

Active Blobs

Stan Sclaroff and John Isidoro
Computer Science Department
Boston University
Boston, MA 02215

Abstract

A new region-based approach to nonrigid motion tracking is described. Shape is defined in terms of a deformable triangular mesh that captures object shape plus a color texture map that captures object appearance. Photometric variations are also modeled. Nonrigid shape registration and motion tracking are achieved by posing the problem as an energy-based, robust minimization procedure. The approach provides robustness to occlusions, wrinkles, shadows, and specular highlights. The formulation is tailored to take advantage of texture mapping hardware available in many workstations, PC's, and game consoles. This enables nonrigid tracking at speeds approaching video rate.

1 Introduction

A key open problem in tracking is that of encoding and comparing shapes as they undergo nonrigid deformation. Simply providing robustness to nonrigid deformation is insufficient, because deformation often provides important information about how shapes are related. To make things worse, tracking must also cope with possible lighting changes, specular highlights, shadows, and occlusions.

In images, the motion of objects is sometimes due to changes in viewing geometry: e.g., projective effects, or change in object pose. In many such cases, a simple affine model or eight parameter projective deformation model is sufficient to encode the resulting image motions. However, in general, these parameterizations are inadequate for representing motions that arise due to a physical deformation. For instance, most biological objects are flexible and articulated: fingers bend, cheeks bulge, fish swim, trees sway in the breeze, etc. Shapes are stretched, bent, tapered, dented, etc., and so it seems logical to employ a model that can encode the ways in which real objects deform.

1.1 Related Work

This rationale led to the physics-inspired modeling paradigm of active contours or *snakes* [18]. A snake has a predefined structure that incorporates prior knowledge about a contour's smoothness and its resistance to deformation.

While snakes enforced constraints on smoothness and the amount of deformation, they could not in their original form be used to constrain the *types* of deformation valid for a particular problem domain or object class. Furthermore, it was difficult to use snakes for recognition because of differences in sampling and parameterization in comparing recovered descriptions.

This led to the development of algorithms that enforce *a priori* constraints on the types of allowable deformations for motion tracking [5, 10], deformable templates [16, 34, 36], trainable snakes [2, 9], and deformable prototypes [27]. Such approaches provide a low-dimensional characterization of shape that enables recognition and comparison of nonrigid motions.

Another promising family of approaches is based on correlation of deforming image patches [3, 15, 21, 32]. These approaches integrate information over an image region, and therefore tend to be more immune to noise and/or low-contrast, especially if a robust estimator formulation is employed [4]. To date, most correlation-based models for nonrigid motion require off-line processing, though multi-scale techniques offer some hope for realtime performance [15, 32]. Real-time approaches for tracking of parameterized patches have been developed [12, 13]; however, these methods do not address general nonrigid motion tracking.

1.2 New Approach: Active Blobs

To address the general nonrigid tracking problem, we will introduce a new deformable model formulation: *active blobs*. Active blobs employ a texture-mapped triangular mesh model for tracking deforming shapes in color images.

The goal is to robustly track nonrigidly deforming shapes at speeds approaching video frame-rate on a standard graphics workstation. To gain better robustness, active blobs incorporate information about not only shape, but also color image appearance. Active blobs also provide some robustness to photometric variations, including specular highlights and shadows. By taking advantage of texture mapping hardware available in many workstations, active blobs have achieved peak rates of over twelve frames/sec. on an SGI Indigo2 Impact R10K.

*This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version will be superseded.

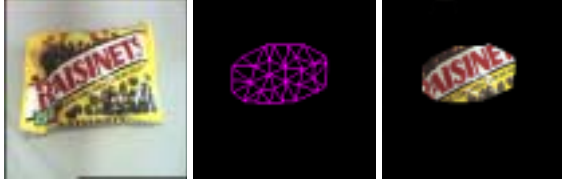


Figure 1: Model construction using a color image. From left to right: a.) input image with region of interest overlaid, b.) resulting triangle mesh, c.) texture mapped model.

