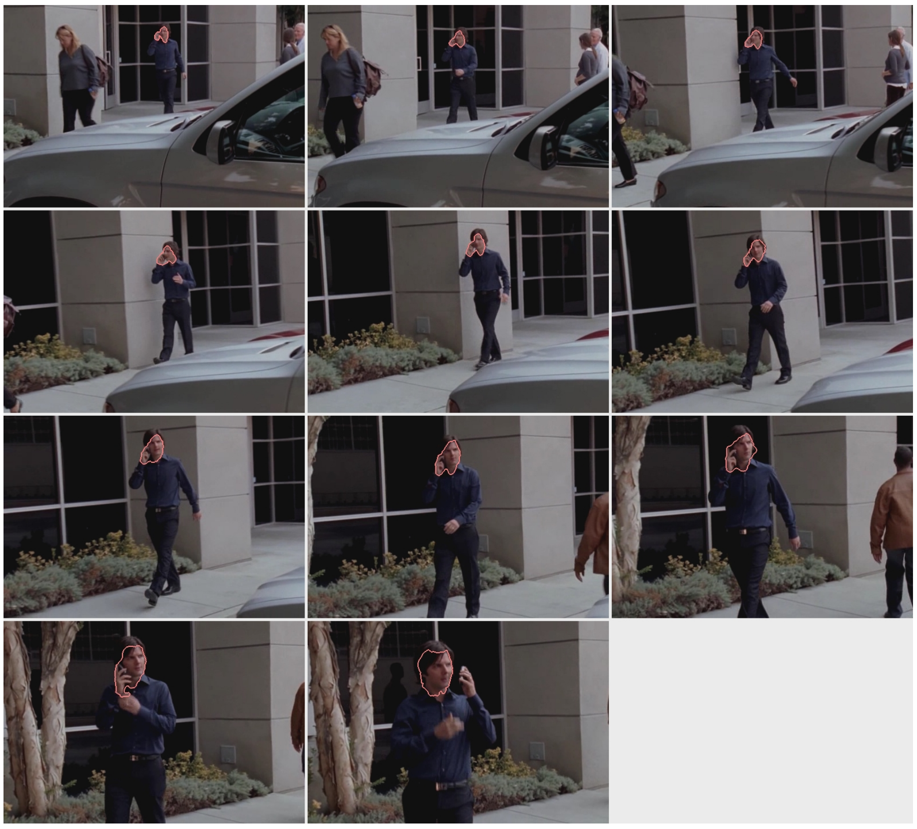

Figure 12: Tracking output of every 15^{th} frame of the face sequence (local tracking)

finger while tracking is accurate on the lower portions of the finger which are in high contrast with his black jacket. The same sequence has also been tested with global tracking and the results are shown in Figure 14. We can see that neither local nor global on their own resulted in successful tracking. However, the local algorithm performs better on this sequence as it keeps track of one edge of the finger. Again, tracking could be improved for either algorithm by intelligently selecting another measure to include in the characteristic vector, but this has not been tested yet.

