GRAPHICAL MODELS AND IMAGE PROCESSING
Vol. 58, No. 6, November, pp. 553–573, 1996
ARTICLE NO. 0046

Tracking Motion and Intensity Variations Using Hierarchical 2-D Mesh Modeling for Synthetic Object Transfiguration[1]

CANDEMIR TOKLU

Department of Electrical Engineering and Center for Electronic Imaging Systems, University of Rochester, Rochester, New York 14627

A. TANJU ERDEM[2]

Department of Electrical and Electronics Engineering, Bilkent University, Ankora, Turkey 06533

M. IBRAHIM SEZAN[2]

Sharp Laboratories of America, Inc., 5700 N. W. Pacific Rim Blvd., Camas, Washington 98607

AND

A. MURAT TEKALP[3]

Department of Electrical Engineering and Center for Electronic Imaging Systems, University of Rochester, Rochester, New York 14627

Received November 9, 1995; revised May 16, 1996; accepted August 1, 1996

We propose a method for tracking the motion and intensity variations of a 2-D mildly deformable image object using a hierarchical 2-D mesh model. The proposed method is applied to synthetic object transfiguration, namely, replacing an object in a real video clip with another synthetic or natural object via digital postprocessing. Successful transfiguration requires accurate tracking of both motion and intensity (contrast and brightness) variations of the object-to-be-replaced so that the replacement object can be rendered in exactly the same way from a single still picture. The proposed method is capable of tracking image regions corresponding to scene objects with nonplanar and/or mildly deforming surfaces, accounting for intensity variations, and is shown to be effective with real image sequences.

1. INTRODUCTION

Synthetic transfiguration refers to digital post-processing for replacing a moving image object in a real video clip with synthetic (e.g., text and 2-D graphics) and/or natural (e.g., still image) content. It is often employed in movie post-production, advertising, training in virtualized environments, computer guided maintenance and repair, and consumer design, where it facilitates the generation of ''augmented reality.'' For example, it is possible to render a new dress with different fabric textures or with virtual necklaces before ordering. Synthetic transfiguration thus requires accurate tracking of the *boundary*, *local motion*, *and intensity variations* of the ''object-to-be-replaced'' in the original video clip, and rendering the ''replacement object'' with the same local motion and intensity variations.

Boundary tracking has been addressed in [1–3] using a locally deformable contour model (a.k.a. snakes [4]), and in [5] using a locally deformable template model. The snake model employed in [1] includes only intra-frame energy constraints, such as edge energy, while [2] uses both intra- and interframe constraints. Both methods, however, lack the ability of tracking rapidly moving objects because they do not have a prediction mechanism to initialize the snake.

[1] This work is supported in part by a National Science Foundation SIUCRC grant and a New York State Science and Technology Foundation grant to the Center for Electronic Imaging Systems at the University of Rochester, and a grant by Eastman Kodak Company.

[2] Formerly with Imaging Research and Advanced Development, Eastman Kodak Company, Rochester, NY 14650-1816.

[3] To whom correspondence should be addressed. E-mail: tekalp@ee.rochester.edu.

1077-3169/96 $18.00

In order to handle large motion, [3] employs a region-based motion prediction to guide the snake into a subsequent frame. However, the prediction relies on a global (i.e., not locally varying) motion assumption, and thus may not be satisfactory when there are local deformations within the boundary. The deformable template model proposed in [5] is shown to be successful in tracking locally deformable objects and partly handling occlusions. However, the method of [5] requires a training phase and thus it is applicable only for long image sequences.

Local motion within a region of interest (ROI) is commonly estimated by means of 2-D mesh models, which first became popular in very low bit-rate video coding applications. A motion compensation method based on control grid interpolation for use in video compression is proposed in [6] where the nodes of a mesh with rectangular patches have been used as the control grids. Nakaya and Harashima suggested a locally optimum search method for 2-D mesh based motion estimation in [7]. Adaptive extensions of the mesh models have been proposed in [8–10] based on node point selection and energy optimization. Hierarchical extensions of mesh models based on quadtree decomposition have been studied in [11, 12]. Szeliski and Shum also proposed a hierarchical mesh model for motion estimation in [12]. They have suggested a global closed form solution using conjugate gradient optimization. However, in their solution, the preservation of the connectivity of the mesh elements is not guaranteed. Furthermore, none of the above methods address tracking of the ROI itself (because they treat the whole frame as the ROI), and do not consider frame-to-frame intensity variations.

Intensity variations in motion estimation are accounted for in [12–14] through the use of contrast and brightness parameters. However, these methods assume that the same intensity variation applies to all pixels; i.e., they do not allow for spatial variation in the values of these parameters. It should also be pointed out that the literature on simultaneous tracking of local motion and intensity variations is very limited.

In this paper, we present a hierarchical 2-D mesh tracking method (without the need for long sequences) for tracking the boundary, local motion, and intensity variations of an ROI. The results clearly demonstrate the importance of local compensation of intensity variations, as well as the advantages of a hierarchical mesh model for accurate object tracking. In our formulation, the ROI, object-to-be-replaced, in the reference frame is first enclosed by a polygon (called the reference polygon). The reference polygon may exactly coincide with the outline of the ROI, or it may be a larger polygon containing the ROI when the ROI has an irregular shape. A 2-D mesh (called the reference mesh) is then fit to the reference polygon. The tracking of the reference mesh in a subsequent frame is guided by the estimated motion at the corners of the reference polygon. This initialization of the mesh in a subsequent frame is referred to as mesh propagation. The motion (or displacement) at the corners of the reference polygon is estimated using a novel block matching algorithm which employs motion models ranging in complexity from translation to perspective transformation. It is assumed that the actual boundary of the ROI in a subsequent frame remains completely inside the displaced reference polygon in that frame. Next, the initial locations of the nodes of the mesh are refined using a hierarchical scheme based on image intensity matching. The hierarchical approach employs a set of coarse-to-fine mesh models; it thus reduces the sensitivity of the proposed method to the size of mesh elements and provides significant computational savings. The proposed method compensates for intensity variations using a spatially continuous model. It associates a set of contrast and brightness parameters with each node of the 2-D mesh and bilinearly interpolates the contrast and brightness parameters for each pixel within a patch from those of the nodes of the patch.

