Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

ThP06.5

Active Contours and Optical Flow for Automatic Tracking of Flying Vehicles

Jincheol Ha*, Christopher Alvino†, Gallagher Pryor‡, Marc Niethammer†, Eric Johnson* and Allen Tannenbaum†

*School of Aerospace Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332-0150

‡College of Computing
Georgia Institute of Technology, Atlanta, Georgia 30332-0280

†School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332-0250

***Abstract*— In this paper, we describe fast implementations of optical flow and geometric active contours to reliably track flying vehicles. Given the position of the vehicle at time $t-1$, optical flow information is used to initially place an active contour in the basin of attraction of a region of interest in a given dynamical image at time t. For real-time tracking, fast convergence of the active contour as well as rapid computation of the optical flow are crucial. In this note, we will describe algorithms that make fast tracking possible in this framework using only standard computing platforms.**

I. Introduction

In this paper, we consider the use of geometric active contours in conjunction with a fast implementation of optical flow for the problem of tracking the position of flying vehicles in real-time. Tracking is a basic control problem in which we want the output to follow or track a reference signal, or equivalently we want to make the *tracking error* as small as possible relative to some well-defined criterion (say energy, power, peak value, etc.). In our case this amounts to minimizing the difference between the position of an object calculated by means of the active contour/optical flow approach and its actual position. Even though tracking in the presence of a disturbance is a classical control issue, the problem at hand is very difficult and challenging because of the highly uncertain nature of the disturbance.

The problem of visual tracking differs from standard tracking problems in the sense that the feedback signal is measured using imaging sensors. In particular, it has to be extracted via computer vision algorithms and interpreted by a reasoning scheme before being used in the control loop. Furthermore, the response speed is a critical aspect. Consequently, from the control point of view, we have a tracking problem in the presence of a highly uncertain disturbance which we want to attenuate. Note that the uncertainty is due to the sensor noise (classical), the algorithmic component described above (uncertainty in extracted features, likelihood of various hypotheses, etc.), and modeling uncertainty.

All correspondence concerning this paper should be sent to Allen Tannenbaum. Email: tannenba@ece.gatech.edu; fax: 404-894-7583.

The capture range of active contours is limited. To assure convergence to the desired image features (the flying vehicles) in each frame, a reasonable initial estimate of the contour position is thus required. We focus on using optical flow information (i.e. the estimated velocity on the contour) to obtain this initial estimate for each frame, and to guarantee reliable tracking. The predicted position is then refined by evolving the predicted contour on each individual image.

The computation of optical flow has proved to be an important tool for problems arising in active vision. The optical flow field is the velocity vector field of apparent motion of brightness patterns in a sequence of images, assumed to be the result of relative motion, large enough to register a change in the spatial distribution of intensities on the images. Note that relative motion between an object and a camera as well as among objects in a scene being imaged by a static camera can give rise to optical flow.

In previous implementations of optical flow ideas in this context, one could only compute the flow in a small region of interest if one wanted real-time tracking. Clearly, in uncertain adversarial environments, it would be desirable to have a more global optical flow computation in initializing the placement of the active contour in each frame. We now have sufficiently fast implementations of active contours (based on level sets) as well as optical flow (based on multigrid ideas) to precisely accomplish these goals. Full details about the L^1 optical flow will appear in Alvino *et al.* [1] which also has a complete set of references for related multigrid work for the computation of optical flow. For other approaches for using optical flow in conjunction with deformable contours see [2] and the references therein.

Finally we also refer the reader to recent work of Niethammer *et al.* [3], in which truly dynamic tracking is performed in a level set framework, leading to a codimension 3 flow.

The paper is organized follows. Section II gives a background on active contours. Section III describes the optical flow computation, including classical gradient descent and multigrid. Section IV shows simulation results. The paper ends with some conclusions and suggestions for future work.

II. Background on Active Contours

We briefly review in this section some of our work on *snakes* or *active contours* based on ideas from curvature driven flows and the calculus of variations. Snakes are autonomous processes which employ image coherence in order to track various features of interest over time. They fit very naturally into a control framework and indeed have been employed in conjunction with Kalman filtering; see [4] and the references therein. In particular, deformable contours have the ability to conform to various object shapes and motions. Snakes have been used for segmentation, edge detection, shape modeling, and visual tracking.

