ACADEMIC PRESS

Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Computer Vision and Image Understanding xxx (2003) xxx–xxx

Computer Vision and Image Understanding

www.elsevier.com/locate/cviu

# Active blobs: region-based, deformable appearance models

Stan Sclaroff* and John Isidoro

*Image and Video Computing Group, Computer Science Department, Boston University, Boston, MA 02215, USA*

Received 17 December 2002; accepted 17 December 2002

**Abstract**

A 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, PCs, and game consoles. This enables nonrigid tracking at speeds approaching video rate.


*Keywords:* Deformable templates; Appearance models; Region tracking; Robust estimation; Image registration

## 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, since deformation often provides important information about how shapes are related. Image registration is one commonly used approach for tracking nonrigid objects in video. In this approach, the main goal is to estimate a set of geometric transformations that describe the changes observed in a video

* Corresponding author.
*E-mail addresses:* sclaroff@cs.bu.edu (S. Sclaroff), jisidoro@cs.bu.edu (J. Isidoro).

1077-3142/03/$ - see front matter © 2003 Published by Elsevier Science (USA).
doi:10.1016/S1077-3142(03)00003-1

stream due to the object's motion. For tracking individual objects, the registration is more effective when it takes place over the subregion of each video frame that contains only the object of interest. Despite the computational advantages of using a smaller region, there are still many nuances that can make realtime, on-line object tracking a challenging problem.

Perhaps the most difficult aspect of modeling nonrigid deformation is that there is no universally defined way to describe it. In images, the motion of objects is sometimes due to changes in viewing geometry: e.g., projective effects, or changes in object pose. In many such cases, a simple affine model or eight parameter projective deformation model is sufficient for registration. 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. Fig. 1 shows two simple examples of the types of nonrigid object motion that we want to be able to track.

Another challenging problem is presented by illumination changes. One classic approach to motion estimation is to employ the constant brightness assumption: all changes in pixel intensity are due to object motion. However, in general this is not the case; illumination also plays a major role. This is best seen when an object moves in the presence of fixed lighting. The lighting effects on the surface of an object almost always change when the object moves. For planar surfaces, the change due to lighting may be adequately modeled using two terms, contrast and brightness. However, if a system is going to track objects with nonplanar surfaces, or take into account light attenuation from distance, a more extensive lighting model is required. Fig. 2 shows some simple examples of lighting changes that we would like our tracking formulation to accommodate.

In addition to modeling deformation and illumination, a registration-based tracking system must also gracefully handle all sorts of other common phenomena, including shadows, specular highlights, and partial occlusions. Shadows are difficult

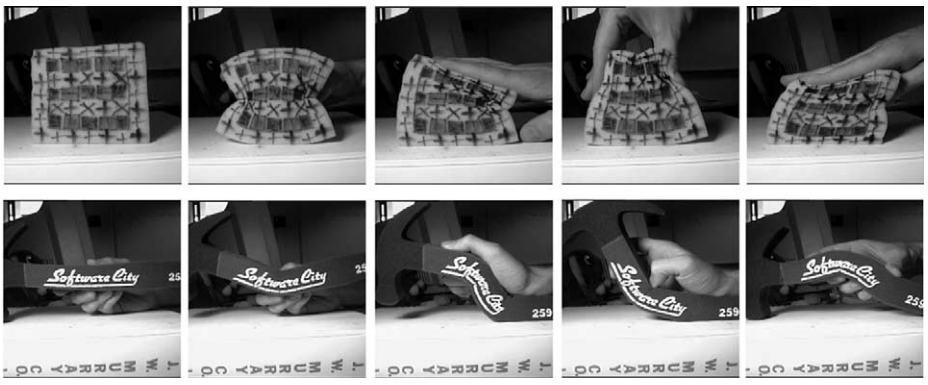


Fig. 1. Examples of the types of nonrigid deformations we would like to be able to track.


Fig. 2. Examples of changing lighting conditions: (a) planar surface; (b) nonplanar surface.


Fig. 3. Examples of outliers. This figure shows pairs of images; the first of the pair shows an ''outlier-free object,'' and the second shows the object affected by the outlier: (a) shadows; (b) occlusions (the second example is partially occluded by the viewport); (c) specular highlights (notice the 'p' in OpenGL, and the second 'i' in Raisinets).

to model when the 3D structure of an object is unknown. Another cumbersome problem occurs when the object being tracked becomes partially obscured by another object, or partially moves out of the camera's field of view. Such occlusions are difficult to deal with, especially when the location of the occluding object is unknown. Specular highlights, a small reflection of the light source which can occur if the object has a shiny surface, are also very difficult to deal with. In this paper, these phenomena will fall under the general category of *outliers*, i.e., portions of the data that are not explicitly included in the model. Fig. 3 shows some examples of the outliers just mentioned. As will be seen later in this paper, robust estimation methods can enable reliable tracking despite the presence of such outliers.

## 2. Overview of our approach

To address the aforementioned challenges, we will introduce a deformable model formulation: *active blobs*. An active blob is a texture-mapped deformable 2D triangular mesh which is registered with the incoming video. The construction of an example active blob model is shown in Fig. 4. 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 the example shown in Fig. 4, the user circled the object of interest. A 2D active blob mesh model is then constructed using a modified Delaunay triangular meshing algorithm. To deform the model, we deform this mesh.

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 re-sampling of the texture-mapped triangles. By defining image warping in this way, it is possible to harness hardware-accelerated triangle texture mapping capabilities prevalent in workstations, PCs, and computer game consoles. By taking advantage of texture mapping hardware commonly available in PCs, the active blobs system has achieved peak rates of over 20 frames/s.[1]

Nonrigidity is modeled by a set of global, parametric shape warping functions that control the blob's deformation during registration. In this paper the warping functions are defined by the 2D affine transformations and the 2D free-vibration modes. The free-vibration modes are found by using the finite element method (FEM) in conjunction with modal analysis [41,47]. At an intuitive level, the modal analysis of an object is analogous to a Fourier transformation for shape. It provides a set of warping functions that take the form of a frequency ordered orthogonal transformation basis. The columns of this basis are known as the modal shape vectors. Modes provide only one of the many possible choices of a deformation basis. For instance, deformations can be derived from a statistical analysis over a training

[1] An implementation is available via anonymous FTP at `http://www.cs.bu.edu/groups/ivc/ActiveBlobs`.

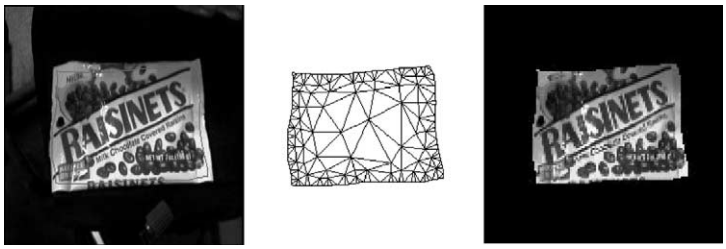


Fig. 4. Model construction using a color image. From left to right: (a) input image with region of interest overlaid; (b) resulting triangle mesh; (c) texture mapped model.

set of shapes [14,38]. One of the the reasons modes are used in active blobs is because they simplify computation of the active blob's internal energy.

In active blobs, an illumination basis is used to model changes in the blob's appearance due to lighting. In the spirit of not assuming any 3D properties of the object being tracked, the illumination basis is a Taylor series approximation to the lighting. A zeroth order model provides only global brightness and contrast terms while a second order model can approximate lighting conditions on spheres, cylinders and planes with a total of 12 terms. Objects with more complex geometries can be modeled with a higher-order lighting model.

Tracking is then posed as the problem of *active blob registration*. The registration procedure minimizes an objective function that includes both an image difference energy term and a geometric deformation energy term.

The image difference energy term decreases as the warped blob image becomes more similar to the input video image. For active blobs to work properly, it is crucial that the object to be tracked contains some sort of pattern or texture. This pattern causes the object's appearance to change when moving or deforming. Each shape parameters' corresponding warped blob should be unique in appearance, so that there is no confusion between parameters. In general, more detailed and varied surface patterns produce better results. The active blobs image energy term formulation also includes a robust error norm in order to handle outliers. This enables active blobs to handle the image artifacts caused by specular highlights, shadows, occlusions, etc.

The geometric deformation energy term, often referred to as a regularization term, is a measure of how much energy it takes for a blob to deform into its current shape. This term makes the blob act as if it were made of a sheet of rubber. As the blob deforms further from its initial configuration, the geometry term increases in energy. This effectively limits the amount of deformation that the blob is allowed to undergo. The regularizer makes the blob resilient to undesirable configurations, thus increasing the system stability, particularly when large numbers of modes are used.