The main technical contributions of this paper, besides successful application of object tracking to synthetic transfiguration of non-planar and/or mildly deforming objects, therefore are: (i) a new method to propagate the mesh structure from one frame to another without the need for motion estimation at each node of the mesh, described in Section 3.1; (ii) a hierarchical implementation of the mesh refinement process, described in Section 3.2; (iii) incorporation of novel convexity constraints into the search procedure to preserve the connectivity of the mesh, discussed in Section 3.2.1; and (iv) compensation of intensity variations based on a spatially continuous model, described in Section 2.4. The organization of the paper is as follows: Section 2 presents the models employed in the proposed object tracking method. The details of the tracking method are discussed in Section 3. Synthetic object transfiguration is addressed in Section 4. Finally, experimental results are given in Section 5 to demonstrate the effectiveness of the proposed method in real life applications.

List of Symbols

$\boldsymbol{T}, \boldsymbol{Z}$	A 2-D spatial transformation, ranging in complexity from translation to perspective transformation
$\boldsymbol{H}$	A 2-D affine or bilinear transformation
$\boldsymbol{A}$	A 2-D affine transformation
$\mathscr{R}$	Set of real numbers

$\boldsymbol{x}$, (x, y)	$\boldsymbol{x}$ is a 2-D point, (x, y) are its coordinates
$I_R(\cdot)$, $I_c(\cdot)$, I_{new}	Reference and current frames, and the still image of replacement object, respectively
γ, η	Contrast and brightness parameters
δ	Logarithmic search step size
α	Accuracy of node displacement estimates
Δ	A member of the triangular partitioning of the reference polygon
w_i, w_b, w_c	Search window size for inner, boundary, and corner nodes, respectively
G	A node of the mesh
S	Cost polygon of a node
F	Search space of a node
P_R, P_c, P_{new}	Reference, current, and replacement polygons, respectively
M_R, M_c	Reference and current meshes, respectively
$\delta(\cdot)$	Kronecker delta function

2. MODELING

The proposed method for object tracking and synthetic transfiguration is composed of (i) selection of the reference frame and the reference polygon bounding the ROI, (ii) fitting a 2-D mesh to the reference polygon, (iii) motion estimation at the corners of the reference polygon, (iv) propagation of the mesh to the next frame, (v) hierarchical mesh refinement, and (vi) transfiguration. The reference frame is selected to be the one in which the view of the ROI best matches the perspective of the still image of the replacement object. The reference polygon, which is completely specified by the coordinates of its corners, should enclose the ROI as tightly as possible. The global motion of the reference polygon is described by the displacement vectors at the corner points, whereas local motion within the polygon is described by the displacement vectors at the node points of the 2-D mesh fitted within. In the following, we present models used in tracking the corners of the reference polygon, and tracking the local motion and intensity variations within the reference polygon.

2.1. Motion Models for Tracking the Reference Polygon

In order to find the displacement vectors at the corner points of the reference polygon (hence the global motion of the reference polygon) from the reference frame to any other frame, we define a region S about each corner point (see Fig. 1) such that all pixels within S obey a single spatial transformation $\boldsymbol{T}$, which may range in complexity from simple translation to a bilinear transformation. We allow the corners to have different types of transformation. At each corner point, first a *translational* motion model, given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \boldsymbol{T}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad a_1, a_2 \in \mathscr{R} \tag{1}$$

is tested, where (x, y) and (u, v) are the coordinates of a point before and after a spatial transformation, respectively, (a_1, a_2) are the translation parameters, and $\mathscr{R}$ denotes the set of real numbers. If the model (1) is not deemed accurate enough to represent the motion of all pixels within R, then we consider a *translation–rotation–zoom* model, given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & -a_2 \\ a_2 & a_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_3 \\ a_4 \end{bmatrix}, \quad a_1, \ldots, a_4 \in \mathscr{R} \tag{2}$$

with the parameters $(a_1, \ldots, a_4)$. The motion model that is next in complexity is the 6-parameter *affine* model

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}, \tag{3}$$

which adds shear and directional scaling to the translation–rotation–zoom model. We also consider two 8-parameter

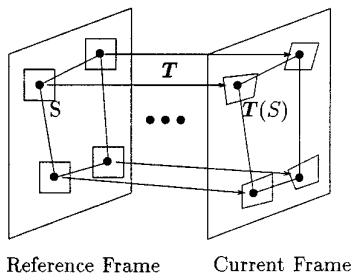


FIG. 1. Tracking of reference polygon via its corners. S denotes the cost polygon of a corner in the reference frame and $\boldsymbol{T}(S)$ denotes its deformed version in the current frame.

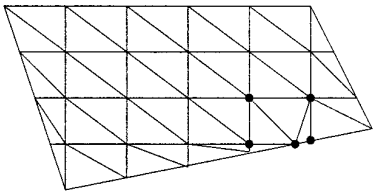


FIG. 2. Division of reference polygon to triangular elements. A pentagon whose corners are marked is divided into three triangles.

non-linear spatial transformations, the *perspective* and *bilinear* transformations, which are given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{a_{31}x + a_{32}y + 1}\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}\right), \quad a_{11}, \ldots, a_{32} \in \mathscr{R}, \quad (4)$$

and

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix}\begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix}, \quad a_1, \ldots, b_4 \in \mathscr{R}. \quad (5)$$

respectively. Once the most accurate spatial transformation $\boldsymbol{T}^*$ at each corner point is obtained, as described in Section 3.1.1, the displacement vector $\boldsymbol{d}_C$ of a corner point C is computed as

$$\boldsymbol{d}_C = \boldsymbol{T}^*\boldsymbol{x}_C - \boldsymbol{x}_C, \quad (6)$$

where $\boldsymbol{x}_C$ denotes the coordinates of C in the current frame.