In the classical theory of deformable contours, energy minimization methods are used where controlled continuity splines are allowed to move under the influence of external image dependent forces, internal forces, and certain constraints set by the user. As is well-known there may be a number of problems associated with this approach such as initializations, existence of multiple minima, and the selection of the elasticity parameters. Moreover, natural criteria for the splitting and merging of contours (or for the treatment of multiple contours for the tracking of multiple objects) are not readily available in this framework.

In [5], we propose an active contour model to help treat such problems. The underlying mathematics is based on the Euclidean curve shortening evolution, which defines the gradient direction in which a given curve is shrinking as fast as possible relative to Euclidean arc-length, and on the theory of conformal metrics. We multiply the Euclidean arc-length by a conformal factor defined by the features of interest which we want to extract, and then we compute the corresponding gradient evolution equations. The features which we want to capture therefore lie at the bottom of a potential well to which the initial contour will flow. Moreover, our model may be easily extended to extract 3D contours based on motion by mean curvature [5], [6].

The starting point of this work is [7], [8] in which an active contour model founded on the level set formulation of the Euclidean curve shortening equation is proposed. More precisely, the model is

$$\frac{\partial \Psi}{\partial t} = \phi(x,y)\|\nabla\Psi\|(\text{div}\left(\frac{\nabla\Psi}{\|\nabla\Psi\|}\right) + \nu). \tag{1}$$

Here the function $\phi(x,y)$ depends on the given image and is used as a "stopping term." For example, typically the term $\phi(x,y)$ is chosen to be small near an intensity-based edge, and so acts to stop the evolution when the contour gets close to an edge. One may take [7], [8]

$$\phi := \frac{1}{1+\|\nabla G_\sigma * I\|^2}, \tag{2}$$

where I is the (grey-scale) image and G_σ is a Gaussian (smoothing filter) filter. The function $\Psi(x,y,t)$ evolves in (1) according to the associated level set flow for planar curve evolution in the normal direction with speed a function of curvature which was introduced in [9], [10].

Our approach to active contours begins by noting that the curve shortening part of this evolution, namely

$$\frac{\partial \Psi}{\partial t} = \|\nabla\Psi\|\text{div}\left(\frac{\nabla\Psi}{\|\nabla\Psi\|}\right) \tag{3}$$

is derived as a gradient flow for shrinking the perimeter as quickly as possible. As is explained in [7], the constant *inflation term* ν is added in (1) in order to keep the evolution moving in the proper direction. This part corresponds to an area-minimizing flow [11].

Thus we will modify the model (1) in a precise manner dictated by length/area minimizing ideas. We change the ordinary arc-length function along a curve $C = (x(p), y(p))^T$ with parameter p given by

$$ds = (x_p^2 + y_p^2)^{1/2} dp,$$

to

$$ds_\phi = (x_p^2 + y_p^2)^{1/2} \phi dp,$$

where $\phi(x,y)$ is a positive differentiable function. Then we want to compute the corresponding gradient flow for shortening length relative to the new metric ds_ϕ.

Hence, by introducing an artificial time parameter t for the curve evolution, $C = (x(p,t), y(p,t))^T$, we define

$$L_\phi(t) := \int_0^1 \|\frac{\partial C}{\partial p}\|\phi dp.$$

Taking the first variation of the modified length function L_ϕ, and using integration by parts (see [5]), we get that

$$L'_\phi(t) \quad = \quad -\int_0^{L_\phi(t)} \langle \frac{\partial C}{\partial t}, \phi\kappa\vec{\mathcal{N}} - (\nabla\phi\cdot\vec{\mathcal{N}})\vec{\mathcal{N}}\rangle,$$

where $\kappa = \|C_{ss}\|$ is the curvature, and $\vec{\mathcal{N}} = \frac{1}{\kappa}C_{ss}$ denotes the unit normal to the curve C. The direction in which the L_ϕ perimeter is shrinking as fast as possible is then given by

$$\frac{\partial C}{\partial t} = (\phi\kappa - (\nabla\phi\cdot\vec{\mathcal{N}}))\vec{\mathcal{N}}. \tag{4}$$