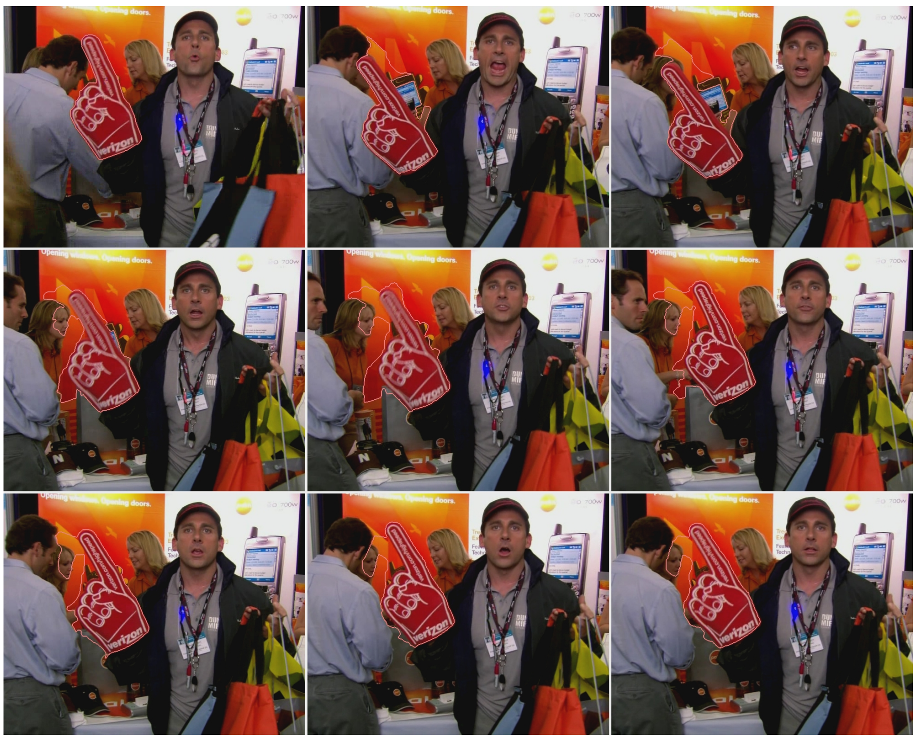

Figure 13: Tracking output of every 10th frame of the finger sequence (local tracking)

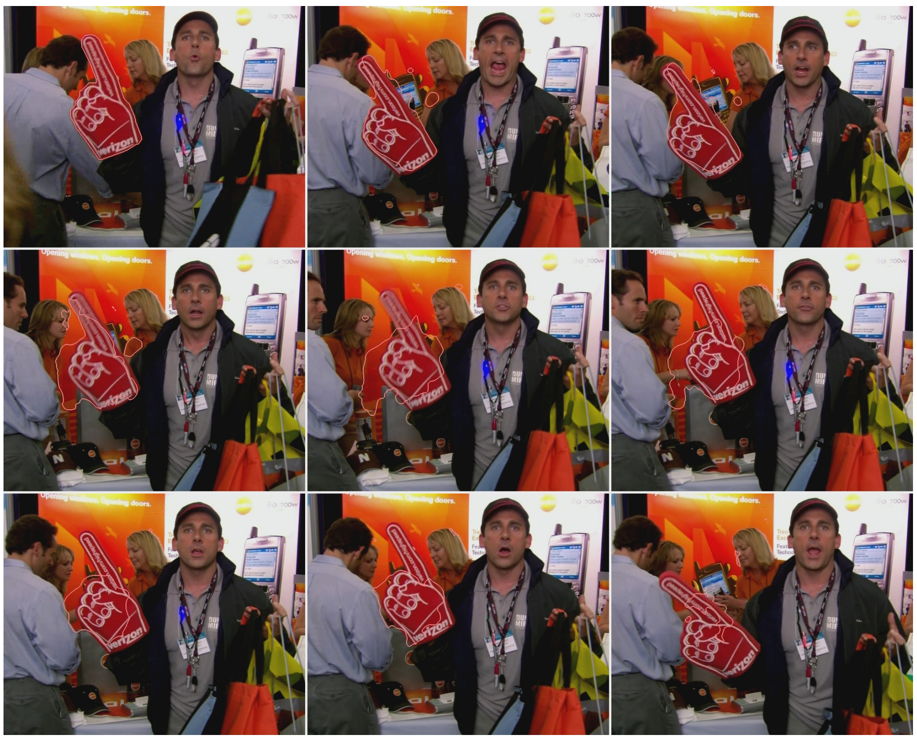

Figure 14: Tracking output of every 10th frame of the finger sequence (global tracking)

6.6 Bow

We experimented with both local and global tracking on the bow sequence. This sequence is difficult to track because of the size of the bow. In general, the regularization term needs to be strong enough to enforce smoothness but not dramatically change the shape of the region. This is very difficult when the region is small as there is already very high curvature along the contour. As we see in Figure 15, the contour shrinks away around frame 40 due to the regularization term. We relaxed the regularization term when using the global tracking algorithm and the results are shown in Figure 16. We can see that the contour no longer shrinks away, however, at frame 50 the region splits into two and takes part of the gift into the region.