2.2. Modeling Local Motion

We employ a 2-D mesh model to keep track of the local variations in motion and intensity within the reference polygon. The 2-D mesh, which may be composed of triangular and/or rectangular elements (patches), fitted to the reference polygon in the reference frame will be called as the reference mesh. We design a uniform (regular) mesh by overlaying the reference polygon on a regular rectangular grid with a predetermined initial density of grid points as depicted in Fig. 2. A number of non-rectangular patches (e.g., trapezoids and pentagons) may be formed along the boundary of the reference polygon. These are further divided into an appropriate number of triangles (see Fig. 2). For instance, a trapezoid is divided into two triangles, a pentagon is divided into three triangles, etc. A reference mesh thus formed will be referred to as a *quadrilateral* mesh. A *triangular* reference mesh is formed by further dividing each rectangular patch into two triangle patches.

The basic premise in modeling the local motion by means of a 2-D mesh is that the motion within each patch can be accurately represented by a single spatial transformation. This, however, depends on the density of grid (node) points. Motion uniformity calls for smaller patches (higher grid point density). However, as the patch size gets smaller, the accuracy of local motion estimation may suffer. This is because the likelihood of an erroneous match increases as the number of data points (i.e., pixels) within a patch decreases. Furthermore, a dense set of nodes would result in an increased computational load. To satisfy these conflicting requirements, we propose a hierarchical mesh structure that starts with an initial patch size and successively reduces the patch size by a factor of two until satisfactory motion compensation is achieved. The advantages of hierarchical mesh modeling are (i) the tracking accuracy is less sensitive to the initial grid point density, and (ii) the procedure is computationally more robust and efficient, as larger local motion can be tracked in less time using larger patches, and smaller patches can track local motion more accurately when their initial location is determined by larger patches.

The spatial transformation (motion model) to be employed within a patch is related to the geometry of the patch. We use affine (3) and bilinear (5) mappings in triangular and quadrilateral patches, respectively. This is because given three (four) point correspondences (at the respective vertices of patches), the parameters of an affine (bilinear) transformation can be uniquely determined by solving a system of three (four) linear equations [15]. The transformed coordinates of a point within the patch can be subsequently calculated using the affine (bilinear) mapping. The choice of rectangular patch with bilinear mapping vs triangular patch with affine mapping depends on the local surface geometry of the scene objects. The affine transformation models 3-D rigid motion of a planar surface in the image plane under the orthographic projection, whereas the bilinear transformation approximates 3-D rigid motion of a quadratic surface under the orthographic projection [16, 17]. It is important to note that the bilinear transformation (with rectangular patches) and affine transformation preserve continuous image intensity distributions across patch boundaries (due to their ratio preserving properties [15], a property not shared by perspective transformation). However, the bilinear mapping does not have this property when used with arbitrary quadrilateral patches as straight lines with orientations other than horizontal and vertical are mapped onto curved lines under bilinear mapping.

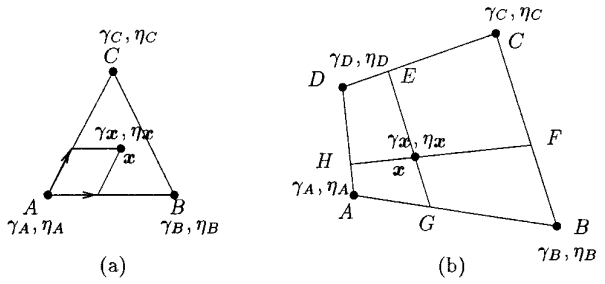


FIG. 3. Interpolation of the γ and η, of image points inside triangular in (a) and rectangular in (b) patches.

2.3. Modeling Intensity Variations

In tracking regions corresponding to nonplanar or deforming surfaces, it is important to account for variations in the intensity of pixels between the reference frame and the current frame (photometric effects). We assume that the intensity I_c of a pixel in the current frame is related to the intensity I_R of the corresponding pixel in the reference frame by

$$I_c = \gamma I_R + \eta, \tag{7}$$

where γ and η are the contrast and brightness parameters, respectively. Unlike previous approaches [13, 14], where a single set of parameters $\{\gamma, \eta\}$ is estimated for a block of pixels, we associate a set of parameters with each node, and bilinearly interpolate the contrast and brightness parameters for each pixel within a patch from those of the nodes of the patch.

For a triangular patch ABC (Fig. 3a), the bilinear interpolation for the brightness parameter of a pixel located at $\boldsymbol{x}$ within ABC is given by

$$\eta_{\boldsymbol{x}} = (1 - p_{\boldsymbol{x}} - q_{\boldsymbol{x}})\,\eta_A + p_{\boldsymbol{x}}\eta_B + q_{\boldsymbol{x}}\eta_C, \tag{8}$$

where η_A, η_B, and η_C denote the brightness parameters at the nodes A, B, and C, respectively, and the coefficients $p_{\boldsymbol{x}}$ and $q_{\boldsymbol{x}}$ are such that

$$\boldsymbol{x} - \boldsymbol{x}_A = p_{\boldsymbol{x}}(\boldsymbol{x}_B - \boldsymbol{x}_A) + q_{\boldsymbol{x}}(\boldsymbol{x}_C - \boldsymbol{x}_A), \tag{9}$$

where $\boldsymbol{x}_A$, $\boldsymbol{x}_B$, and $\boldsymbol{x}_C$ denote the locations of the nodes A, B, and C, respectively.