An example of nonrigid tracking with an active blob is shown in Fig. 5. The user defined a rectangular region of interest. A shape parameterization using 12 FEM modes was used for tracking. As can be seen, the blob model tracks the bag of candy quite well, despite nonrigid deformation, wrinkles, shadows, and specular highlights. More rigorous testing can be found in the experimental results section of this paper.


Fig. 5. Nonrigid tracking with an active blob: (a) every 15th frame in the input video; (b) active blob tracking. For visualization purposes, an outline of the active blob is shown overlaid on the input images in the top row.

## 3. Related work

Active contours, also known as *snakes* [28], can be used to find image contours of an object and track moving contours in video. The snake's mathematical formulation includes both image and geometric energy terms. Typically, the image term takes the form of an edge detector that causes the snake to be pulled toward the high contrast areas of an image. The geometric term can take a variety of forms. Most implementations use a continuity term which minimizes the distance between points along a snake. Some have the continuity term also penalize uneven spacing between the adjacent points along a contour [64]. Another possibility for a continuity term is a balloon model [11] which causes the snake to expand to meet the edges. Also, most implementations include a curvature term which prevents unwanted sharp bends from occurring in the snake. In a departure from the snake's energy-based formulation, level set methods for boundary tracking [34] have also been proposed.

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 [1,8,49], deformable templates [25,55,65], trainable snakes [3,12], and deformable prototypes [46]. Such approaches provide a low-dimensional characterization of shape that enables recognition and comparison of nonrigid motions.

One problem with snakes is the fact that they only track the contours of an object, and ignore the region inside the snake. This is what makes snakes very different from active blobs. Snakes often fail in situations where the tracked object and the back-

ground are highly textured. The snake may find edges which are not on the contour of the object, and for this reason the snake loses the track. Conversely, active blobs work best in highly textured situations, but fail in cases where the region of interest does not exhibit substantial texture.

Another promising family of approaches is based on registration of deforming image patches [7,16,23,38,45,52,53]. These approaches integrate information over an image region, and therefore tend to be more immune to noise and/or low-contrast. There has also been a rich vein of related work in medical image registration methods (see [32] for a review); however, for the sake of brevity, our discussion is limited to methods for motion tracking. A nice unifying view of registration approaches was recently presented in [2].

Most techniques for registration rely on minimizing a sum of squared differences (SSD) between the warped template image, and a novel input image. A nonlinear minimization procedure is iterated until the closest possible match is found. This basic registration approach has been used in various applications such as mosaicing [51], tracking [7,65], image enhancement/super-resolution [16,23], and computer–human interfaces [10,24]. To date, most registration-based methods require off-line processing, although multi-scale techniques offer some hope for realtime performance [23,52]. Approximation methods can also be used [4,61].

A very efficient approach to minimization is the difference decomposition [19]. In the difference decomposition, the input image is inverse warped to match the template image. By doing this, the set of difference images can be precalculated once for a template rather than once per iteration of the registration algorithm. In the case of tracking objects in video, this is a huge improvement because the difference decomposition needs only be computed once. A similar precomputation approach is proposed by [20] for forward warping, but it involves more computation than in the difference decomposition. Both approaches enable realtime tracking of deforming image patches for the affine case; however, the methods do not address general nonrigid motion tracking.

Very often, a simple affine or eight-parameter projective model cannot accurately represent the image motion present. Objects in the real world can often bend, twist, and taper in ways that elude such a simple deformation representation. This is especially true in the case of biological motion, where soft tissues can deform in complicated ways. Other instances of nonrigid motion can be due to soft toys, clothes, and even paper objects. Many researchers have proposed solutions to nonrigid tracking. As mentioned earlier, snakes are one possibility, but they only can track contours in their original form. Some other 2D approaches to tracking employed deformable templates [7,15,65] or connected templates [53]. However, the templates used were specific to the object being tracked (facial features) and only involve nonlinear terms which were quadratic.

A number of authors have proposed methods that use linear combinations of warped images to synthesize novel views [13,27,38,46,60]. In [13,38,46], the warping functions used are derived via a Principal Components Analysis (PCA) over a training set of aligned face images. In [60] an iterative algorithm was employed along with PCA to generate a set of principal flow fields which also used nonrigid warping to

model and synthetically generate whole classes of objects. In each of these approaches, nonrigidity is used to find correspondences between different objects, and is not necessarily used for tracking.

In other related work, 2D active mesh representations have been proposed for video compression. A regular quadrilateral, active mesh formulation was given by [62], in which mesh nodes dynamically become more densely distributed in image regions that contain edges/corners. Motion of the mesh is determined by tracking image features corresponding to each node. Color over each quadrilateral is approximated by interpolating colors stored only at mesh nodes; no texture map is stored. A geometric term that models virtual springs between mesh nodes keeps the mesh stable. A quadtree-based reformulation [30,63] enables multi-scale representation, and therefore greater flexibility in mesh adaptation. Later in [58,59], 2D hierarchical, constrained Delaunay triangulation was used with texture mapping. In [33], a FEM modal analysis formulation for the geometry term was proposed, and a robust, least median of squares approach was used in estimating mesh deformation. Finally, in [31] a mesh model was used in facial-image coding. In each of these methods, mesh nodes are placed at feature locations in the first image, and the image energy term is determined by tracking the features only. In other words, the methods do not employ image registration over all pixels within the region of interest, instead they only use pixels around specified feature locations.

Perhaps the most daunting challenge of registration-based approaches is that is difficult to model all the phenomena which may occur in the real world. Many of the above techniques make use of prior knowledge and/or regularization to constrain the solution. Nonetheless, occlusions, specular highlights, shadows, and other lighting nuances can violate the basic Gaussian noise assumption implicit in SSD registration methods. Hopefully only a small percentage of the pixels in the region of interest will be polluted with these image artifacts. Such pixels are *outliers*, and can easily corrupt the solution. To address this issue, one can replace the quadratic error norm (i.e., SSD) with a *robust error norm* that prevents the outliers from having a significant impact on the solution. An in-depth discussion of robust statistics pertaining to vision is given in [6]. To perform robust registration using a least squares approach, most robust error norms are implemented using iteratively re-weighted least squares (IRLS) [5,20,39]. In active blobs we use a Lorentzian robust error norm, and derive an IRLS version of difference decomposition in Section 5.2.

Finally, it should be noted there has been a great deal of work in fitting 3D deformable models to image data (see [36] for a review). One of the problems of fitting a 3D model to a single image is that depth is almost impossible to determine, and the tracking problem is generally underconstrained. As a result, 3D model-based nonrigid motion tracking techniques require either strong prior assumptions about the shape of the object (e.g., symmetry [57], deformable superquadrics [37,40,66], human faces [15,17,56]) or that multiple views of the object are available. By using a 2D deformable image template, the active blobs formulation has the advantage of being able to track general nonrigid motion within the image plane without the need to define a full 3D model nor strong prior assumptions about the shape of the object to be tracked. No training is required to build the set of warping functions since they can

be derived using a modal analysis of the finite element model. Also, because the system is inherently 2D, only one viewpoint is required for tracking. Despite its 2D nature, the nonrigidity of the model allows tracking of simple nonplanar objects as well.

## 4. Active blob formulation

A generic model formulation will now be derived. The formulation will make it possible to utilize any of a number of different nonrigid deformation parameterizations. Model parameters for changes in image brightness and contrast will also be defined. By the end of the section, the deformation and lighting parameters will be unified in a single generic active blobs formulation.

### 4.1. Blob warping

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, $\mathbf{I}$

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

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

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

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 an eight-parameter projective model [23,52]. 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 [7]; however, such polynomials cannot model general nonrigid motion.

### 4.2. Finite element modes

A more general parameterization of nonrigid motion can be obtained via the use of the modal representation [41,47]. 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 constraints on smoothness and the amounts of deformation.

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

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

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

The modes are ordered by increasing eigenvalue, $\omega_j$. These eigenvalues correspond with each mode's frequency of free-vibration. For a 2D 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 enables an 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 [41].

For a given modal parameter vector, $\tilde{\mathbf{u}}$ obtained in tracking, we can compute the strain energy associated with modal deformation

$$E_{\text{modal}} = \sum_{j=1}^{m} \tilde{u}_j^2 \omega_j^2. \tag{4}$$

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

### 4.3. Statistical modes

As has been pointed out by Terzopoulos [54], 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 [29]; they are particularly well-suited to modeling data sampled from a Markov Random Field (MRF), with Gaussian noise added [18]. The same principles hold true for regularization [9,54], where the energies of a physical model can be related directly with measurement and prior probabilities used in Bayesian estimation [50].