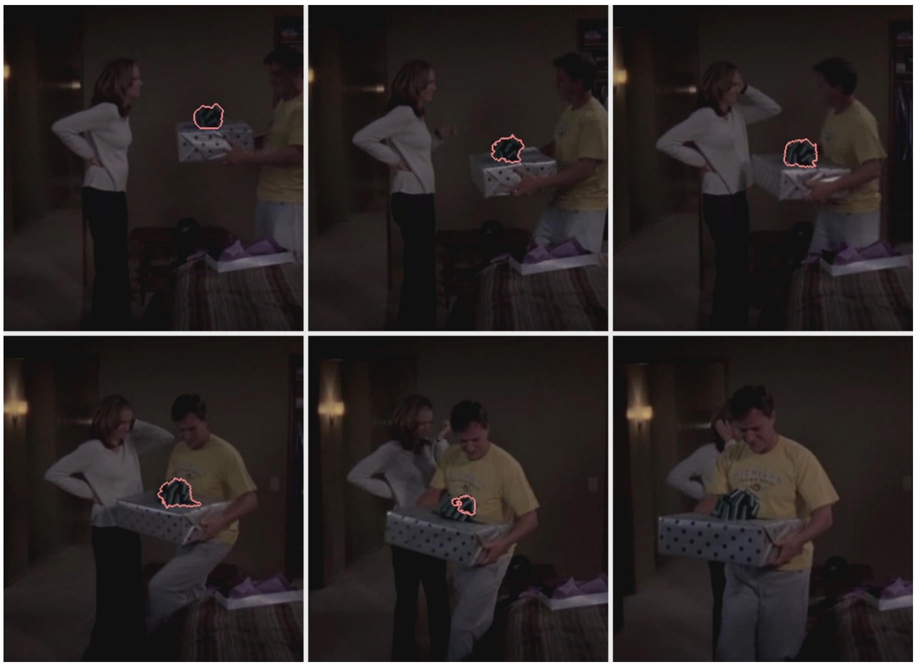

Figure 15: Tracking output of every 10^{th} frame of the bow sequence (local tracking)

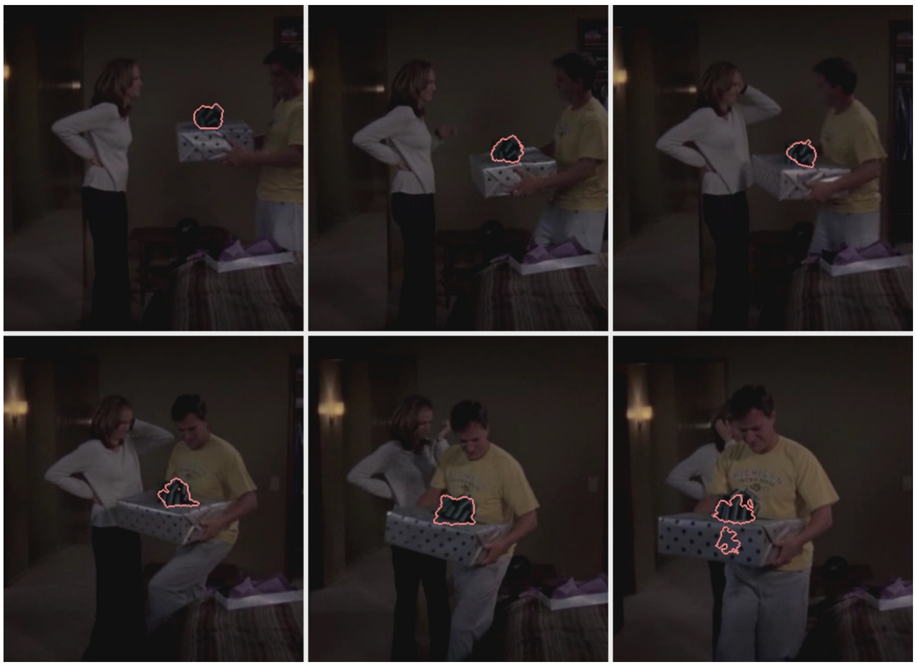

Figure 16: Tracking output of every 10th frame of the bow sequence (global tracking)