For a quadrilateral patch $ABCD$ (Fig. 3b), the bilinear interpolation for he brightness parameter of a pixel located at $\boldsymbol{x}$ within $ABCD$ is given by

$$\begin{aligned}\eta_{\boldsymbol{x}} = {} & (1 - p_{\boldsymbol{x}})(1 - q_{\boldsymbol{x}})\,\eta_A + p_{\boldsymbol{x}}(1 - q_{\boldsymbol{x}})\,\eta_B \\ & + p_{\boldsymbol{x}} q_{\boldsymbol{x}}\eta_C + (1 - p_{\boldsymbol{x}}) q_{\boldsymbol{x}}\eta_D,\end{aligned} \tag{10}$$

where η_A, η_B, η_C, and η_D denote the brightness parameters at the nodes A, B, C, and D, respectively, and the coefficients $p_{\boldsymbol{x}}$ and $q_{\boldsymbol{x}}$ are obtained as

$$\begin{aligned} p_{\boldsymbol{x}} &= \frac{\|\boldsymbol{x}_G - \boldsymbol{x}_A\|}{\|\boldsymbol{x}_B - \boldsymbol{x}_A\|} = \frac{\|\boldsymbol{x}_E - \boldsymbol{x}_D\|}{\|\boldsymbol{x}_C - \boldsymbol{x}_D\|}, \\ p_{\boldsymbol{x}} &= \frac{\|\boldsymbol{x}_H - \boldsymbol{x}_A\|}{\|\boldsymbol{x}_D - \boldsymbol{x}_A\|} = \frac{\|\boldsymbol{x}_F - \boldsymbol{x}_B\|}{\|\boldsymbol{x}_C - \boldsymbol{x}_B\|}, \end{aligned} \tag{11}$$

where $\|\cdot\|$ denotes vector magnitude, and $\boldsymbol{x}_A, \ldots, \boldsymbol{x}_G$ denote the locations of the points $A, \ldots, G$, respectively. The contrast parameter $\gamma_{\boldsymbol{x}}$ at each pixel location $\boldsymbol{x}$ can be computed by replacing $\eta_{\boldsymbol{x}}$ with $\gamma_{\boldsymbol{x}}$ in (8) and (10).

2.4. The Mild Deformation Assumption

In the following, we assume that the ROI is not occluded by another object throughout the sequence. However, we allow for mild elastic deformations of the object, defined by the following conditions:

(i) There is no self-occlusion, i.e., occlusion due to local motion within the ROI.

(ii) For every patch in the current mesh there is a corresponding patch in the reference mesh, i.e., the ROI moves in such a way that each patch stays entirely within the displaced reference polygon in each frame of the image sequence.

(iii) The partitioning of the reference polygon (with L corners) into $L - 2$ triangles is preserved throughout the sequence. We elaborate more on this condition in Section 3.1.2.

An example of an object undergoing mild deformations is the flag shown in the "Flag sequence" in Section 5.2.

3. MESH TRACKING

In the following, we provide a detailed discussion of major components of the proposed method, mesh propagation and hierarchical mesh refinement, accounting for intensity variations in both steps.

3.1. Mesh Propagation

Mesh propagation is a two-step procedure for predicting the locations of the nodes of the mesh in the current frame from those in the previous frame. This prediction provides an initial estimate for the subsequent process of refining the locations of the nodes. We group the nodes of the reference mesh into three different classes: (i) nodes that are inside the polygon, "inner nodes"; (ii) nodes that are on the boundaries of the polygon, "boundary nodes"; and (iii) nodes that are on the vertices of the polygon, "corner nodes." In the first step, we predict the locations of the

corner nodes by estimating their displacement from the previous frame to the current frame. In the second step, we predict the locations of the "inner" and "boundary" nodes by partitioning the reference polygon into the smallest number of triangles and using affine transformations between the corresponding triangles in the current and previous polygons. The mild deformation assumption, introduced in the previous section, facilitates this two-step procedure by guaranteeing the preservation of triangular partitioning and the mesh structure throughout the image sequence.

3.1.1. *Corner Nodes*

The parameters of the spatial transformations $\boldsymbol{T}$ (introduced in Section 2.1), at each corner point can be found by minimizing the MSE criterion

$$E(\boldsymbol{T}) = \sum_{\boldsymbol{Tx}\in S} [I_c(\boldsymbol{x}) - \gamma I_R(\boldsymbol{Tx}) - \eta]^2, \tag{12}$$

where $I_R(\cdot)$ and $I_c(\cdot)$ denote the intensity distribution in the reference and current frames (see Fig. 1), respectively, and the parameters γ and η are used to model multiplicative and additive intensity variations, respectively. Note that the MSE criterion is computed within a local region S about each node point as depicted in Fig. 1. Minimization of (12), in general, cannot be performed analytically and requires a numerical solution for $\boldsymbol{T}$. The numerical solution involves the computation of (12) for a set of candidate mappings $\boldsymbol{T}$ and then picking the one that results in the minimum MSE. In (12), $\boldsymbol{Tx}$ can fall on an inter-pixel location with non-integer coordinates. In such cases, the image intensity is determined using spatial bilinear interpolation of neighboring pixel values. For each candidate mapping, we find the optimal values of γ and η using the closed-form expressions given in [13].

There is a direct and an indirect approach to choosing the candidate mappings in (12). In the direct approach, one assigns a search space to each parameter of the motion model, and samples this search space at a rate determined by the computational limitations. Each combination of model parameters results in a distinct candidate mapping. In the indirect approach, on the other hand, one picks a number of search points (equal to half the number of free parameters in the mapping) within the region S, and associates a displacement space with each search point. The displacement space for each search point is then sampled at a rate again determined by the available computational resources. The search points can be, for example, the vertices of the region S. For each combination of search point displacements, a candidate mapping $\boldsymbol{T}$ can be computed by solving a set of linear equations. When the motion model is other than translational, the two approaches differ in their selection of candidate mappings.