This is precisely the gradient flow corresponding to the minimization of the length functional L_ϕ. The level set version of this is

$$\frac{\partial \Psi}{\partial t} = \phi\|\nabla\Psi\|\text{div}\left(\frac{\nabla\Psi}{\|\nabla\Psi\|}\right) + \nabla\phi\cdot\nabla\Psi. \tag{5}$$

One expects that this evolution should attract the contour very quickly to the feature which lies at the bottom of the potential well described by the gradient flow (5). As in [7],

[8], we may also add a constant inflation term, and so derive a modified model of (1) given by

$$\frac{\partial \Psi}{\partial t} = \phi \|\nabla \Psi\| (\text{div}\left(\frac{\nabla \Psi}{\|\nabla \Psi\|}\right) + \nu) + \nabla \phi \cdot \nabla \Psi. \quad (6)$$

(This is referred to a *length/area minimizing flow* in [11].) Notice that for ϕ as in (2), $\nabla \phi$ will look like a doublet near an edge. Of course, one may choose other candidates for ϕ in order to pick out other features.

We now have very fast implementations of these snake algorithms based on a conjugate gradient approach applied to an appropriate energy functional [12]. One may also use level set ideas [9], [10]. Clearly, the ability of the snakes to change topology, and quickly capture the desired features makes them a powerful tool for visual tracking algorithms. See [13] for more details about this.

For multiple objects in an uncertain dynamic environment, the problem is of course how to initially place each contour in a given frame at time t to make sure that it will flow to the desired object(s). Clearly, one may need some other information to do this. Optical flow which provides an approximation of the velocity vector field is therefore a natural tool to include in such a scenario. The next part of the note is thus devoted to a very fast implementation of this methodology which can be used in real-time tracking.

III. Optical Flow Computation

We summarize here some of the key points about our new work on the computation of optical flow using multigrid ideas. In Alvino *et al.* [1], we describe this in the more powerful context of L^1 based optical flow [14] which can track edges much better than the more classical L^2 methods. Moreover, we also give a fully Euclidean invariant version of L^1 optical flow in this work. In this note, however, we suffice to describe multigrid in the classical setting. Multigrid methods [15], [16] have been successfully applied to various elliptic PDE problems including those related to optical flow. See [1] for a detailed set of references. Here we are only interested in using this tool in conjunction with active contours for fast tracking.

A. Optical Flow

Visual motion in an image sequence provides crucial information for object tracking. The computational aspects of visual motion are well understood [17]. Optical flow of a time-varying image sequence is the vector-valued function that describes the spatial direction in which image intensities are moving as time progresses. Beauchemin and Barron survey a number of ways to compute optical flow of a changing image [18]. Horn and Schunck described a cost functional that ensures the resulting vector field simultaneously captures image intensity motion and is smooth [19].

We recall that this functional is constructed by combining an optical flow constraint term, e_c, with a smoothness term, e_s. These terms are defined as

$$e_c = \int \int_{\text{Image}} (I_x u + I_y v + I_t)^2 \, dx\, dy\,, \quad (7)$$

and

$$e_s = \int \int_{\text{Image}} \left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) \, dx\, dy\,, \quad (8)$$

where $I(x, y, t)$ is the image intensity function, u and v are the velocity components in the x and the y directions respectively, and subscripts denote partial derivative with respect to the subscripted variable. The solution to Horn and Schunck's optical flow is a vector-valued function whose components, $u(x, y)$ and $v(x, y)$, minimize the functional,

$$J(u, v) = \lambda e_c + e_s\,. \quad (9)$$

The parameter λ controls tradeoff between the optical flow constraint and the smoothness of the vector field; lower values of λ result in a smoother vector field.