2 The Basic Idea

The construction of an example active blob model is shown in Fig. 1. The input to blob construction is an example image plus segmentation information — provided as a binary support region or as a contour that encloses the shape. The input can also include interior feature points to be used as nodes in the triangular mesh. In this example, the user circled the object of interest.

A 2D active blob model is then constructed using a modified Delaunay triangular meshing algorithm. To deform the model, we deform this mesh. Nonrigid deformation of the mesh can be specified in terms of parametric functions; e.g., affine deformations, eight parameter projective deformations, application-specific deformations [3], finite element modal deformations [22, 28], or principal deformations derived from a statistical analysis over a training set of shapes [9, 20, 21].

The blob's appearance is then captured as a color texture map and applied directly to the triangulated model. A blob warp is defined as a deformation of the mesh and then a bilinear resampling of the texture mapped triangles. By defining image warping in this way, it is possible to harness hardware accelerated triangle texture mapping capabilities becoming prevalent in mid-end workstations, PC's, and computer game consoles (e.g., Nintendo 64).

Tracking is then posed as the problem of *active blob registration*. The registration procedure minimizes a function that accounts for both the priors on shape (the deformation parameters) and the priors on appearance (the color texture map). Through the use of a robust error norm, registration can be made robust to specular highlights, shadows, some photometric variations, and small occlusions. Furthermore, the use of color imagery enables tracking in situations where grayscale tracking might be less robust.

An example of nonrigid tracking with an active blob is shown in Fig. 2. The user defined a rectangular region of interest. A finite element modal parameterization was then employed for tracking. As can be seen, the blob model tracks the bag of candy quite well, despite nonrigid deformation, wrinkles, shadows, and specular highlights.



Figure 2: Nonrigid tracking with an active blob. This figure shows every fifteenth frame in a tracking sequence. For visualization purposes, an outline of the active blob is shown overlaid on the input images in the top row. The resulting active blob tracking is shown below each input image.

3 Active Blob Formulation

In the active blobs formulation, nonrigid deformation is controlled by parametric functions. These functions are applied to the node points that define the active blob's 2D triangle mesh. Image warping and interpolation are accomplished by displacing the mesh vertices and then resampling using bilinear interpolation. Thus we define a warping function for an input image, \mathbf{I} :

$$\mathbf{I}' = W(\mathbf{I}, \mathbf{u}) \quad (1)$$

where \mathbf{u} is a vector containing warping parameters, and \mathbf{I}' is the resulting warped image.

The warping parameters control functions that deform the triangle mesh via displacement at the mesh node points:

$$\mathbf{X}' = f(\mathbf{X}, \mathbf{u}), \quad (2)$$

where the vector \mathbf{X} contains the triangle node point locations \mathbf{x}_i , and \mathbf{X}' contains the resulting displaced nodes.

Perhaps the simplest warping functions to be used in Eq. 2 are those of a 2D affine model or an eight parameter projective model [15, 32]. Unfortunately, these functions are only suitable for approximating the rigid motion of a planar patch. The functions can be extended to include linear and quadratic polynomials [3]; however, the extended formulation cannot model general nonrigid motion.

3.1 Finite Element Modes

A more general parameterization of nonrigid motion can be obtained via the use of the modal representation [22]. In the modal representation, deformation is represented in terms of eigenvectors of a finite element (FE) model. The underlying FE formulation offers the added advantage that

it can be used in obtaining a regularized solution to the nonrigid tracking problem, since it can enforce *a priori* constraints on smoothness and the amounts of deformation.

Taken together, modes form an orthogonal basis set for describing nonrigid shape deformation [22, 28]. Blob deformation can be expressed as the linear combination of orthogonal modal displacements:

$$\mathbf{X}' = \mathbf{X} + \sum_{j=1}^m \phi_j \tilde{u}_j, \quad (3)$$

where \tilde{u}_j is the j^{th} mode's parameter value, and the eigenvector ϕ_j defines the displacement function for the j^{th} modal deformation.