In this paper, we propose a new indirect method that has significantly less computational requirements than [18], when the motion around a corner point can be described by simpler models than perspective or bilinear mappings and/or has a large translational component but only mild higher-order components such as rotation, scale, shear, etc. The generalized block-matching motion estimation (GBMME) method [18] always uses an 8-parameter mapping with four-point correspondences whenever the motion around a corner cannot be described by a translation. Our method handles the translation and higher-order components of the motion at a corner separately, which provides computational savings and greater robustness. The proposed method can be summarized as follows:

Let $2n$ denote the number of free parameters in the selected mapping $\boldsymbol{T}$ and $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n$ denote the coordinates of the search points within S about a corner in the reference frame. For example, we have $n = 1$ for translational model, $n = 2$ for the translation-rotation-zoom model, and so on. We select $\boldsymbol{r}_1$ as the corner point itself. The remaining search points may be associated with the vertices of the region S. Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ denote the initial locations that correspond to $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n$ in the current frame; which are set to the best locations of the search points corresponding to $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n$ in the previous frame. Thus, if the previous frame is the reference frame, we set

$$\boldsymbol{q}_k = \boldsymbol{r}_k, k = 1, \ldots, n. \tag{13}$$

Suppose $\mathcal{S}_t = 2^{h_t} \times 2^{h_t}$ and $\mathcal{S}_r = 2^{h_r} \times 2^{h_r}$ denote the search windows for the translational and higher-order components of the motion of the corner, respectively, where h_r and h_t are integers such that $h_r \le h_t$. We employ a multi-step logarithmic search with the initial step sizes δ_t and δ_r given by

$$\delta_t = 2^{h_t - 2} \text{ and } \delta_r = 2^{h_r - 2} \tag{14}$$

which ensure that the logarithmic search can cover the entire search window. For each combination of the displacement vectors $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n$ defined as

$$\begin{aligned} &\boldsymbol{d}_1 = \delta_t \begin{bmatrix} i \\ j \end{bmatrix}, i, j = -1, 0, 1, \text{ and } \boldsymbol{d}_k = \delta_r \begin{bmatrix} i \\ j \end{bmatrix}, \\ &i, j = -1, 0, 1, k = 2, \ldots, n, \end{aligned} \tag{15}$$

we have candidate displacements $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ given by

$$\begin{aligned} \boldsymbol{p}_1 &= \boldsymbol{q}_1 + \boldsymbol{d}_1, \\ \boldsymbol{p}_k &= \boldsymbol{q}_k + \boldsymbol{d}_1 + \boldsymbol{d}_k, \quad k = 2, \ldots, n. \end{aligned} \tag{16}$$

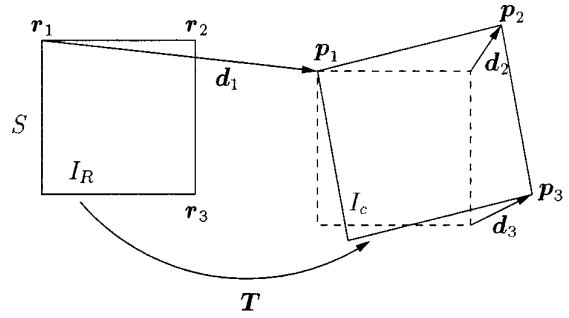


FIG. 4. Finding a candidate affine mapping $\boldsymbol{T}$ for determining the motion at a corner. The triplets $(\boldsymbol{r}_1\boldsymbol{r}_2\boldsymbol{r}_3)$ and $(\boldsymbol{d}_1\boldsymbol{d}_2\boldsymbol{d}_3)$ determine $(\boldsymbol{p}_1\boldsymbol{p}_2\boldsymbol{p}_3)$, which in turn is used to determine the affine mapping $\boldsymbol{T}$.

Observe that each search point is displaced by the same translational component $\boldsymbol{d}_1$; and only search points $2, \ldots, n$ are perturbed by the increments $\boldsymbol{d}_k$, $k = 2, \ldots, n$ in order to center higher-order motion (e.g., rotation, scale, etc.) about the corner point $\boldsymbol{r}_1$. Next, for each combination of the candidate displacements, we solve for $\boldsymbol{T}$ from the following set of equations

$$\boldsymbol{r}_i = \boldsymbol{T}\boldsymbol{p}_i \text{ for all } i = 1, \ldots, n. \tag{17}$$

An example showing the relationship between $\boldsymbol{T}$ and $\boldsymbol{r}_i$, $\boldsymbol{p}_i$, $\boldsymbol{d}_i$, $i = 1, \ldots, n$ for $n = 3$ (i.e., affine motion case) is given in Fig. 4. Once all candidate transformations $\boldsymbol{T}$ are computed, we select $\boldsymbol{T}^*$ that minimizes (12) at the present step of the logarithmic search, and update

$$\boldsymbol{r}_k = \boldsymbol{T}^*\boldsymbol{q}_k, k = 1, \ldots, n \text{ and } \delta_t = \frac{\delta_t}{2} \tag{18}$$

for the next step. If $\delta_t < \delta_r$, we also let

$$\delta_r = \frac{\delta_r}{2}. \tag{19}$$

The procedure (15)–(19) is repeated until we reach $\delta_t = \delta_r = \alpha$, where α specifies the desired accuracy of the logarithmic search. The transformation $\boldsymbol{T}^*$ obtained in the last step of the logarithmic search is the estimated transformation for the present corner.

The estimated transformations, at all corner points, determine the warping of the reference polygon (by means of corner-correspondence vectors computed from these mappings) between the reference and current frames. The total number of transformations $\boldsymbol{T}$ tested by the proposed method at each corner point is given by $\mathcal{N}^n + (h_t - \log_2 \alpha)(\mathcal{N}^{n-1})$, where $\mathcal{N}$ denotes the number or tested positions for the corner point at each accuracy level. We only tested the 8 nearest neighboring position to the current position of the corner in our experiments which is the $\mathcal{N} = 9$ case. However, only in $\mathcal{N}^{n-1} + (h_r - \log_2 \alpha)(\mathcal{N}^{n-1} - 1)$ many of these tests, we need to solve (17), because the remaining transformations differ only in the parameter $\boldsymbol{d}_1$. It should be noted that if the corner displacements result in a current polygon that does not fit the mild deformation model, i.e., violating the preservation of partitioning and mesh structure assumptions, due to possible errors in displacement estimation, corner tracking should be repeated by a different set of search parameters.