6.7 Principle Curvatures

We have engineered a test sequence, shown in Figure 17, on which both the local and global tracking algorithms will fail. Due to the high resolution of the sequence it is not accurately reproduced here, so we will briefly describe it. The region of interest is a ball with alternating black and white horizontal lines whereas the background has alternating vertical lines. We can see that within any small window around a pixel, there are both white and black pixels, hence local tracking will fail. Furthermore, the intensity distributions of the foreground and background are identical and hence global tracking will fail. We can see that throughout the sequence, the directions of principal curvatures are invariant, and hence we use them in our characteristic vector. To be specific, let $\theta \in [0, \pi]$ be one principle direction. To ensure that the principal directions defined by $\theta = 0$ and $\theta = \pi$ are considered the "same" direction, we use the term $2\cos\theta$ in our characteristic vector. Using only this term in our characteristic vector, we tracked the sequence using the global tracking algorithm and the results are shown in Figure 17

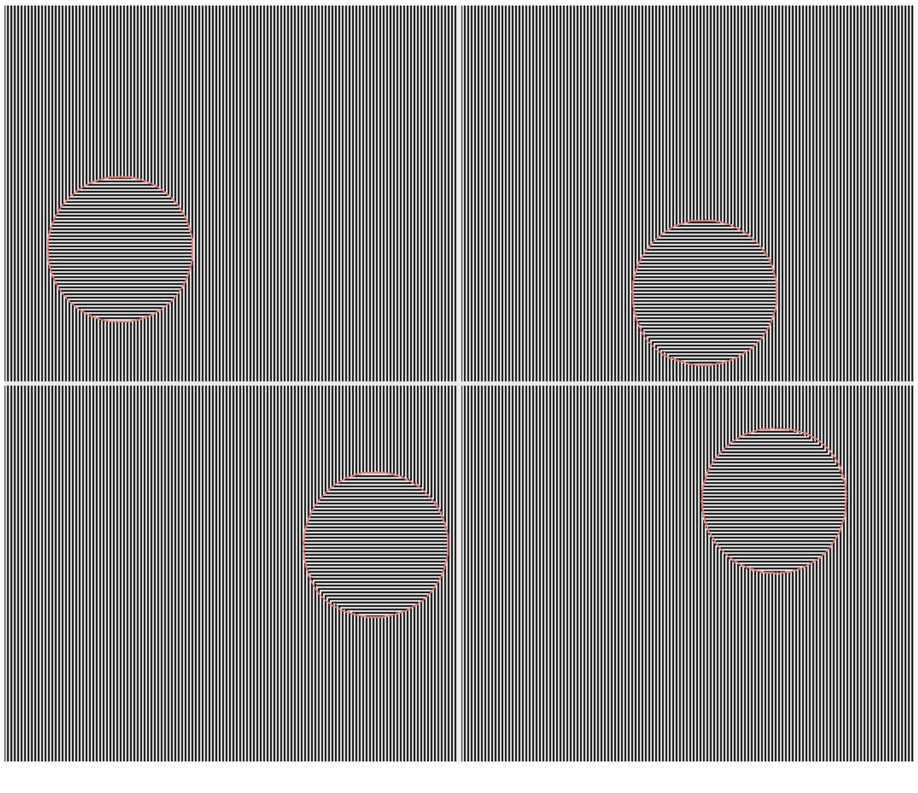

Figure 17: Tracking output of every 15th frame of the synthetic sequence (global tracking)

7 Conclusions

Although we have not been able to fully test every one of our ideas, we can draw some conclusions about the strengths and weaknesses of both the global and local tracking algorithms. As seen in the results section, the local tracking algorithm often provides a tighter fit around the region, but requires the background to be significantly different than the region. This does not mean that there needs to be a strong contrast between the region and the background, but local tracking performs better when colours in the region are not present in the background near the region.