The modes are ordered by increasing eigenvalue, ω_j . These eigenvalues correspond with each mode's frequency of vibration. For a 2-D problem, the first three modes are translation and linearized rotation. The next lowest-order modes correspond qualitatively with human's notions of nonrigid deformation: scaling, stretching, skewing, bending, etc. The rest are higher-order nonrigid modes. Such an ordering of shape deformation allows us to select which types of deformations are to be observed. For instance, it may be desirable to make tracking rotation, position, and/or scale independent. To do this, we ignore displacements in the appropriate low-order modes.

The modal decoupling also allows efficient and robust solution to the alignment problem. By discarding high frequency modes the amount of computation required can be reduced without significantly altering tracking accuracy. Discarding the highest-frequency modes can also make tracking more robust to noise [22].

For a given modal parameter vector obtained in tracking, we can compute the strain energy associated with modal deformation:

$$E_{modal} = \sum_{j=1}^m \tilde{u}_j^2 \omega_j^2. \quad (4)$$

Each eigenvalue ω_j defines the stiffness associated with changes in a particular mode parameter. As will be explained in the next section, strain energy can be used to enforce the prior probabilities on the shape's deformations.

3.2 Statistical Modes

As has been pointed out by Terzopoulos [33], and by others [7, 8, 20], there is a well understood link between physically-motivated deformable models and statistical estimation. Splines were perhaps some of the first "physically-based" models employed in statistical estimation [19]; they are particularly well-suited to modeling data sampled from a Markov Random Field (MRF), with Gaussian noise added [11]. The same principles hold true for

regularization [6, 33], where the energies of a physical model can be related directly with measurement and prior probabilities used in Bayesian estimation [30].

Rather than modeling the system as an elastic material, we can instead assume nothing about it, collect data samples of the displacements of each node, and then perform a principal components analysis [9, 20, 21]. The principal directions are defined in terms of the eigenvectors and eigenvalues of the displacement covariance matrix. Each eigenvector defines a *principal deformation*, and can be used directly in Eq. 3. The stiffness associated with each mode is inversely proportional to its corresponding eigenvalue, and can be used directly in Eq. 4.

This connection leads to two useful observations. First, using a FE model is equivalent to making assumptions about the distribution of samples we expect to see. Not using any model, just collecting data and using statistics, on the other hand, implies no *a priori* knowledge of this distribution and instead represents an attempt to statistically approximate it through experimental observation.

3.3 Photometric Variation

We would also like to derive a parameterization for modeling brightness and contrast variations. It is possible to account for photometric variation by extending our previous warping function to include brightness and contrast terms:

$$\mathbf{I}' = c\mathcal{W}(\mathbf{I}, \mathbf{u}) + b, \quad (5)$$

$$b(x, y, \alpha) = \alpha_0 x + \alpha_1 xy + \alpha_2 y + \alpha_3, \quad (6)$$

$$c(x, y, \alpha) = \alpha_4 x + \alpha_5 xy + \alpha_6 y + \alpha_7, \quad (7)$$

where α is a vector of coefficients for bilinear functions that vary with image coordinates (x, y) .

In our current system, the photometric correction terms are defined to scale the red, green, and blue channels equally. Photometric correction is accomplished via image blending capabilities provided by the graphics workstation.

3.4 Combined Parameterization

Deformation and photometric parameters can be combined in generic parameter vector \mathbf{a} . The generic form allows us to utilize active blobs with any combination of the above parameterizations. The image warping function becomes:

$$\mathbf{I}' = \mathcal{W}(\mathbf{I}, \mathbf{a}), \quad (8)$$

and the deformation energy term becomes:

$$E_{deformation} = \sum_{j=1}^m a_j^2 \psi_j^2. \quad (9)$$

where ψ_j^2 are the stiffnesses associated with each parameter. If no stiffness value is available for a particular parameter, then it is set to zero. The stiffnesses can also be determined via statistical estimation [8, 20].