3.1.2. *Inner and Boundary Nodes*

We start by partitioning the reference polygon into the smallest number of triangles whose vertices coincide with its corners. That is, a polygon with L corners is partitioned into $L - 2$ triangles. We assume that there is at least one triangular partitioning of the reference polygon that is preserved (can be tracked) from frame to frame throughout the sequence (see the mild deformation assumption). The concept of triangular partitioning and its preservation is illustrated in Fig. 5, where Fig. 5a shows a reference polygon and its partitioning into 5 triangles. In Fig. 5b, the corner nodes of the reference polygon are displaced to $\{a', b', c', d', e', f'\}$, forming the current polygon. In this case, the current polygon allows for the same triangular partitioning as the reference polygon. In other words, the same partitioning results in triangles that are within the current polygon, corresponding to their counterparts in the previous polygon. Figure 5c on the other hand shows a case where the current polygon does not allow the same partitioning as the reference polygon.

In order to reduce the likelihood of the occurrence of a case as in Fig. 5c, we employ the following algorithm to partition the reference polygon. Let

$$\{\Delta_1, \ldots, \Delta_{L-2}\} \tag{20}$$

represent a triangular partitioning of the reference polygon, where Δ_i, $i = 1, \ldots, L - 2$, denote the triangles in the partitioning. For every possible triangular partitioning, we find the triangle Δ for which

$$\frac{\text{area of } \Delta}{\text{length of the longest side of } \Delta} \tag{21}$$

is minimum. This is the minimum value of half the distance between any corner of the polygon and the sides of the triangles in the partitioning. We then pick the partitioning for which this minimum value is maximum.

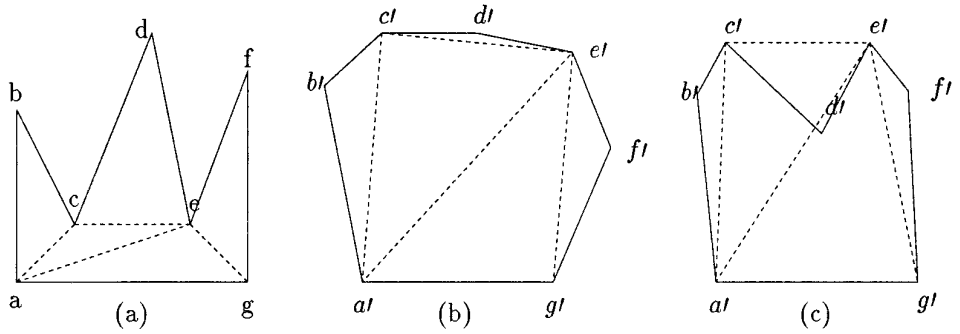


FIG. 5. A reference polygon with corner nodes $\{a, b, c, d, e, f, g\}$ is shown in (a). The nodes are displaced to $\{a', b', c', d', e', f'\}$, forming the polygon shown in (b). In this case, the current polygon allows for the same triangular partitioning as the reference polygon. The current polygon in (c) does not allow the same partitioning as the reference polygon.

Suppose the triangles given by

$$\Delta_i = (\boldsymbol{s}_{m_i}, \boldsymbol{s}_{n_i}, \boldsymbol{s}_{k_i}),$$
$$m_i, n_i, k_i = 1, \ldots, L, i = 1, \ldots, L-2, \tag{22}$$

where $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_L$ denote the corners of the reference polygon, make up the triangular partitioning obtained as a result of the above partitioning algorithm. We let $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_L$ denote the corners of the previous polygon and let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_L$ denote the estimated locations of the corners of the current polygon. We compute the affine transformations $\boldsymbol{A}_i$, $i = 1, \ldots, L-2$, between the corresponding triangles in the previous and current polygons,

$$\boldsymbol{A}_i: (\boldsymbol{t}_{m_i}, \boldsymbol{t}_{n_i}, \boldsymbol{t}_{k_i}) \rightarrow (\boldsymbol{u}_{m_i}, \boldsymbol{u}_{n_i}, \boldsymbol{u}_{k_i}), i = 1, \ldots, L-2. \tag{23}$$

Then, the location $\boldsymbol{n}_c$ of a node of the current mesh is predicted from the location $\boldsymbol{n}_p$ of the corresponding node of the previous mesh by using the appropriate affine transformation:

$$\boldsymbol{n}_c = \boldsymbol{A}_\ell \boldsymbol{n}_p,$$
$$1 \le \ell \le L-2 \text{ such that } \boldsymbol{n}_p \in (\boldsymbol{t}_{m_\ell}, \boldsymbol{t}_{n_\ell}, \boldsymbol{t}_{k_\ell}). \tag{24}$$

In Fig. 6, we show a partitioning of previous and current polygons with $L = 4$ into two triangles and the corresponding affine transformations $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. We note that nodes $\boldsymbol{a}$ and $\boldsymbol{c}$ are mapped by $\boldsymbol{A}_2$ and node $\boldsymbol{b}$ is mapped by $\boldsymbol{A}_1$. Since affine transformations preserve ratios, if a node is on the boundary of two triangles, it will be mapped to the same location by both transformations.

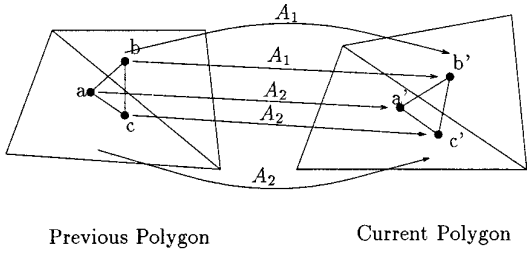


FIG. 6. Corresponding polygons are divided into triangles and the affine mapping parameters among each pair is calculated. These mapping parameters are used to map the nodes of the previous mesh.