In situations where there are similar colours in the region and the background, global tracking performs better than local tracking. However, as the Kullback-Leibler divergence does not capture the local structure in the image, the contour does not follow the edge of the object accurately. The respective strengths and weaknesses of global and local tracking naturally lead us to try combining the two algorithms. Unfortunately, it was difficult to properly set the parameters with both local and global terms enabled, and we have no successful results. In Section 8, we discuss our alternative ideas for combining the two terms.

We have some very promising theoretical results using principal curvature directions for tracking. As seen in section 6, we are able to track a synthetic sequence using principal directions where tracking by pixel intensities would fail. With further testing, we believe it may turn out that principal values are good higher-order invariant measures of the objects we wish to track.

8 Future Work

Due to the inherent time limitations of a 4th year thesis project, we were unable to pursue many ideas that may have lead to improved tracking. This section is dedicated to future work that could be done if this project were to be continued.

Although we did a significant amount of testing, more testing would provide a better understanding the functional parameters. We were unable to successfully combine local and global tracking, and believe this is primarily due to the fact that we could not effectively combine the global and local functional parameters. Most of the parameters were set based on trial and error. A better understanding of the roles these parameters play througout the gradient descent flow would enable us to make more intelligent parameter selections.

Another method for combining local and global tracking that was discussed involves iteratively switching between the global and local tracking terms during the curve evolution iterations within the processing of a single frame. This could be done in many ways, for example, you could switch between local and global upon each iteration, or run global for the first half of the processing of the frame and local for the second half (or vice versa), or implement a "coin-flipping" process where the algorithm randomly chooses local or global for each iteration. It is believed that by combining local and global tracking in this way it may be possible to find local or global minimums that would otherwise be unreachable.

Principal curvature provided some promising theoretical results, however we were only able to test it on engineered sequences. Hence, more research and testing into this area must be done in order to gain a better understanding of its applicability and effectiveness in real image sequences.

Although code was implemented for FIR filter testing, which would enable the expansion of the characteristic vector with higher order terms for image characteristic, we did not make use of this code during testing. With more time we certainly would have explored the impact of using FIR filters on tracking.

Finally, our group implemented but did not test extensively an idea that that was posed by Professor Mansouri regarding local tracking using local pixel densities rather than single pixel intensity values. Essentially, a PDF is determined for a given window around a pixel, which is used as a local tracking invariant rather than simply the pixel intensities. This can be thought of as another way to combine the local and global tracking algorithms.

It is clear that there are still many areas to explore within the scope of this region tracking project. With a better understanding of each of these areas, and an appropriate amount of testing, the performance of the tracking algorithm may be increased.

References

[1] A-R Mansouri. Region tracking via level set pdes without motion computation. *IEEE Transations on Pattern Analysis and Machine Intelligence*, July 2002.

[2] A-R Mansouri and A. Mitiche. Region tracking via local statistics and level set pdes. *International Conference on Image Processing*, 2002.

[3] A-R Mansouri Problem Description, 2007.

[4] S. V. Fomin I. M. Gelfand. *Calculus of Variations.* Dover Publications, INC., 2000.

[5] J.A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences.* Cambridge University Press, 1996.

[6] D. Freedman and T. Zhang. Active contours for tracking distributions. *IEEE Transations on Image Processing*, April 2004.

[7] Jr. Anthony Yezzi. Modified curvature motion for image smoothing and enhancement. *IEEE Transations on Image Processing*, 1998.

[8] Dirk J. Struik. *Lectures on Classical Differential Geometry.* Dover Publications, INC., 1988.

[9] R. Kimmel. *Numerical Geometry of Images: Theory, Algorithms, and Applications.* Springer, 2003.

[10] D. Freedman and T. Zhang. Improving performance of distribution tracking through background mismatch. *IEEE Transations on Pattern Analysis and Machine Intelligence*, February 2005.

[11] A-R Mansouri Tutorial Notes, 2007.