B. Partial Differential Equations and Gradient Descent

The calculus of variations allows us to obtain partial differential equations whose solution minimizes the functional in equation (9). Indeed, these Euler Lagrange equations may be computed to be

$$\begin{aligned} 0 &= \lambda(I_x u + I_y v + I_t) I_x - \Delta u & (10)\\ 0 &= \lambda(I_x u + I_y v + I_t) I_y - \Delta v\,, & (11)\end{aligned}$$

where Δ is the Laplacian operator.

In addition to being the partial differential equations whose solution minimizes equation (9), a result from calculus of variations ensures that the right hand sides of equations (10) and (11) are also infinite dimensional gradients which tell us the direction to perturb u and v respectively to most quickly maximize the functional. Therefore, by perturbing u and v in the direction opposite to this, we guarantee local decrease in the cost functional. Since equations (10) and (11) are elliptic second-order partial differential equations, we can use gradient descent to minimize equation (9).

Although the previous discussion is continuous in time and spatial variables, this can be applied to discretely-sampled sequences of images, both in spatial and temporal variables. The spatially sampled points form a grid. This is convenient for implementation. To represent all spatial derivatives in the image interior, we use central difference approximations. To represent temporal derivatives, we use one-sided difference approximations between image frames. We use Neumann conditions on the boundaries of the images.

The gradient descent algorithm yields new estimates $\tilde{u}'$ and $\tilde{v}'$ from the old estimates, $\tilde{u}$ and $\tilde{v}$, by the equations,

$$\begin{aligned} \tilde{u}' &= \tilde{u} - \gamma\left(\lambda(I_x u + I_y v + I_t) I_x - \nabla^2 u\right) & (12)\\ \tilde{v}' &= \tilde{v} - \gamma\left(\lambda(I_x u + I_y v + I_t) I_y - \nabla^2 v\right)\,, & (13)\end{aligned}$$

where γ is the gradient descent perturbation parameter. For more details on computing optimal values for γ based on the image sequence, we refer the reader to [1].

C. Multigrid Computation

Gradient descent methods are inefficient at determining the solution to a partial differential equation when the discrete grid samples are fine as compared to the coarse scale structure of the solution. Multigrid methods are well known in the numerical analysis literature as being efficient at solving partial differential equations whose solutions have such smooth structure [16], [15]. Multigrid methods exploit the coarse scale structure of the solution by computing the solution on a coarser grid, where the number of computations is smaller.

We implemented multigrid by restructuring equations (10) and (11) in the form of a linear operator on u and v, defined by,

$$\mathcal{L}_1(u,v) = \lambda(I_x u + I_y v)I_x - \Delta u \tag{14}$$

$$\mathcal{L}_2(u,v) = \lambda(I_x u + I_y v)I_y - \Delta v\,. \tag{15}$$

Now, it is easily seen that Eqs. (10) and (11) can be rewritten as the vector equation,

$$\left[\begin{array}{c}\mathcal{L}_1(u,v)\\ \mathcal{L}_2(u,v)\end{array}\right] = \left[\begin{array}{c}-\lambda I_t I_x\\ -\lambda I_t I_y\end{array}\right]. \tag{16}$$

The multigrid algorithm is then described as follows. Starting out with a guess solution $(\hat{u},\hat{v})$, we use the gradient descent equations as described in Eqs. (12) and (13) for ν_1 iterations to refine this guess to be closer to the actual solution.

We then compute the residual on the refined guess,

$$r = \left[\begin{array}{c}\mathcal{L}_1(\hat{u},\hat{v})\\ \mathcal{L}_2(\hat{u},\hat{v})\end{array}\right] - \left[\begin{array}{c}-\lambda I_t I_x\\ -\lambda I_t I_y\end{array}\right]. \tag{17}$$