4 Active Blob Registration

The goal of our system is nonrigid shape tracking. To achieve this, the system recovers warping parameters that register a template image \mathbf{I}_0 with a stream of incoming video images. The maximum likelihood solution to this two image registration problem consists of minimizing the squared error for all the pixels within the blob:

$$E_{image} = \frac{1}{n} \sum_{i=1}^n e_i^2 \quad (10)$$

$$e_i = \|\mathbf{I}'(x_i, y_i) - \mathbf{I}(x_i, y_i)\|, \quad (11)$$

where $\mathbf{I}'(x_i, y_i)$ is a pixel in the warped template image as prescribed in Eq. 8, and $\mathbf{I}(x_i, y_i)$ is the pixel at the same location in the input. The above equation is formulated for comparing two color images; thus, it incorporates the sum of squared difference over all channels at each pixel.

Traditional image registration can be easily corrupted by outliers. The process can be made less sensitive to outliers if we replace the quadratic error norm with an *influence function* [14]:

$$E_{image} = \frac{1}{n} \sum_{i=1}^n \rho(e_i, \sigma), \quad (12)$$

where σ is an optional scale parameter, and ρ is the influence function. Such functions are also known as robust error norms [4]; they can be used to control the bias a particular measurement has on the registration solution.

If it is assumed that noise is Gaussian distributed, then the optimal error norm is simply the quadratic norm $\rho(e_i, \sigma) = e_i^2/2\sigma^2$. However, robustness to outliers can be further improved via the use of a Lorentzian influence function [4]:

$$\rho(e_i, \sigma) = \log\left(1 + \frac{e_i^2}{2\sigma^2}\right) \quad (13)$$

This norm replaces the traditional quadratic norm found in least squares. Using the Lorentzian is equivalent to the incorporation of an analog outlier process in our objective function [4]. The formulation results in better robustness to specular highlights and occlusions. For efficiency, the log function can be implemented via table look-up.

Equation 12 includes a data term only; thus it only enforces the recovered model's fidelity to the image measurements. The formulation can be extended to include a regularizing term that enforces the priors on the model parameters \mathbf{a} :

$$E = \frac{1}{n} \sum_{i=1}^n \rho(e_i, \sigma) + \gamma \sum_{j=1}^m a_j^2 \psi_j^2 \quad (14)$$

where γ is a constant that controls the relative importance of the regularization term, and the ψ are the "stiffnesses" associated with each model term as described in Sec. 3.4.

4.1 Marquardt-Levenberg

Registration requires minimization of the residual error with respect to the deformation and lighting parameters. A common approach to multi-dimensional minimization problems is the Marquardt-Levenberg method. This method requires the first and second partial derivatives of E with respect to the unknown model parameters \mathbf{a} . For the Lorentzian error norm, the first partials take the form:

$$\frac{\partial E}{\partial a_k} = \frac{2}{n} \sum_{i=1}^n \frac{e_i}{2\sigma^2 + e_i^2} \frac{\partial \mathbf{I}'}{\partial a_k} + 2\gamma a_k \psi_k^2, \quad (15)$$

The second partials take the form:

$$\begin{aligned} \frac{\partial^2 E}{\partial a_l \partial a_k} = & \frac{2}{n} \sum_{i=1}^n \left[\frac{2\sigma^2 - e_i^2}{(2\sigma^2 + e_i^2)^2} \frac{\partial \mathbf{I}'}{\partial a_k} \frac{\partial \mathbf{I}'}{\partial a_l} + \frac{e_i}{2\sigma^2 + e_i^2} \frac{\partial^2 \mathbf{I}'}{\partial a_l \partial a_k} \right] + \\ & \begin{cases} 2\gamma \psi_k^2 & \text{if } k = l \\ 0 & \text{if } k \neq l \end{cases} \end{aligned} \quad (16)$$

The partial $\frac{\partial \mathbf{I}'}{\partial a_k}$ with respect to a particular model parameter a_k can be approximated by adding small δ to that parameter, warping the model, and then measuring the resulting change in the residual error. All gradient calculations can be made more efficient via the use of the available graphics hardware texture mapping capability.