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 (PCA) [12,35,38]. The principal directions are defined in terms of the eigenvectors and eigenvalues of the displacement covari-

ance 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 useful insights. 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 prior knowledge of this distribution and instead represents an attempt to statistically approximate it through experimental observation. A hybrid formulation of FE modes and PCA is described in [14].

### 4.4. Photometric variation

We include within our parameterization models for both brightness and contrast variations. Photometric variations over the surface are modeled with a Taylor series approximation. In theory multiple colored light sources could be handled by dealing with the red, green, and blue channels separately. However, in the majority of real world situations lights are nearly monochromatic and close to white in color. Therefore, a simple monochromatic model will suffice. Our formulation is

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

where $\alpha$ is a vector of coefficients, and the contrast $\mathbf{c}(\alpha)$ and brightness $\mathbf{b}(\alpha)$ image functions are:

$$\mathbf{c}(\alpha) = 1 + \sum_{i=0,j=0}^{i+j \leqslant d} \kappa_{i,j} x^i y^j, \tag{6}$$

$$\mathbf{b}(\alpha) = \sum_{i=0,j=0}^{i+j \leqslant d} \beta_{i,j} x^i y^j. \tag{7}$$

In the above equations, $\kappa_{i,j}$ is the subvector of parameters in $\alpha$ that modulate contrast, and $\beta_{i,j}$ is the subvector of parameters that modulate brightness. The degree of the Taylor series approximation is given by $d$. The $\mathbf{x}$ and $\mathbf{y}$ are the set of pixel locations for each pixel within the active blob. Note that one $\alpha$ vector is used to parameterize the photometric correction over the entire blob. Photometric correction can be accomplished via the color blending capabilities provided by the graphics hardware.

### 4.5. 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

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

and the deformation energy term becomes

$$E_{\text{deformation}} = \sum_{j=1}^{m} a_j^2 \psi_j^2, \tag{9}$$

where $\psi_j^2$ 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 [14,35].

One useful combined parameterization we have employed is the combination of 2D affine and free vibrational modes, as well as photometric parameters. We have found that using the six affine deformations plus 4–8 additional free vibrational modes works well in practice. Gram-Schmidt orthogonalization is then used to factor the affine components out of the modes of free vibration. Any mode vectors with magnitude close to zero after subtracting off the projection of the affine components are not used.

## 5. 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:

$$E_{\text{image}} = \frac{1}{n} \sum_{i=1}^{n} e_i^2, \tag{10}$$

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

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 differences 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* [21]

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

where $\sigma$ is an optional scale parameter, and $\rho$ is the influence function. Such functions are also known as robust error norms. 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

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

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 [6]. The formulation results in better robustness to shadows, specular highlights, and partial occlusions. For efficiency, the log function can be implemented via table look-up.

Eq. (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 **a**

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

where $\gamma$ is a regularization parameter, and controls the relative strength to which the priors are enforced. The value $\gamma$ can be thought of as a general "stiffness" of the model parameters. Each $\psi_j$ specifies the stiffness on the $i$th model parameter. When using modes of free-vibration, then $\psi_j = \omega_j$, the stiffnesses associated with each mode. For the photometric variation parameters, the stiffness values were chosen heuristically to be the same order of magnitude as the shape stiffness terms.

Registration requires minimization of the residual error with respect to the deformation and lighting parameters. Two multi-dimensional minimization methods have been implemented and tested. The first method, Levenberg–Marquardt [42,51], requires computation of the Hessian and the gradient at each iteration. The second method, the difference decomposition [19], requires a set of difference images that can be precomputed and reused at each iteration. We will now describe both methods. Results using both registration methods are given in Section 7.

### 5.1. Levenberg–Marquardt

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 Levenberg–Marquardt method. This method requires the first and second partial derivatives of $E$ with respect to the unknown model parameters **a**. For the Lorentzian error norm, the first partials take the form

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

The second partial derivatives take the form:

$$\frac{\partial^2 E}{\partial a_l \partial a_k} = \frac{2}{n}\sum_{i=1}^{n}\frac{\partial}{\partial a_l}\frac{e_i}{2\sigma^2+e_i^2}\frac{\partial \mathbf{I}'}{\partial a_k} + 2\gamma\psi_k^2\frac{\partial a_k}{\partial a_l} \tag{16}$$

$$= \frac{2}{n}\sum_{i=1}^{n}\left[\frac{2\sigma^2-e_i^2}{(2\sigma^2+e_i^2)^2}\frac{\partial \mathbf{I}'}{\partial a_k}\frac{\partial \mathbf{I}'}{\partial a_l} + \frac{e_i}{2\sigma^2+e_i^2}\frac{\partial^2 \mathbf{I}'}{\partial a_l a_k}\right] + \begin{cases} 2\gamma\psi_k^2 & \text{if } k=l, \\ 0 & \text{if } k\neq l. \end{cases} \tag{17}$$

The partial $\partial \mathbf{I}'/\partial a_k$ with respect to a particular model parameter $a_k$ can be approximated by adding a small $\delta$ 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 texture mapping capability of available graphics hardware.

Following [42], the partial derivatives are then used to approximate the 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:

$$g_k \equiv \frac{1}{n}\sum_{i=1}^{n} \frac{e_i}{2\sigma^2 + e_i^2}\frac{\partial \mathbf{I}'}{\partial a_k} + \gamma\psi_k^2 a_k, \tag{18}$$

$$h_{kl} \equiv \frac{1}{n}\sum_{i=1}^{n} \frac{2\sigma^2 - e_i^2}{(2\sigma^2 + e_i^2)^2}\frac{\partial \mathbf{I}'}{\partial a_k}\frac{\partial \mathbf{I}'}{\partial a_l} + \begin{cases} \gamma\psi_k^2 & \text{if } k = l, \\ 0 & \text{if } k \neq l. \end{cases} \tag{19}$$

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

$$(\mathbf{H} + \lambda\mathbf{I})\Delta\mathbf{a} = \mathbf{g}, \tag{20}$$

where $\lambda$ is a stabilization parameter. The stabilization parameter is initially set to a large value. At each iteration, the $\lambda$ 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.

### 5.2. Difference decomposition

Levenberg–Marquardt 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 minimization, it remains 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 [19]. 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 set of difference images generated by adding small changes to each of the blob parameters. Each difference image takes the form

$$\mathbf{b}_k = \mathbf{I}_0 - \mathcal{W}(\mathbf{I}_0, \mathbf{n}_k), \tag{21}$$

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

$$\mathbf{D} = \mathbf{I}_0 - \mathcal{W}^{-1}(\mathbf{I}, \mathbf{a}). \tag{22}$$

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

$$\mathbf{D} \approx \mathbf{B}\mathbf{q}, \tag{23}$$

where $\mathbf{q}$ is the vector of coefficients.

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

$$\mathbf{q} = (\mathbf{B}^{\mathrm{T}}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{D}. \tag{24}$$

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

$$\Delta\mathbf{a} = \mathbf{N}\mathbf{q}, \tag{25}$$

where $\mathbf{N}$ has columns formed by the parameter displacement vectors $\mathbf{n}_k$ used in generating the difference decomposition. The difference decomposition matrix and its inverse matrix can be precomputed; this leads to greatly reduced computational complexity for online tracking.

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

$$\bar{\mathbf{b}}_k(x_i, y_i) = \mathrm{sign}(\mathbf{b}_k(x_i, y_i))\sqrt{\rho(\mathbf{b}_k(x_i, y_i), \sigma)}, \tag{26}$$

where $\mathbf{b}_k(x_i, y_i)$ is the value of the $k$th difference decomposition image 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

$$\mathbf{E} = [\mathbf{B}\mathbf{q} - \mathbf{D}]^{\mathrm{T}}[\mathbf{B}\mathbf{q} - \mathbf{D}] + \gamma[\mathbf{a} + \Delta\mathbf{a}]^{\mathrm{T}}\Psi^2[\mathbf{a} + \Delta\mathbf{a}]. \tag{27}$$

Using Eq. (25) and differentiating with respect to $\mathbf{q}$ we obtain the constrained least squares solution:

$$\mathbf{q} = \left[\mathbf{B}^{\mathrm{T}}\mathbf{B} + \gamma\mathbf{N}^{\mathrm{T}}\Psi^2\mathbf{N}\right]^{-1}\left[\mathbf{B}^{\mathrm{T}}\mathbf{D} - \gamma\mathbf{N}^{\mathrm{T}}\Psi^2\mathbf{a}\right] \tag{28}$$

$$= \left[\mathbf{B}^{\mathrm{T}}\mathbf{B} + \gamma\mathbf{N}^{\mathrm{T}}\Psi^2\mathbf{N}\right]^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{D} - \gamma\left[\mathbf{B}^{\mathrm{T}}\mathbf{B} + \gamma\mathbf{N}^{\mathrm{T}}\Psi^2\mathbf{N}\right]^{-1}\mathbf{N}^{\mathrm{T}}\Psi^2\mathbf{a} \tag{29}$$

$$= \mathbf{P}\mathbf{D} - \mathbf{Q}\mathbf{a}, \tag{30}$$

where $\mathbf{P} = [\mathbf{B}^{\mathrm{T}}\mathbf{B} + \gamma\mathbf{N}^{\mathrm{T}}\Psi^2\mathbf{N}]^{-1}\mathbf{B}^{\mathrm{T}}$ and $\mathbf{Q} = \gamma[\mathbf{B}^{\mathrm{T}}\mathbf{B} + \gamma\mathbf{N}^{\mathrm{T}}\Psi^2\mathbf{N}]^{-1}\mathbf{N}^{\mathrm{T}}\Psi^2$ can be calculated once as a precomputation (since these matrices remain constant).