Note that r is a pair of images, which are stored as values on an original grid. We then smooth and downsample the residual, r, to obtain r_c, the residual on a coarser grid. Next, we solve for the functions on the coarse grid, f_c and g_c, which satisfy the equation,

$$r_\mathrm{c} = \left[\begin{array}{c}\mathcal{L}_{1\mathrm{c}}(f_\mathrm{c},g_\mathrm{c})\\ \mathcal{L}_{2\mathrm{c}}(f_\mathrm{c},g_\mathrm{c})\end{array}\right], \tag{18}$$

where $\mathcal{L}_{1\mathrm{c}}$ and $\mathcal{L}_{2\mathrm{c}}$ are the coarse scale versions of the operators $\mathcal{L}_1$ and $\mathcal{L}_1$. We then obtain corrected guesses by adding the refined guess solution to f and g, the interpolated original grid versions of f_c and g_c. This gives us corrected guesses,

$$\hat{u} \;=\; \hat{u} + f \tag{19}$$

$$\hat{v} \;=\; \hat{v} + g\,. \tag{20}$$

Finally, we refine $(\hat{u},\hat{v})$ a final time with gradient descent for ν_2 iterations to obtain the output of a two grid correction method.

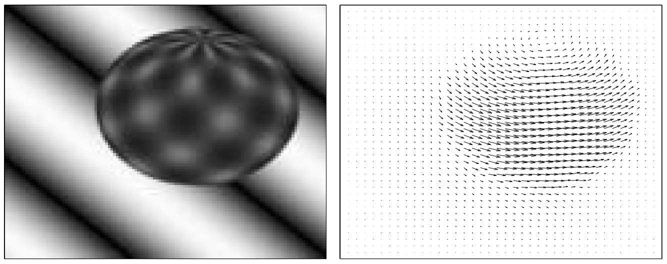
Fig. 1. Image from 200 by 200 pixel rotating sphere sequence (left) and corresponding optical flow computed with multigrid (right).

Since equation (18) is of the same form as equation (16), it should be solved with similar difficulty. However, equation (18) exists on a coarser grid where there are less grid points, allowing us to solve it more efficiently with gradient descent. Since using gradient descent alone on the original grid level takes the most time to obtain the coarse scale part of the solution, we gain efficiency by obtaining the coarse scale part of the solution on the coarser grid, where there are less computations to be done. Optical flow using Horn and Schunk's method has adequate coarse scale structure due to its smoothing term.

We will now explain the logical connection of this two grid correction method to multigrid correction. Note that since equation (18) is of the same form as equation (16), we can choose to solve equation (18) by coarsifying its residual and making a coarse scale correction in the same way we solved for the solution to the original equation. Since, this will result in yet another equation of the same form to be solved, we can implement this algorithm recursively, making as many coarse grid correction steps as necessary, until the resulting linear equation becomes easy to solve exactly. This, in essence, is the multigrid correction method as applied to optical flow.

The values of ν_1 and ν_2 control how accurate the solutions will be. By virtue of ν_2 corresponding to the final gradient descent iterations, it is generally more important than ν_1, as it is responsible for all of the fine scale information obtained after the coarse scale correction. Finally, in our experiments we downsampled by factors of 2 in each spatial dimension. For more details of the described methods, see [1].

Figure 1 shows a single image from the rotating sphere sequence obtained from the Computer Vision Research Group at the Department of Computer Science at the University of Otago, New Zealand. It also shows the corresponding optical flow vector field for a moderate value of the smoothness constant, λ. The image sequence consists of 200 by 200 pixel images.

Figure 2 shows a computational comparison on the rotating sphere sequence. Multigrid increased the efficiency significantly and in fact in our preliminary code converges about an order of magnitude faster than conventional gradient descent. We believe that for larger image sequences,

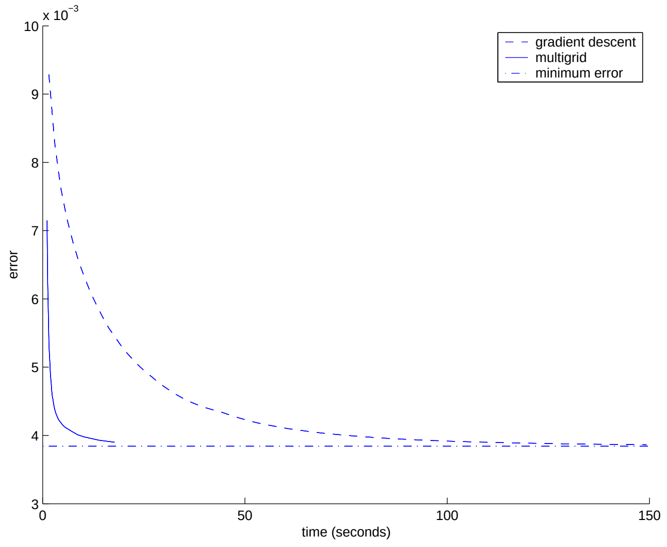

Fig. 2. Error for gradient descent and multigrid calculation of optical flow on two images of the rotating sphere sequence as a function of time. The horizontal line signifies minimum error values.