Following [23], the partials are then used to approximate Hessian matrix \mathbf{H} and a weighted gradient vector \mathbf{g} . It is conventional to drop the image second derivative term and remove the factors of two, thereby defining:

$$g_k \equiv \frac{1}{n} \sum_{i=1}^n \frac{e_i}{2\sigma^2 + e_i^2} \frac{\partial \mathbf{I}'}{\partial a_k} + \gamma \psi_k^2 a_k, \quad (17)$$

$$h_{kl} \equiv \frac{1}{n} \sum_{i=1}^n \frac{2\sigma^2 - e_i^2}{(2\sigma^2 + e_i^2)^2} \frac{\partial \mathbf{I}'}{\partial a_k} \frac{\partial \mathbf{I}'}{\partial a_l} + \begin{cases} \gamma \psi_k^2 & \text{if } k = l \\ 0 & \text{if } k \neq l \end{cases} \quad (18)$$

The deformation and photometric correction parameters are then iteratively updated by solving the linear system:

$$(\mathbf{H} + \lambda \mathbf{I}) \Delta \mathbf{a} = \mathbf{g}, \quad (19)$$

where λ is a stabilization parameter. The stabilization parameter is initially set to a large value. At each iteration, the λ is increased/decreased by a scale factor (typically an order of magnitude) depending on whether the error residual has increased or decreased.

This minimization procedure is iterated until the percentage change in the error residual is below a threshold, or the number of iterations exceeds some maximum. At each iteration, the partial derivatives, approximate Hessian and gradient vector are recomputed.

4.2 Difference Decomposition

Marquardt-Levenberg requires the calculation of $O(N)$ gradient images and $O(N^2)$ image products per iteration of minimization, where N is the number of model parameters. Despite the use of graphics hardware in accelerating this calculation, this is still the performance bottleneck, and therefore an algorithm that decreases the number of gradient calculations is needed. As an alternative, we can use a *difference decomposition* approach [12]. The approach offers the benefit that it requires the equivalent $O(1)$ image gradient calculations and $O(N)$ image products per iteration.

In the difference decomposition, we define a basis of difference images generated by adding small changes to each of the blob parameters. Each difference image takes the form:

$$\mathbf{b}_k = \mathbf{I}_0 - \mathcal{W}(\mathbf{I}_0, \mathbf{n}_k), \quad (20)$$

where \mathbf{I}_0 is the template image, and \mathbf{n}_k is the parameter displacement vector for the k^{th} basis image, \mathbf{b}_k . Each resultant difference image becomes a column in a difference decomposition basis matrix \mathbf{B} . This basis matrix can be determined as a precomputation.

During tracking, an incoming image \mathbf{I} is inverse warped into the blob's coordinate system using the most recent estimate of the warping parameters \mathbf{a} . We then compute the difference between the inverse-warped image and template:

$$\mathbf{D} = \mathbf{I}_0 - \mathcal{W}^{-1}(\mathbf{I}, \mathbf{a}). \quad (21)$$

This difference image \mathbf{D} can then be approximated in terms of a linear combination of the difference decomposition's basis vectors:

$$\mathbf{D} \approx \mathbf{B}^T \mathbf{q}, \quad (22)$$

where \mathbf{q} is a vector of basis coefficients.

Thus, the maximum likelihood estimate of \mathbf{q} can be obtained via least squares:

$$\mathbf{q} = (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \mathbf{D}. \quad (23)$$

The change in the image warping parameters is obtained via matrix multiplication:

$$\Delta \mathbf{a} = \mathbf{N} \mathbf{q}, \quad (24)$$

where \mathbf{N} has columns formed by the parameter displacement vectors \mathbf{n}_k used in generating the difference basis. The basis and inverse matrices can be precomputed; this is the key to the difference decomposition's speed.