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.

### 5.3. Gaussian pyramids

A common problem in registration-based tracking systems is that of getting trapped in local minima. Typically, this problem occurs when the object is moved very quickly causing a large inter-frame difference. To overcome this problem, we use Gaussian pyramids during the tracking process. The type of Gaussian pyramid used is an octave pyramid, where each image in the pyramid is half the size of the previous one. Registration is performed starting with the lowest resolution image of the pyramid first. The resulting parameter estimates are then used as the starting parameter values for registration using the next largest level of the pyramid. Iterating through all levels of the pyramid produces the final parameter estimate.

### 5.4. Statistical interpretation

The active blobs formulation can also be derived within a probabalistic framework. Deriving active blobs in this way gives more insight into its assumptions and limitations.

The formulations without regularization can be derived using a maximum likelihood (ML) estimator. Traditionally this equation maximizes the log-likelihood of the incoming video image given the parameters

$$\mathbf{a} = \arg \max_{\mathbf{a}} \log p(\mathbf{I}_\mathbf{v}|\mathbf{a}). \tag{31}$$

Under a Gaussian image noise assumption, the model for the incoming image $\mathbf{I}_\mathbf{v}$ is just the warped image $\mathbf{I}'_\mathbf{a}$ plus independent zero-mean Gaussian white noise for each pixel with variance $\sigma^2$

$$p(\mathbf{I}_\mathbf{v}|a) \approx \mathbf{N}(\mathbf{I}_\mathbf{v}; \mathbf{I}'_\mathbf{a}, \sigma^2 \mathbf{I}_{\text{iden}}). \tag{32}$$

Simplification of the ML equation using this pdf yields

$$\mathbf{a} = \arg \min_{\mathbf{a}} (\mathbf{I}_\mathbf{v} - \mathbf{I}'_\mathbf{a})^{\mathrm{T}} (\mathbf{I}_\mathbf{v} - \mathbf{I}'_\mathbf{a}). \tag{33}$$

This is equivalent to minimizing Eq. (11) using a quadratic error norm as the energy function.

The Lorentzian robust error norm can be shown to be equivalent to imposing a Cauchy distribution on the noise of the incoming image

$$p(\mathbf{I}_\mathbf{v}|a) = \prod_{i=0}^{n} \frac{\sigma}{\pi(\sigma^2 + (\mathbf{I}_\mathbf{v} - \mathbf{I}'_\mathbf{a})_i)}. \tag{34}$$

Each pixel is considered to be identically independently distributed, thus making the total probability the product over all of the individual pixels' probabilities. The number of pixels is signified by $n$. In this case $\sigma$ is a constant describing the width of the function, and is analogous to a variance. The Cauchy distribution is a known as a *heavy-tailed distribution* because its fall-off as the error approaches infinity is linear on a logarithmic scale. This fall-off is not as rapid as a Gaussian. An another interesting aspect of Cauchy distributions (and heavy tailed distributions in general) is

that their variance is infinite. The intuition that can be gleaned from this is that outliers very far from the mean still have a small probability of occurrence, and are thus more tolerable than under the Gaussian noise assumption. Although there are many heavy tailed distributions to choose from, the Cauchy distribution is $C^{\infty}$ continuous, and considerably simplifies the formulation.

Simplification of the ML equation using this pdf yields

$$\mathbf{a} = \arg\min_{\mathbf{a}} \sum_{i=1}^{\mathbf{n}} \log\left(\pi\sigma\left(1 + \frac{(\mathbf{I_v} - \mathbf{I'_a})_i^2}{\sigma^2}\right)\right). \tag{35}$$

This is equivalent to the active blobs formulations using the robust error norm. The $\pi\sigma$ term is a constant, and will not affect the minimization.

Adding the regularization term to the active blobs system can be shown to be equivalent to maximum a posteriori (MAP) estimation

$$\mathbf{a} = \arg\max_{\mathbf{a}} \log p(\mathbf{I_v}|\mathbf{a})p(\mathbf{a}). \tag{36}$$

Using the implicit properties of a deformable model to constrain the solution is equivalent to imposing probabilistic priors on the solution parameters. In the case of the FEM model, adding the strain energy to the total energy of the solution is equivalent to assuming a Gaussian distribution with each parameter's variance equal to the inverse of the stiffness squared ($\psi^2$) [35]

$$\mathbf{p}(\mathbf{a}) \approx \mathbf{N}\left(\mathbf{a}; 0, \frac{1}{\gamma\psi^2}\right). \tag{37}$$

Intuitively this makes sense, because the stiffer a particular parameter is, the less likely it is that it will change. Somewhat less obvious is that by choosing this prior, it imposes a re-weighting of each mode's contribution to the final solution. In general using an FEM approach on an object with uniform material properties generates a set of mode shape vectors with displacement frequency proportional to stiffness. In active blobs, this is true, so in essence the regularizer is making high-frequency components of the solution less likely, thus alleviating the blobs tendency to fit the noise present in the video.

Using this prior model and the robust error norm, the MAP estimate for the active blob parameters is given by

$$\mathbf{a} = \arg\min_{\mathbf{a}} \sum_{i=1}^{n} \log\left(\pi\sigma\left(1 + \frac{(\mathbf{I_v} - \mathbf{I'_a})_i^2}{\sigma^2}\right)\right) + \gamma \sum_{j=1}^{m} \mathbf{a}_j^2\psi^2. \tag{38}$$

This is equivalent to finding the **a** which minimizes the energy term of Eq. (14).

## 6. Implementation

Active blobs have been implemented using both the affine parameterization and the more general, FE modal parameterization. Both the Levenberg–Marquardt

and difference decomposition minimization algorithms were implemented and tested. Details of the implementation are as follows.

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 [43]. 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 [26]. In practice, a single merging pass is sufficient for a user-sketched boundary.

The triangles are then generated using an adaptation of Ruppert's Delaunay refinement algorithm [44,48], which can produce consistent meshes for 2D 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 triangulation 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 $\mathbf{I}_0$. 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 [47] 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 decomposition is precomputed. In practice, four difference images per model parameter are sufficient. For each parameter $a_i$, these four 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.

## 7. Experiments

The performance of the active blobs system was evaluated on real video sequences, as well as synthetic sequences that were designed to allow quantitative evaluation of the formulation.

### 7.1. Real video sequences

In order to demonstrate the active blobs formulation, various real world objects were tracked. The following examples show different tracking conditions that dem-

onstrate the versatility of the system. All the examples in this section are performed online, and were run on a Silicon Graphics Maximum Impact workstation.

In each figure, the region being tracked is outlined in yellow (the outline appears white in grayscale images) and is superimposed on top of the input video. The warped image of the active blob is drawn in transparent pink. In the regions where the tracking does not work well, there will be ghosting effects due to the transparent blob not matching what is underneath it. The outline and the transparent blob make it easier to verify whether or not the system is working.

Fig. 6 shows a basic example of active blobs tracking. In this example a piece of soft foam rubber was deformed. For this sequence, the difference decomposition formulation with a quadratic error norm was used. Tracking was performed using six affine shape parameters, the first six nonrigid modes of free-vibration, and two global lighting parameters (contrast and brightness), thus making a total of 14 model parameters. This example ran at 11 frames per second. The blob contained 63 vertices, 94 triangles, and had a bounding box size of $68 \times 88$ pixels. This is a good example of how a small number of modes can represent arbitrary low-frequency deformations.