3.2. Hierarchical Mesh Refinement

Mesh refinement refers to updating the node locations that are obtained as a result of propagating the reference mesh into the current frame. The criterion for updating the node locations is to have the image intensity distribution within any two corresponding patches in the current and reference meshes match under an affine transformation (or bilinear transformation in the case of a quadrilateral mesh) and an intensity variation model. Because an ideal solution to this problem in general may not exist, we seek a solution that minimizes the mean-squared matching error E defined as

$$E = \frac{1}{N} \sum_{\ell=1}^{L} \sum_{\boldsymbol{x} \in S_\ell} \text{wfd}^2(\boldsymbol{T}_\ell, \boldsymbol{x}), \tag{25}$$

where $S_1, \ldots, S_L$ represent the individual patches in the current mesh, N denotes the total number of pixels in the current mesh, and

$$\text{wfd}(\boldsymbol{T}, \boldsymbol{x}) = I_c(\boldsymbol{x}) - \gamma_{\boldsymbol{x}} I_R(\boldsymbol{T}\boldsymbol{x}) - \eta_{\boldsymbol{x}}, \tag{26}$$

denotes the warped frame difference between the reference and current frames after the correction for intensity variations. We note that the error function (25) has as many variables as twice the number of nodes in the current mesh, i.e., one variable for each coordinate of the position of a node. Thus, a numerical procedure that tests all possible combinations of node positions (within an allowed dis-

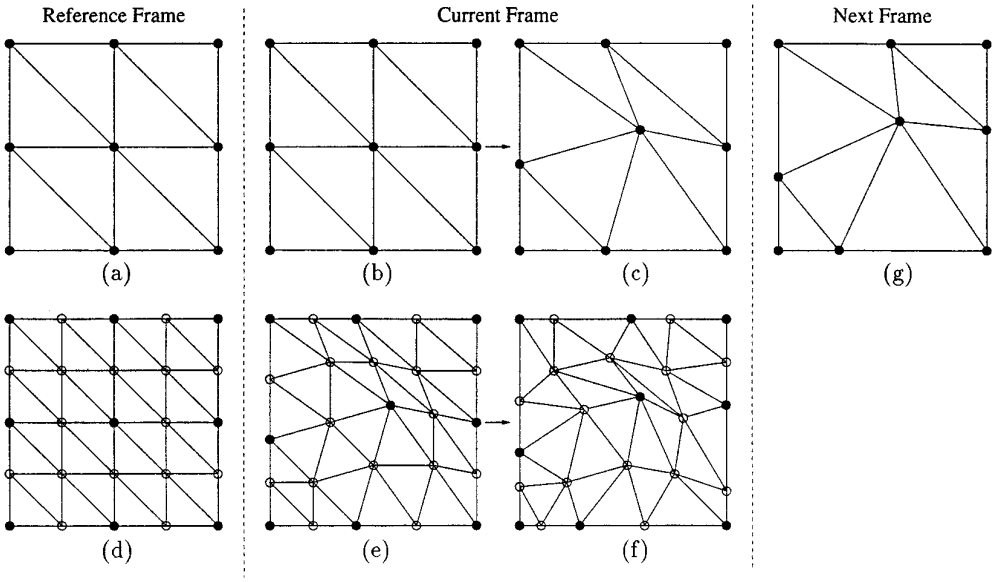


FIG. 7. Depiction of the hierarchical mesh refinement process in the case of two levels. The initial mesh (b) in the current frame is obtained by propagating the reference mesh (a). The initial mesh (b) is refined into the mesh shown in (c) in the first level of the hierarchy. In the second level of the hierarchy, the node densities for the mesh in (a) and (c) are increased by introducing additional nodes half way between the lines connecting the existing nodes, resulting in the meshes shown in (d) and (e). The mesh in (e) is then refined into the mesh in (f). In order to form the initial mesh (g) in the next frame, the first-level nodes of the mesh in (f) are propagated to the next frame.

placement range for each node) to find the optimum combination (in the sense of minimizing (25)) may be unrealistic in practice. Instead, we obtain a suboptimum solution by successively visiting each node of the mesh and moving a node to a new position (within an allowed displacement range) so as to minimize the mean-squared matching error locally. In this case, the minimization is only with respect to two variables—coordinates of the position of the node being visited—and the matching error is computed only over the union of patches whose geometrical definitions are affected by the movement of the node (i.e., those patches that have the node as one of their corners). This process of updating (or *refining*) the location of a node is treated in detail in Section 3.2.2. We also elaborate on how to obtain the aforementioned union of patches for a node in Section 3.2.1. We call this union region the *cost polygon* of the node, as the position of the node is refined based on the matching error computed over this region only. Furthermore, we limit the movement of the node to a subset of the cost polygon called the *search space* of the node in order to satisfy certain convexity constraints. We elaborate on how to obtain the search space of a node also in Section 3.2.1.

We employ an iterative procedure to refine the locations of the nodes. In the first iteration, all nodes of the mesh are visited in the order of inner, boundary, and corner nodes. In the subsequent iterations, while a corner node is visited at every iteration, an inner or a boundary node is visited only if its cost polygon has changed since it has been last visited. Iterations are stopped when there remains no node whose location needs to be refined (i.e., when all nodes assumed their locally optimum positions), or when a predetermined maximum number of iterations is reached. The current mesh is then said to be refined.

Once the current mesh is refined, each one of its patches (and each patch of the reference mesh) is divided into four smaller patches, and the above procedure is repeated to refine the new mesh with a denser set of nodes. Hence the hierarchical nature of the proposed algorithm. The process of increasing the density of nodes and refining the resulting mesh is repeated until the desired density of nodes is achieved.

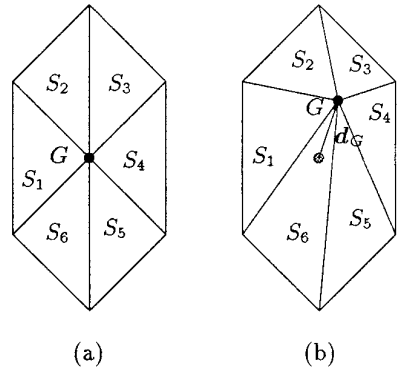


FIG. 8. Triangular patches are deformed by the displacement of grid G.