The difference decomposition can be extended to incorporate the robust error norm of Eq. 13:

$$\bar{\mathbf{b}}_k(x_i, y_i) = \text{sign}(\mathbf{b}_k(x_i, y_i)) \sqrt{\rho(\mathbf{b}_k(x_i, y_i), \sigma)}, \quad (25)$$

where $\mathbf{b}_k(x_i, y_i)$ is the basis value at the i^{th} pixel. The difference template \mathbf{D} is computed using the same formula.

Furthermore, the formulation can be extended to include a regularizing term that enforces the priors on the model parameters. This is accomplished using a constrained least squares formulation. The energy term to be minimized takes the form:

$$\mathbf{E} = [\mathbf{B} \mathbf{q} - \mathbf{D}]^T [\mathbf{B} \mathbf{q} - \mathbf{D}] + \gamma \mathbf{q}^T \mathbf{P} \mathbf{q}, \quad (26)$$

where $\mathbf{P} = \mathbf{N}^T \Psi^2 \mathbf{N}$.

Differentiating with respect to \mathbf{q} and then rearranging terms, we obtain the constrained least squares solution:

$$\mathbf{q} = [\mathbf{B}^T \mathbf{B} + \gamma \mathbf{P}]^{-1} \mathbf{B}^T \mathbf{D}. \quad (27)$$

If needed, this minimization procedure can be iterated at each frame until the percentage change in the error residual is below a threshold, or the number of iterations exceeds some maximum.

5 Implementation

Active blobs have been implemented using both the affine parameterization and the more general, FE modal parameterization. For tracking, both the Marquardt-Levenberg and difference decomposition minimization algorithms were implemented and tested. Details of this implementation will now be described.

Blob construction starts with the determination of a support region for the object of interest. The bounding contour(s) for a support region can be extracted via a standard 4-connected contour following algorithm [24]. Alternatively, the user can define a bounding contour for a region via a sketch interface. In general, the number of contour segments must be reduced. We utilize the tolerance band approach, where the merging stage can be iteratively alternated with recursive subdivision [17]. In practice, a single merging pass is sufficient if for a user-sketched boundary.

The triangles are then generated using an adaptation of Ruppert's Delaunay refinement algorithm [25, 29], which can produce consistent meshes for two-dimensional polygonal boundaries that can be concave and can include holes. Interior node points may also be specified; this allows for the inclusion of interior features in the active blob model. The algorithm accepts two parameters that control angle and triangle size constraints. To satisfy these constraints, additional interior vertices may be added to the original

polygon during mesh generation. The source code is available from <http://www.netlib.org/voronoi/>.

Once a triangle mesh has been generated, a RGB color texture map is extracted from the example image. Each triangle mesh vertex is given an index into the texture map that corresponds to its pixel coordinate in the undeformed example image I_0 . To improve convergence and noise immunity in tracking, the texture map is blurred using a Gaussian filter. Texture map interpolation and rendering were accomplished using OpenGL.

Given a triangle mesh, the FE model can be initialized using Gaussian interpolants with finite support. Due to space limitations, readers are directed to [26] for the mathematical formulation and pseudocode. The generalized eigenvectors and eigenvalues are computed using code from the EISPACK library: <http://www.netlib.org/eispack/>.

If tracking is to be accomplished via the difference decomposition, then the needed basis images are precomputed. In practice, four basis vectors per model parameter are sufficient. For each parameter a_i , these four basis images correspond with the difference patterns that result by tweaking that parameter by $\pm\delta_i$ and $\pm 2\delta_i$. The factor $2\delta_i$ corresponds to the maximum anticipated change in that parameter per video frame.

6 Examples

The aforementioned system was implemented on a SGI Indigo2 Impact with an R10K processor running at 195 MHz with 192 MB of RAM. This workstation provides texture map acceleration in hardware. All performance statistics are reported for unoptimized code. Experiments were conducted using both minimization methods.

In the first example, Fig. 3, an active blob is used to track a region on a deforming green ball. This tracking was done on-line. In other words, the system tracked using the current frame taken from live video input. If tracking computation per frame could not keep up with full video frame-rate, then in-between frames were skipped.