Fig. 7 shows another nonrigid tracking sequence. This example used a much smaller blob and ran at 17 frames per second. The active blob had 45 vertices, 58 triangles, and a bounding box size of $85 \times 25$ pixels. Again, the difference decomposition formulation with a Gaussian error norm was used. The tracking used 10 nonrigid modes, and two global lighting parameters.

Fig. 8 shows tracking of a nonplanar surface (a coffee cup) under time-varying illumination. Again, the difference decomposition formulation with a quadratic error norm was used. Tracking was performed using 22 parameters: six affine modes, four nonrigid free-vibration modes, plus 12 parameters for the second order lighting model. This example ran at six frames per second. The active blob contained 63 vertices, 94 triangles, and had a bounding box size of $56 \times 61$ pixels. There were minor tracking errors at the edges of the active blob when the cup is tilted toward and then away from the camera. The active blob tracked accurately despite the surface nonplanarity and the changing lighting conditions.

Fig. 9 shows another sequence of nonrigid tracking of a nonplanar object. This example ran at nine frames per second. The active blob had 55 vertices, 79 triangles, and a bounding box size of $51 \times 95$ pixels. Tracking employed 10 FEM modes and two global lighting modes. The robust difference decomposition formulation was employed. This allowed the tracker to handle minor outliers. The example ran at a relatively slow six frames per second. In our experience, use of the robust error norm roughly halves the tracker's frame rate.

### *7.2. Quantitative evaluation with synthetic video sequences*

Although testing with real video demonstrates qualitatively good performance under various conditions, testing with synthetic video is required to quantitatively evaluate aspects of performance. Synthetic video was generated by animating an active blob model. Each frame was generated by slightly changing the blob's warping parameters. This change takes the form of a Gaussian random step with a predeter-

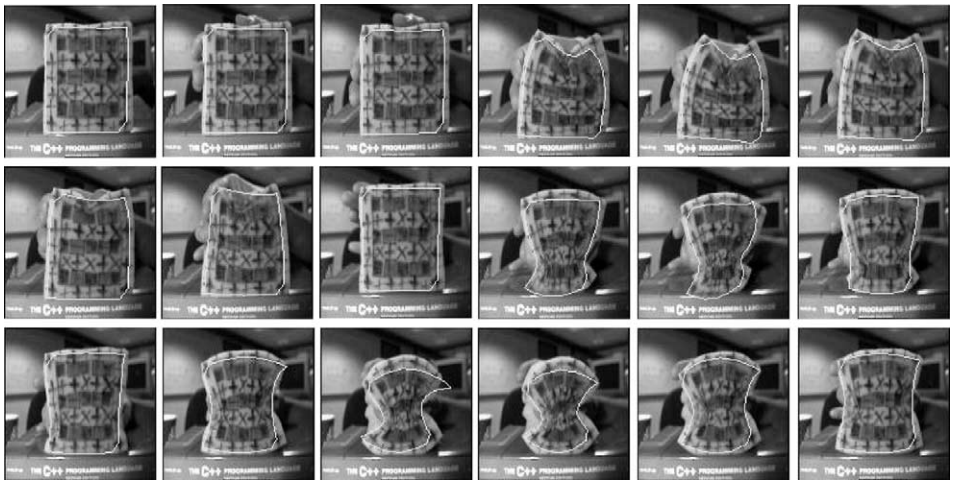

Fig. 6. Example of nonrigid tracking. Every 50th frame is shown.

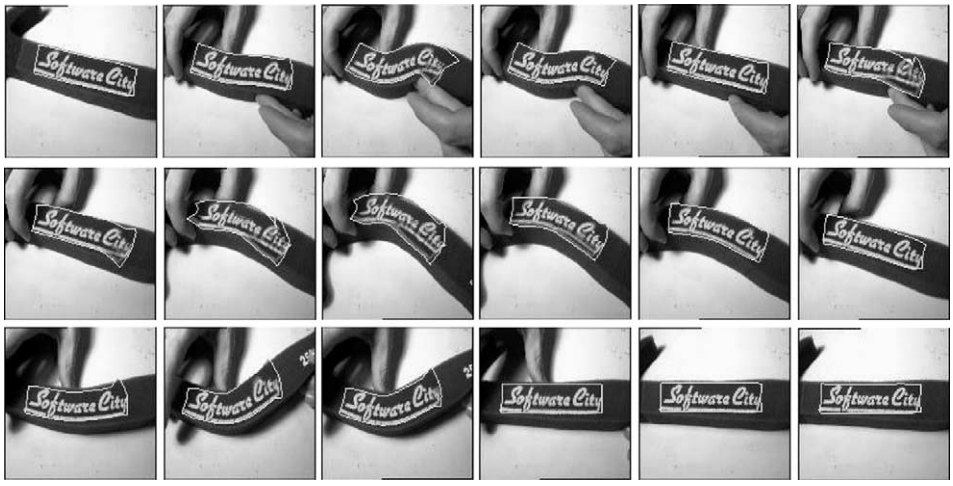


Fig. 7. Example of nonrigid tracking. Every 30th frame is shown.

mined variance for each of the parameters. Because the ground truth is known, using synthetic data allows the accuracy of the parameter estimation to be evaluated.

Experiments were designed to measure system performance with respect to varying amounts of noise, occlusion, and inter-frame motion. This allowed us to observe how the system performed as the tracking conditions gradually deteriorated. The pattern used for the generating the synthetic video data was very similar to the pattern on the object of Fig. 6.

We used a Monte-Carlo method of evaluation where the ground-truth blob takes multiple random walks through parameter space. The results over multiple trials were

Fig. 8. Tracking a nonplanar object under time-varying illumination. Every 40th frame is shown.

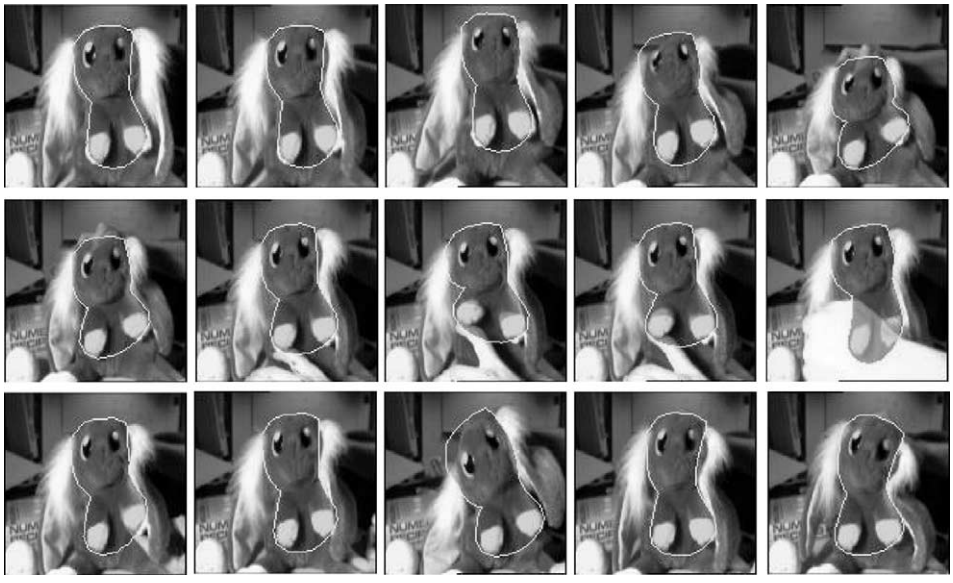

Fig. 9. Tracking sequence of a highly nonplanar deformable object. Every 20th frame is shown.

averaged. Because we wanted the results to be reproducible, the pseudo random number generators used the same set of seeds for every trial with different settings. Every time the settings were changed, 10 tests with the same settings but different seeds were

run. Each complete test (such as the sensitivity to noise test) took between 4 and 8 h to run for the difference decomposition. No synthetic data results are available for Levenberg–Marquardt because the tests would have taken several weeks to run.

The accuracy of the system is defined as the mean squared error of the parameter estimates over all the video frames. Anything less than a value of 1 can be considered as excellent tracking performance—the active blob is visually identical to the synthetic data. Between 1 and 10 corresponds to an error of roughly one pixel in some areas, which is still very good tracking performance, and given another single iteration of the tracker would most likely result in an accuracy less than 1. A result between 10 and 50 corresponds to acceptable tracking with some regions of the object not being tracked well; usually a corner is off, or a sharp bend is not approximated perfectly. Multiple extra iterations are required to get the tracker to converge. Anything above 100 can be considered as a loss of track. Convergence may still be possible with extra iterations, but it is not likely.