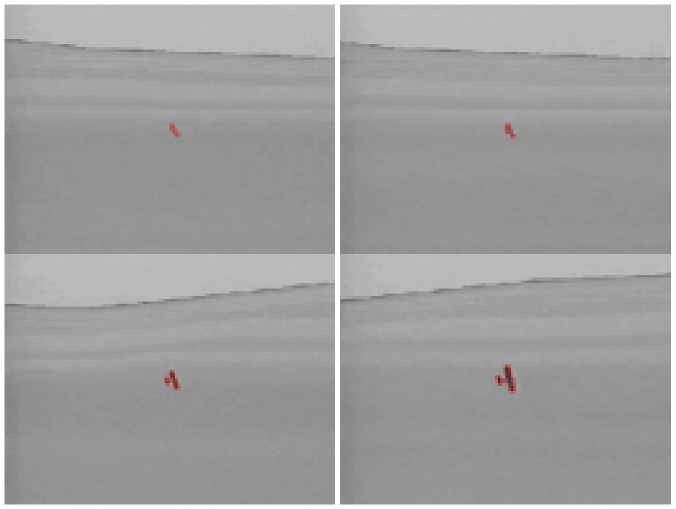

Fig. 3. Four slices of aircraft captured by tracking algorithm from 250 slice temporal sequence.

the savings can be greater. In this experiment, for the multgrid error curve, we enforced the condition $\nu_1 = \nu_2$ for simplicity although this is not necessary and slightly faster computation times might be reached by adjusting ν_1 and ν_2 appropriately.

IV. Simulations

We have tested our algorithms on some simulated aircraft data. We give one such example here. In Figure 3, we show an example of our algorithm capturing an aircraft on four representative slices of a 250 time-point temporal sequence of a "fly-by" data. We should note that the aircraft was automatically captured in all 250 frames. In Figure 4, we render the entire trajectory (time being represented as a third "spatial" dimension in the rendering). The flat plane-like structure is the rendering of the horizon (boundary of earth of sky.)

V. Conclusions and Future Research

In this note, we proposed a straightforward combination of active contours and optical flow for *fast* visual tracking. With fast conjugate-gradient based implementations for active contours and multi-grid optical flow, we have reached tracking speeds of about 30 128×128 frames per second on a standard PC.

There are a number of other paradigms that we plan to test related to active contour tracking. Indeed, snakes may be naturally combined with Bayesian estimation in which the active contour serves as a prior model of the possible shapes and motions of the features of interest which we want to track [20]. Filtering then comes in by adding a dynamic system model to the prior and sensor models in this Bayesian framework.

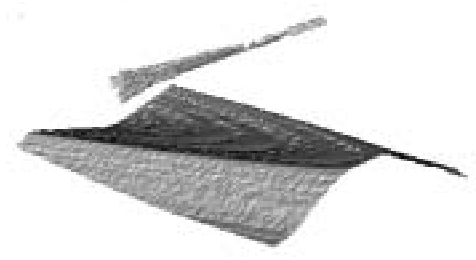

Fig. 4. Trajectory of airplane as captured by active contours and optical flow.

Upon assumption of Gaussian distributions, one may apply the Kalman filter. For multi-modal distributions, particle filters emerge (these are called *CONDENSATION filters* in the computer vision literature; see [4] and the references therein). Particle filtering is a Monte Carlo methodology that is used for nonlinear and non-Gaussian sequential signal processing. These filters are based on the notion of *factored sampling* which generates a random variable from a distribution which approximates the posterior. This is used to search for objects in the image. However, this approach is problematic because it is still too computationally expensive for real-time tracking, which our flying-vehicle applications demand.