Frames from the original video sequence are shown in the top row of Fig. 3. The region contained 2394 pixels, and deformation was modeled using the ten lowest-order modal parameters. The recovered active blob tracking for Marquardt-Levenberg is shown below each input image. Using the Marquardt-Levenberg minimization technique, this example ran at an average of two frames a second, with on average 2 to 3 iterations required for convergence. The precomputation time was 0.13 secs.

The same sequence was successfully tracked using the difference decomposition, at an average rate of 12.4 frames per second. Precomputation of the difference basis and matrix inverses took six seconds.

Figure 4 shows another on-line tracking example, this

time tracking a bag of candy. The user circled the region of interest, as shown in the upper left hand corner of Fig. 4. The resulting active blob contained 2680 pixels, and deformation was modeled using the twenty lowest-order modal parameters.

The figure shows every twentieth frame in the tracking sequence. The resulting difference decomposition tracking is shown below each input image. Tracking was accomplished at 8.4 frames per second via the difference decomposition. The required precomputation for basis and inverse matrices took 21 CPU seconds. As can be seen, tracking performs well, despite very large deformations and changes in orientation, specular highlights, and moving shadows.

The same sequence was tracked using Marquardt-Levenberg, at an average rate of 0.9 frames per second. Tracking was less successful, diverging part way through the sequence. This problem can be solved when the Marquardt-Levenberg procedure is run off-line, allowing tracking using all frames from the video sequence.

7 Discussion

The active blobs formulation allows tracking of nonrigidly deforming color regions at over twelve frames per second when the difference decomposition is employed. This is because the active blobs formulation is tailored to take advantage of texture mapping hardware available in many workstations, PC's, and game consoles. In our experience, higher speed tracking is possible if smaller blobs are used.

As was expected, Marquardt-Levenberg is much slower than the difference decomposition. This is due mainly to the number of image operations calculated per iteration, as explained in Sec. 4.2. This makes Marquardt-Levenberg ill-suited for on-line tracking, unless motions are slow.

Tracking speed is dependent on two parameters: number of pixels included in the blob, and number of deformation parameters employed in tracking. Our experience has shown that for either minimization method, the amount of computation needed per minimization iteration scales approximately linearly with the number of pixels. Varying the number of triangles used in the blob does not impact tracking speed significantly.

As seen in the examples, the robust error norm enables reliable tracking, despite nonrigid deformations, photometric changes, and small occlusions.

8 Acknowledgments

This work is sponsored in part by the Office of Naval Research (Young Investigator Award N00014-96-1-0661) and the National Science Foundation (Faculty Early Career Award #IRI-9624168 and CISE infrastructure awards #CDA-9623865 and #CDA-9529403). John Isidoro is sponsored in part through the U.S. Department of Education (GAANN).



Figure 3: On-line tracking of a deformable ball. Marquardt-Levenberg minimization averaged 2 frames/sec, while the difference decomposition averaged 12.4 frames/sec. For visualization purposes, an outline of the active blob is shown overlaid on the original input images. The recovered active blob warps for Marquardt-Levenberg are shown below each input image. Difference decomposition tracked with comparable accuracy.



Figure 4: On-line tracking a deforming plastic bag filled with candy using difference decomposition. This figure shows every twentieth frame in the tracking sequence. For visualization purposes, an outline of the active blob is shown overlaid on the original input images. Marquardt-Levenberg minimization averaged 0.9 frames/sec, while the difference decomposition averaged 8.4 frames/sec. Results for the difference decomposition are shown below each input image. Marquardt-Levenberg was not able to track the object completely through this sequence.