Another metric used to evaluate performance is the mean squared pixel error (MSPE): the mean of the squared differences between the intensity values of the warped active blob and the video. If the MSPE is greater than the noise level present in the image, then the blob is assumed to have lost the track. However, a lower MSPE does not necessarily mean better tracking performance. In some cases, the blob may contain a repetitive pattern, and large inter-frame motion may cause the blob to track another portion of the pattern with a low pixel error. Also, large single-colored regions on the object to be tracked can cause ambiguities in tracking. Therefore it is important to consider both the parameter estimation error as well as the pixel error (MSPE) in our quantitative evaluation.

*7.2.1. Sensitivity to noise*

To test the sensitivity of the system to sensor noise (assumed to be Gaussian), synthetic data was generated with Gaussian white noise added to every pixel. A $60 \times 50$ pixel rectangular blob was tracked using 20 modes. Trials were run with additive Gaussian noise, with variances ranging from 0.0 to 4.0 intensity levels. The graphs in Fig. 10 show that system performance has a smooth degradation with respect to noise.

Using a quadratic error norm, the system can handle Gaussian noise with a variance up to the entire intensity scale. In most real world conditions, the incoming video would rarely contain this much noise. The use of the robust error norm seems to reduce the system's resilience to Gaussian noise. A possible reason for this is that the robust error norm tends to lower the influence of the pixels containing high intensities in the difference image (the blob image minus the incoming video). These high intensity pixels are the ones that correspond to the movement of sharp edges in the video, and which would be least susceptible to additive noise.

*7.2.2. Sensitivity to outliers*

In order to test the sensitivity of the system to outliers, the system was subjected to partial occlusion over a certain percentage of its area. The percentage area covered by outliers was varied between 0 and 70%. The occluder took the form of a rectangle with

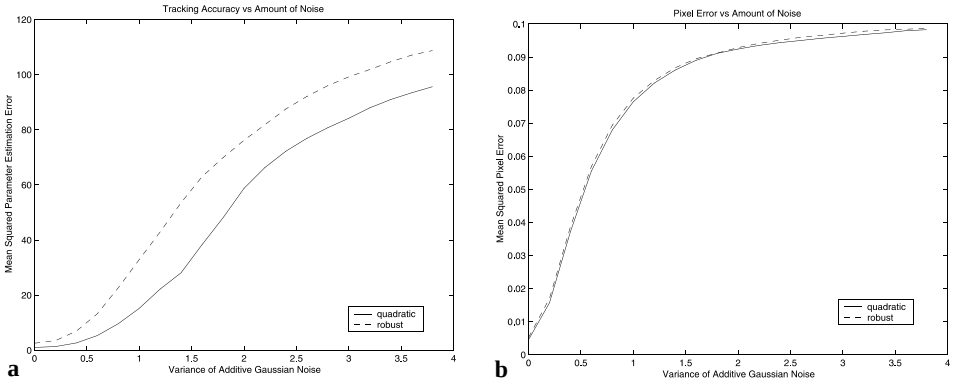


Fig. 10. Performance under varying noise conditions.

zero-mean, unit-variance Gaussian white noise in each color channel of each pixel value. The reason for using this noise pattern to generate the outliers is to attempt to simulate a worst case scenario. For this test, 10 modes were used for tracking.

Fig. 11 shows the results. As it should, the robust error norm out-performed the quadratic norm. In fact, even a very small percentage of occlusion caused the quadratic norm to fail miserably. The robust error norm handled up to 30% outliers with almost no degradation in performance. This is consistent with the literature on robust statistics for the Lorentzian error norm.

Finally, this test demonstrates how a lower pixel error may not indicate better performance. At first glance, the results in Fig. 11b show unusual behavior. Above 50% outliers, the pixel error for the quadratic error norm tracker is lower than the robust one. However, as can be seen in Fig. 11a, the robust error norm produces better estimates of the deformation parameters, no matter what the percentage of outliers. The quadratic norm produced a superior MSPE because it is specifically designed

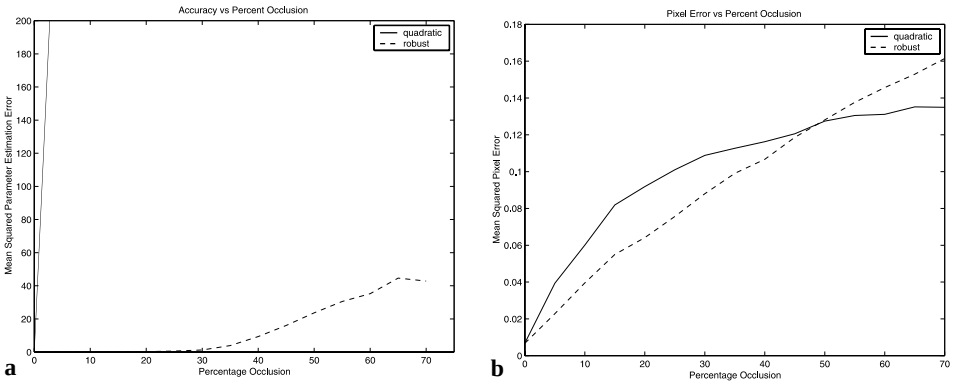


Fig. 11. Performance under varying percentage occlusion.

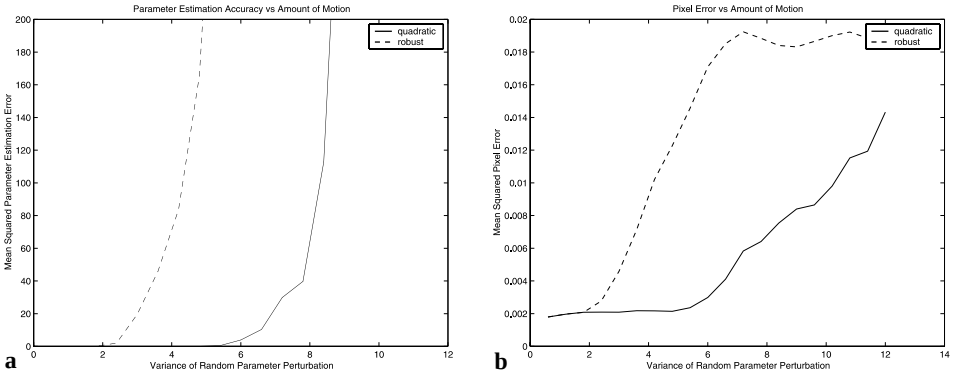


Fig. 12. Performance under varying amounts of inter-frame motion.

to minimize the pixel error; unfortunately, it tries to fit the model to the outliers rather than ignoring them, and this leads to poor estimates of the deformation parameters.

*7.2.3. Tracking vs. amount of inter-frame motion*

Tracking performance was also evaluated with respect to the amount of inter-frame motion. This test was conducted using a $96 \times 64$ blob, using 10 modes. The random walk had perturbations with variance ranging from 0 to 2 times the $\delta$ used to generate the difference images. Gaussian pyramids were disabled for this test. With Gaussian pyramids enabled, every additional level of the pyramid doubles the amount of motion the tracker can handle provided that the decimated image contains sufficient texture detail for tracking.

The results in Fig. 12 show that the system is able to handle one $\delta$ worth of motion per iteration using the quadratic error norm. Using the robust error norm makes the tracker only able to handle roughly half that amount of motion. The reason for this is again due to the fact that the robust error norm lowers the influence of the high intensity pixels in the difference image.

## 8. Discussion

Depending on the tracking conditions robust estimation can drastically improve performance. As shown in the experiments, the robust formulation enables accurate tracking even when the blob has major occlusions. Because of this the robust tracker is less likely to lose the track, and is capable of ignoring slight inconsistencies in the model due to data outliers, or model inadequacies. However, the robust technique cannot handle as large inter-frame motion per iteration as the quadratic version. A simple fix for this is to allow the robust tracker more iterations per frame at the expense of a slower overall tracking speed. Also the robust approach takes more computation per iteration, usually taking twice as long as the quadratic one.

### 8.1. Levenberg–Marquardt vs. difference decomposition

Generally Levenberg–Marquardt should work better in some cases because it does not rely on the local linearity assumption that difference decomposition relies on. However, even though the derivations are significantly different, the resulting equation for a single iteration of this technique is very similar to difference decomposition with a Tikhonov $L = I$ regularizer. The $\lambda$ in the Levenberg–Marquardt equation acts like the influence on the scaled identity matrix added to the approximate Hessian. Because of this, Levenberg–Marquardt can perform better than the difference decomposition when both are allowed the same number of iterations. With each step the $\lambda$ is changed, which makes this numerical technique act like it has some form of heuristically adjusted regularizer. Per iteration, Levenberg–Marquardt runs at up to two frames per second; however, it usually takes 20–50 iterations per pyramid level to converge. Thus the Levenberg–Marquardt approach is only suitable for off-line tracking.