Finally, global statistical information may also be put into active contours as in [21], [22]. We will consider using this approach for vehicle tracking in some future work.

Acknowledgements

This research was partially supported by grants from NSF, AFOSR, MURI, ARO, MRI-HEL, and an STTR through Georgia Tech.

References

[1] C. V. Alvino, C. Curry, A. Tannenbaum, and A. Yezzi, "Multigrid methods for L^1-based optical flow computation," Georgia Institute of Technology, School of Electrical and Computer Engineering, Tech. Rep., 2003, to be submitted to *IEEE Transactions on Image Processing*.

[2] N. Peterfreund, "The velocity snake: deformable contour for tracking in spatio-velocity space," *Computer Vision and Image Understanding*, vol. 73, no. 3, pp. 346–356, 1998.

[3] M. Niethammer and A. Tannenbaum, "Dynamic level sets for visual tracking," Georgia Institute of Technology, School of Electrical and Computer Engineering, Tech. Rep., 2003, to be submitted to *IEEE Transactions on Automatic Control*.

[4] A. Blake and M. Isard, *Active Contours*. Springer-Verlag, 1998.

[5] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Conformal curvature flows: from phase transitions to active vision," *Archive for Rational Mechanics and Analysis*, vol. 134, pp. 275–301, 1996, a short version of this paper has appeared in the *Proceedings of ICCV*, June 1995.

[6] A. Tannenbaum, "Three snippets of curve evolution theory in computer vision," *Mathematical and Computer Modelling Journal*, vol. 24, pp. 103–119, 1996.

[7] V. Caselles, F. Catte, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numerische Mathematik*, vol. 66, pp. 1–31, 1993.

[8] R. Malladi, J. Sethian, and B. Vermuri, "Shape modelling with front propagation: a level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 158–175, 1995.

[9] S. J. Osher and J. A. Sethian, "Front propagation with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[10] J. A. Sethian, "Curvature and the evolution of fronts," *Communications in Mathematical Physics*, vol. 101, pp. 487–499, 1985.

[11] Y. Lauziere, K. Siddiqi, A. Tannenbaum, and S. Zucker, "Area and length minimizing flows for segmentation," *IEEE Transactions on Image Processing*, vol. 7, pp. 433–444, 1998.

[12] A. Yezzi and A. Tannenbaum, "4D active surfaces for cardiac analysis," in *Proceedings of the Conference on Medical Image Computing and Computer-Assisted Intervention*, 2002, pp. 667–673.

[13] A. Tannenbaum and A. Yezzi, *The Confluence of Vision and Control*, ser. Lecture Notes in Control and Information Sciences. Springer-Verlag, 1998, vol. 237, ch. Visual tracking, active vision, and gradient flows.

[14] A. Kumar, A. Tannenbaum, and G. Balas, "Optical flow: a curve evolution approach," *IEEE Transactions on Image Processing*, vol. 5, pp. 598–611, 1996.

[15] A. Brandt, "Multi-level adaptive solutions to boundary value problems," *Mathematics of Computation*, vol. 31, pp. 333–390, 1977.

[16] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*. SIAM Publications, 2000.

[17] E. C. Hildreth, "Computations underlying the measurement of visual motion," *Artificial Intelligence*, vol. 23, pp. 309–354, 1984.

[18] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys*, vol. 27, pp. 433–467, 1995.

[19] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[20] D. Terzopoulos and R. Szeliski, *Active Vision*. MIT Press, 1992, ch. Tracking with Kalman Snakes, pp. 3–20.

[21] N. Paragios and R. Deriche, "Geodesic active regions: a new framework to deal with frame partiton problems in computer vision," *Journal of Visual Communication and Image Representation*, vol. 13, pp. 249–268, 2002.

[22] S. Haker, G. Sapiro, and A. Tannenbaum, "Knowledge-based segmentation of SAR images," *IEEE Transactions on Image Processing*, vol. 9, pp. 298–302, 2000.