References

- [1] K. Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, 1982.
- [2] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In *Proc. ECCV*, pp. 299–308, 1994.
- [3] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *Proc. ICCV*, 1995.
- [4] M.J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, 19(1):57–91, 1996.
- [5] A. Blake, R. Curwen, and A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *IJCV*, 11(2):127–146, 1993.
- [6] A. Blake and A. Zisserman. *Visual Reconstruction*. M.I.T. Press, 1987.
- [7] T. Boult, S. Fenster, and T. O'Donnell. Physics in a fantasy world vs. robust statistical estimation. in, *Object Representation in Computer Vision*, vol. 994 of *Lecture Notes in Computer Science*, pp. 277–296. Springer-Verlag, 1995.
- [8] T. Cootes. Combining point distribution models with shape models based on finite element analysis. In *Proc. BMVC*, 1994.
- [9] T. Cootes, D. Cooper, C. Taylor, and J. Graham. Trainable method of parametric shape description. *Image and Vision Comp.*, 10(5):289–294, 1992.
- [10] J. Duncan, R. Owen, L. Staib, and P. Anandan. Measurement of non-rigid motion using contour shape descriptors. In *Proc. CVPR*, pp. 318–324, 1991.
- [11] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images. *PAMI*, 6(11), 1984.
- [12] M. Gleicher. Projective registration with difference decomposition. In *Proc. CVPR*, pp. 331–337, 1997.
- [13] G.D. Hager and P.N. Belhumeur. Real time tracking of image regions with changes in geometry and illumination. In *Proc. CVPR*, pp. 403–410, 1996.
- [14] F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stehel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley, 1986.
- [15] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP: Graphical Models and Image Processing*, 53:231–239, 1991.
- [16] A.K. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *PAMI*, 18(3):267–278, 1996.
- [17] R. Jain, R. Kasturi, and B. Shunck. *Machine Vision*. McGraw-Hill, 1995.
- [18] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1:321–331, 1987.
- [19] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation and on stochastic processes and smoothing by splines. *An. of Math. Stat.*, 41(2):495–502, 1970.
- [20] J. Martin, A. Pentland, and R. Kikinis. Shape analysis of brain structures using physical and experimental modes. In *Proc. CVPR*, 1994.
- [21] C. Nastar and A. Pentland. Matching and recognition using deformable intensity surfaces. In *Proc. IEEE Sym. on Comp. Vision*, 1995.
- [22] A. Pentland and S. Sclaroff. Closed-form solutions for physically-based shape modeling and recognition. *PAMI*, 13(7):715–729, 1991.
- [23] W. Press, Brian Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge U. Press, 1988.
- [24] A. Rosenfeld and A. Kak. *Digital Picture Processing*. Academic Press, 1976.
- [25] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. of Algs.*, 18(3):548–585, 1995.
- [26] S. Sclaroff. *Modal Matching: A Method for Describing, Comparing, and Manipulating Digital Signals*. PhD thesis, MIT Media Lab, 1995.
- [27] S. Sclaroff and A. Pentland. Physically-based combinations of views: Representing rigid and nonrigid motion. In *Proc. IEEE Workshop on Nonrigid and Articulate Motion*, 1994.
- [28] S. Sclaroff and A. Pentland. Modal Matching for Correspondence and Recognition. *PAMI*, 17(6):545–561, 1995.
- [29] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Proc. ACM Workshop on App. Comp. Geom.*, pp. 124–133, 1996.
- [30] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision*. Kluwer, 1989.
- [31] R. Szeliski. Video mosaics for virtual environments. *IEEE CG&A*, 16(2):22–30, 1996.
- [32] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *Proc. CVPR*, pp. 194–201, 1994.
- [33] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *PAMI*, 8(4):413–424, 1986.
- [34] D. Terzopoulos. On matching deformable models to images: Direct and iterative solutions. In *Topical Meeting on Machine Vision*, vol. 12 of *Tech. Digest Series*, pp. 160–167, 1987. Optical Soc. of America.
- [35] L. Williams. Pyramidal Parametrics. *Comp. Graphics*, 17(3):1–11, 1983.
- [36] A. Yuille, D. Cohen, and P. Hallinan. Feature extraction from faces using deformable templates. In *Proc. CVPR*, pp. 104–109, 1989.