### 8.2. Selection of regularization parameter $\gamma$

Choosing the optimal $\gamma$ is a challenging problem, because in some ways it depends on what kind of tracker behavior is desired. The larger the value of $\gamma$, the stiffer and more resilient to deformation the blob is.

One possible method of choosing $\gamma$ is the use of the L-curve [22]. The L-curve is a method of finding the point where the both the solution error (in this case pixel error) and the solution norm (in this case the error in parameter space) are jointly minimized. To generate an L-curve, solutions to the problem (in this case one frame of tracking) with various values for $\gamma$ are generated. On a log–log graph the vertical axis is the pixel error, while the horizontal axis is the accuracy. The optimal $\gamma$ is the one which at the point of maximum curvature on the L-curve.

The value of $\gamma = 10^{-5}$ works well for all cases we have tested. A value of $\gamma = 10^{-4}$ makes the system stiff to the point where the blob can barely move, and a value of $\gamma = 10^{-6}$ makes the system too unstable when large numbers of modes are employed.

### 8.3. Performance issues and bottlenecks

The system has been shown to run at speeds of up to 20 frames per second on inexpensive PC hardware, with the main speed limitation being buffer copies. Both the video to texture memory, and the graphics buffer to memory transfers have a large performance impact. Hardware video texture mapping can drastically improve performance when tracking is done in texture space using inverse warping. However, it is important to note that our experiments were conducted on PCs with older graphics cards.

Performance could be significantly enhanced by utilizing the higher precision pixel pipe, and programmable pixel shader functionality of the most recent generation of graphics cards. Using this, it is possible to perform down-sampling, image subtraction and dot product operations, as well as many other vectorizable matrix operations completely on the graphics card. Due to the fact that many recent graphics

cards also have video inputs and hardware video texturing, the $\mathbf{B}^T\mathbf{D}$ matrix for all levels of the Gaussian pyramid can be computed entirely on the graphics card per frame. We expect that this approach would reduce the need to transfer between CPU and graphics card.

Given any particular choice of minimizer and error norm, tracking speed is dependent mostly on two properties: the number of pixels included in the bounding box of the blob, and the number of deformation parameters employed in tracking. The amount of computation needed per minimization iteration scales approximately linearly with the number of pixels. Computation vs. the number of parameters scales quadratically. However, with the small number of parameters used for active blobs (at most 30 in our experiments) the number of parameters does not have as significant an impact on performance as blob size. Changing the granularity of the triangular tessellation used in the blob affects tracking speed only marginally.

## 9. Conclusion

The active blobs formulation has been shown to handle the nonrigid 2D tracking problem effectively, while providing some ability to track nonplanar 3D objects as well. In addition to this, the use of the robust error norm provides resilience of the system to outliers caused by shadows, highlights, and partial occlusions. The use of texture mapping hardware allows the system to achieve near realtime performance while still using full-color images. Despite these positive aspects of the formulation, there are a number of issues that remain for future work.

The first issue is that the object to be tracked needs some sort of strong surface texture or pattern. In the real world, some nonrigid objects do not have strong enough texture, or even may be a single color. In these situations only the contour of the object is a reliable feature to track, and snakes may work better than a registration-based system like active blobs. Conversely, snakes tend to fail when the object and/or the background are highly textured. In such cases, the snake may find edges which are not on the contour of the object, and for this reason the snake loses the track. Therefore, a hybrid formulation that includes aspects of both active blobs and snakes may address this issue.

Another issue that needs to be addressed is that of large inter-frame motions. The active blobs formulation can handle significant inter-frame motion through the use of a Gaussian pyramid, as described in Section 5.3. However, in practice there is a limit on the amount of inter-frame motion that the formulation can handle. Inter-frame motions that are larger than roughly 1/16 the blob's size are difficult to track. This limitation is due to the fact that the lowest pyramid level must have an image size of at least $16 \times 16$ in order to retain sufficient texture for active blob tracking. This problem can be alleviated only somewhat by reducing the number of degrees of freedom for the active blob. Possible solutions to this problem include predictive tracking via Kalman filtering or particle filtering.

Another open issue is that of extending the active blobs formulation to 3D models. The basic active blobs approach in this paper has already been extended for use

in tracking human head motion with a 3D cylindrical model [10]. In this case, the active blobs model was embedded in the 3D surface texture map. Application of the active blobs formulation to tracking of general 3D deformable solids remains an issue for future work. The biggest problem will be automatic initialization of the 3D model, and the increased degrees of freedom in the model. In specific applications (like head tracking) domain constraints can be used to gain automatic model initialization, and restrict the degrees of freedom.

## Acknowledgments

This work was 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 was sponsored in part through the U.S. Department of Education (GAANN).

## References

[1] A.A. Amini, S. Tehrani, T.E. Weymouth, Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints, in: Proc. Internat. Conf. on Comput. Vision, 1988, pp. 95–99.
[2] S. Baker, I. Matthews, Equivalence and efficiency of image alignment algorithms, in: Proc. IEEE Comput. Vision and Pattern Recognition Conf., vol. 1, 2001, pp. 1090–1097.
[3] A. Baumberg, D. Hogg, Generating spatiotemporal models from examples, Image and Vision Comput. 14 (8) (1996) 525–532.
[4] M. Betke, N.C. Markis, Fast object recognition in noisy images using simulated annealing, in: Proc. Internat. Conf. on Comput. Vision, 1995, pp. 523–530.
[5] M.J. Black, D.J. Fleet, Y. Yacoob, Robustly estimating changes in image appearance, Comput. Vision Image Understanding 78 (1) (2000) 8–31.
[6] M.J. Black, A. Rangarajan, On the unification of line processes, outlier rejection, and robust statistics with applications in early vision, Internat. J. Comput. Vision 19 (1) (1996) 57–91.
[7] M.J. Black, Y. Yacoob, Recognizing facial expressions in image sequences using local parameterized models of image motion, Internat. J. Comput. Vision 25 (1) (1997) 23–48.
[8] A. Blake, R. Curwen, A. Zisserman, A framework for spatiotemporal control in the tracking of visual contours, Internat. J. Comput. Vision 11 (2) (1993) 127–146.
[9] A. Blake, A. Zisserman, Visual Reconstruction, MIT Press, Cambridge, MA, 1987.
[10] M. La Cascia, S. Sclaroff, V. Athitsos, Fast reliable head tracking under varying illumination: an approach based on registration of textured-mapped 3D models, IEEE Trans. Pattern Anal. Machine Intell. 22 (4) (2000) 322–336.
[11] L. Cohen, On active contour models and balloons, CVGIP: Image Understanding 53 (2) (1992) 211–218.
[12] T.F. Cootes, D. Cooper, C. Taylor, J. Graham, Trainable method of parametric shape description, Image and Vision Comput. 10 (5) (1992) 289–294.
[13] T.F. Cootes, G.J. Edwards, C.J. Taylor, Active appearance models, IEEE Trans. Pattern Anal. Machine Intell. 23 (6) (2001) 681–684.
[14] T.F. Cootes, C.J. Taylor, Combining point distribution models with shape models based on finite-element analysis, Image and Vision Comput. 13 (5) (1995) 403–409.
[15] D. de Carlo, D. Metaxas, Optical flow constraints on deformable models with applications to face tracking, Internat. J. Comput. Vision 38 (2) (2000) 99–127.

[16] F. Dellaert, S. Thrun, C. Thorpe, Jacobian images of super-resolved texture maps for model based motion estimation and tracking, in: Proc. IEEE Workshop on Appl. Comput. Vision, 1998.
[17] I.A. Essa, A.P. Pentland, Coding, analysis, interpretation and recognition of facial expressions, IEEE Trans. Pattern Anal. Machine Intell. 19 (7) (1997) 757–763.
[18] S. Geman, D. Geman, Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images, IEEE Trans. Pattern Anal. Machine Intell. 6 (11) (1984).
[19] M. Gleicher, Projective registration with difference decomposition, in: Proc. IEEE Comput. Vision and Pattern Recognition Conf., 1997, pp. 331–337.
[20] G.D. Hager, P.N. Belhumeur, Efficient region tracking with parametric models of geometry and illumination, IEEE Trans. Pattern Anal. Machine Intell. 20 (10) (1998) 1025–1039.
[21] F. Hampel, E. Ronchetti, P. Rousseeuw, W. Stehel, Robust Statistics: The Approach Based on Influence Functions, Wiley, New York, 1986.
[22] P. Hansen, Analysis of discrete ill-posed problems by means of the L-curve, SIAM Rev. 34 (4) (1992) 561–680.
[23] M. Irani, S. Peleg, Improving resolution by image registration, CVGIP: Graphical Models Image Process. 53 (1991) 231–239.
[24] J. Isidoro, S. Sclaroff, Active voodoo dolls: a vision based input device for non-rigid control, in: Proc. Comput. Animation, 1998, pp. 137–143.
[25] A.K. Jain, Y. Zhong, S. Lakshmanan, Object matching using deformable templates, IEEE Trans. Pattern Anal. Machine Intell. 18 (3) (1996) 267–278.
[26] R. Jain, R. Kasturi, B. Shunck, Machine Vision, McGraw-Hill Inc., New York, 1995.
[27] M. J, Jones, T. Poggio, Multidimensional morphable models: a framework for representing and matching object classes, Internat. J. Comput. Vision 29 (2) (1998) 107–131.
[28] M. Kass, A. Witkin, D. Terzopoulos, Snakes: active contour models, Internat. J. Comput. Vision 1 (1987) 321–331.
[29] G. Kimeldorf, G. Wahba, A correspondence between Bayesian estimation and on stochastic processes and smoothing by splines, Ann. Math. Statist. 41 (2) (1970) 495–502.
[30] O. Lee, Y. Wang, A. Vetro, Use of two-dimensional deformable mesh structures for video coding, part II – the analysis problem and a region-based coder employing an active mesh representation, IEEE Trans. Circuits and Systems for Video Technol. 6 (6) (1996) 647–659.
[31] H. Li, P. Roivainen, R. Forchheimer, A3D motion estimation in model-based facial image coding, IEEE Trans. Pattern Anal. Machine Intell. 15 (6) (1993) 545–555.
[32] J. Maintz, M. Viergever, A survey of medical image registration, Med. Image Anal. 2 (1) (1998) 1–36.
[33] S. Malassiotis, M.G. Strintzis, Tracking textured deformable objects using a finite-element mesh, IEEE Trans. Circuits and Systems for Video Technol. 8 (6) (1998).
[34] R. Malladi, J.A. Sethian, B.C. Vemuri, Shape modeling with front propagation: a level set approach, IEEE Trans. Pattern Anal. Machine Intell. 17 (2) (1995) 158–175.
[35] J. Martin, A. Pentland, S. Sclaroff, R. Kikinis, Characterization of neuropathological shape deformations, IEEE Trans. Pattern Anal. Machine Intell. 20 (2) (1998) 97–112.
[36] T. McInerny, D. Terzopoulos, Deformable models in medical image analysis: a survey, Med. Image Anal. 1 (2) (1996) 91–108.
[37] D. Metaxas, D. Terzopoulos, Shape and nonrigid motion estimation through physics-based synthesis, IEEE Trans. Pattern Anal. Machine Intell. 15 (6) (1993) 580–591.
[38] C. Nastar, B. Moghaddam, A.P. Pentland, Flexible images: matching and recognition using learned deformations, Comput. Vision Image Understanding 65 (2) (1997) 179–191.
[39] J.M. Odobez, P. Bouthemy, Robust multiresolution estimation of parametric motion models, J. Visual Commun. Image Representation 6 (1995) 348–365.
[40] A. Pentland, B. Horowitz, Recovery of non-rigid motion and structure, IEEE Trans. Pattern Anal. Machine Intell. 13 (7) (1991) 730–742.
[41] A. Pentland, S. Sclaroff, Closed-form solutions for physically-based shape modeling and recognition, IEEE Trans. Pattern Anal. Machine Intell. 13 (7) (1991) 715–729.
[42] W. Press, B. Flannery, S. Teukolsky, W. Vetterling, Numerical Recipes in C, Cambridge University Press, Cambridge, UK, 1988.

[43] A. Rosenfeld, A. Kak, Digital Picture Processing, Academic Press, New York, 1976.
[44] J. Ruppert, A Delaunay refinement algorithm for quality 2-dimensional mesh generation, J. Algorithms 18 (3) (1995) 548–585.
[45] S. Sclaroff, J. Isidoro, Active blobs, in: Proc. Internat. Conf. on Comput. Vision, 1998, pp. 1146–1153.
[46] S. Sclaroff, A. Pentland, Physically-based combinations of views: representing rigid and nonrigid motion, in: Proc. IEEE Workshop on Nonrigid and Articulate Motion, Austin, TX, 1994.
[47] S. Sclaroff, A. Pentland, Modal matching for correspondence and recognition, IEEE Trans. Pattern Anal. Machine Intell. 17 (6) (1995) 545–561.
[48] J.R. Shewchuk, Triangle: engineering a 2D quality mesh generator and Delaunay triangulator, in: Proc. ACM First Workshop on Appl. Comput. Geometry, Philadelphia, 1996, pp. 124–133.
[49] L.H. Staib, J.S. Duncan, Boundary finding with parametrically deformable models, IEEE Trans. Pattern Anal. Machine Intell. 14 (11) (1992) 1061–1075.
[50] R. Szeliski, Bayesian Modeling of Uncertainty in Low-Level Vision, Kluwer Academic Publishers, Dordrecht, 1989.
[51] R. Szeliski, Video mosaics for virtual environments, IEEE Comput. Graphics Appl. 16 (2) (1996) 22–30.
[52] R. Szeliski, H.Y. Shum, Motion estimation with quadtree splines, IEEE Trans. Pattern Anal. Machine Intell. 18 (12) (1996) 1199–1207.
[53] H. Tao, T.S. Huang, Connected vibrations: A modal analysis approach to non-rigid motion tracking, in: Proc. IEEE Comput. Vision and Pattern Recognition Conf., 1998, pp. 735–740.
[54] D. Terzopoulos, Regularization of inverse visual problems involving discontinuities, IEEE Trans. Pattern Anal. Machine Intell. 8 (4) (1986) 413–424.
[55] D. Terzopoulos, On matching deformable models to images: direct and iterative solutions, in: Topical Meeting on Machine Vision, Technical Digest Series, vol. 12, Optical Society of America, Washington, DC, 1987, pp. 160–167.
[56] D. Terzopoulos, K. Waters, Analysis and synthesis of facial image sequences using physical and anatomical models, IEEE Trans. Pattern Anal. Machine Intell. 15 (6) (1993) 569–579.
[57] D. Terzopoulos, A. Witkin, M. Kass, Symmetry-seeking models for 3D object reconstruction, in: Proc. Internat. Conf. on Comput. Vision, London, England, 1987, pp. 269–276.
[58] C. Toklu, A.T. Erdem, M.I. Sezan, A.M. Tekalp, Tracking motion and intensity variations using hierarchical 2-D mesh modeling for synthetic object transfiguration, Graphical Models Image Process. 58 (6) (1996) 553–573.
[59] P. van Beek, A.M. Tekalp, N. Zhuang, I. Celasun, M. Xia, Hierarchical 2-D mesh representation, tracking, and compression for object-based video, IEEE Trans. Circuits and Systems for Video Technol. 9 (2) (1999) 353–369.
[60] T. Vetter, T. Poggio, Linear object classes and image synthesis from a single example image, IEEE Trans. Pattern Anal. Machine Intell. 19 (7) (1997) 733–742.
[61] P.A. Viola, W.M. Wells, Alignment by maximization of mutual information, Internat. J. Comput. Vision 24 (2) (1997) 137–154.
[62] Y. Wang, O. Lee, Active mesh – a feature seeking and tracking image sequence representation scheme, IEEE Trans. Image Process. 3 (5) (1994).
[63] Y. Wang, O. Lee, Use of two-dimensional deformable mesh structures for video coding, part I – the synthesis problem: mesh-based function approximation and mapping, IEEE Trans. Circuits Systems for Video Technol. 6 (6) (1996) 636–646.
[64] D.J. Williams, M. Shah, A fast algorithm for active contours and curvature estimation, CVGIP: Image Understanding 55 (1) (1992) 14–26.
[65] A.L. Yuille, D.S. Cohen, P.W. Hallinan, Feature extraction from faces using deformable templates, Internat. J. Comput. Vision 8 (2) (1992) 99–111.
[66] L. Zhou, C. Kambhamettu, Hierarchical structure and nonrigid motion recovery from 2D monocular views, in: Proc. IEEE Comput. Vision and Pattern Recognition Conf., vol. 2, 2000, pp. 752–759.