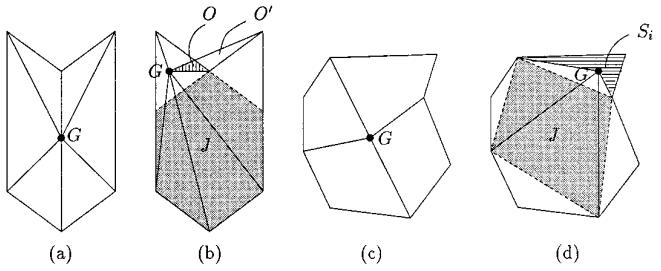


FIG. 9. Problems associated with nonconvex cost polygons in case of triangular (a) and rectangular (c) meshes. In case of a triangular mesh, depending on the location of the node during the search process, two patches can overlap, resulting in an ambiguity for the overlap region denoted by O, or a region can be mapped out of the cost polygon resulting in the outside region denoted O', as shown in (b). In case of a rectangular mesh, a convex patch inside the cost polygon can become a nonconvex polygon, denoted as S_i in (d), during the search process. The shaded regions in (b) and (d) indicate the allowed region J for node G.

The final patch size used for transfiguration should be small enough to capture the local motion accurately. However, as the patch size gets smaller, the number of data points in the patch decreases. Thus, in order to lower the likelihood of an erroneous match, the search space (in terms of the displacement of the node locations) of a patch must be reduced as the patch size gets smaller. However, a reduced search space requires a good initial estimate for the patch. In the present method, this is indeed provided by the proposed hierarchical approach to refining the node locations (hence the patches).

An example as to how the proposed hierarchical mesh refinement process works is given in Fig. 7, where, without loss of generality, a two-level hierarchy is depicted for a triangular mesh. For the purpose of clarity, we assumed that the corner nodes are fixed. Let the reference mesh be as given in Fig. 7a, where the nodes are indicated with solid circles. Suppose that the current frame is the first frame following the reference frame. Then, the propagated mesh (Fig. 7b) in the current frame will be identical to the reference mesh (fixed-corner assumption). Call this mesh the 1st-level mesh and the nodes of the mesh the 1st-level nodes. Suppose that the refined 1st-level mesh is as given in Fig. 7c. The initial estimate of the mesh in the 2nd-level of the hierarchy is formed by introducing new inner and boundary nodes half way between each pair of connected nodes in the refined 1st-level mesh. The resulting mesh (Fig. 7e) is called the 2nd-level mesh and the newly introduced nodes are called the 2nd-level nodes. The new nodes are indicated by white circles in Fig. 7e. The density of nodes in the reference mesh is likewise increased in the 2nd-level of the hierarchy (Fig. 7d). The locations of both the 1st-level and the 2nd-level nodes are then refined to obtain the refined mesh in the 2nd-level of the hierarchy. Suppose that the resulting mesh is given in Fig. 7f. When the hierarchy has more than two levels, the above process is simply repeated for the higher levels of the hierarchy, as well, to obtain the final refined mesh in the current frame. The initial mesh at the next frame, shown in Fig. 7g, is formed by propagating *only the 1st-level nodes* of the final refined mesh (in this case, the refined 2nd-level mesh) to the next frame. Thus, the mesh refinement process in the next frame also starts with a lowest resolution mesh as was the case for the current frame.

At this point, we would like to note that the iterative refinement procedure (without any hierarchy) is originally suggested by Nakaya *et al.* in [7], where they only have considered inner nodes. We point out the differences in the refinement of corner and boundary nodes in Section 3.2.2. In the following, we also propose a logarithmic search process to refine the location of a node, which is significantly faster than the exhaustive search process of [7]. Furthermore, we provide a recipe for finding the search space for inner, boundary, and corner nodes in Section 3.2.1. Incorporation of intensity variations (Section 3.2.3) in the refinement of a node location is another novelty of the present paper. In addition to improving the performance of the mesh refinement process, the proposed space-varying model for intensity variations provides a more realistic rendition of image intensities for the purposes of transfiguration.

3.2.1. *The Cost Polygon and Search Space for a Node*

Before the location of a given node can be refined, a cost polygon and search space associated with the given node are to be determined. The definition of the cost polygon and search space for a node, and the process for obtaining them are presented here for the inner, boundary, and corner nodes.

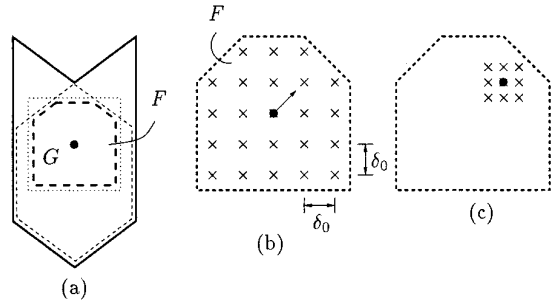


FIG. 10. (a) The cost polygon and the search space. Because the cost polygon is nonconvex, it is trimmed. A two-level logarithmic search is depicted in (b) and (c), where an enlarged version of the search space is shown. The candidate locations for G in the first level of the logarithmic search is shown in (b). Candidate locations in the second level, eight locations around the best location found in the first level, are shown in (c).

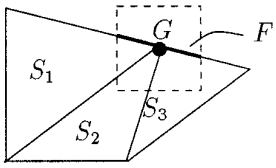


FIG. 11. Hexagonal search for a boundary grid using logarithmic method.

A. Inner Nodes. Let G denote an arbitrary inner node in a triangular or rectangular mesh, let K denote the number of patches that are connected to G, and let $S_1, \ldots, S_K$ denote these patches. A search for the refined position of G is conducted within the region $S = \cup_{i=1}^{K} S_k$, which is referred to as the "cost polygon" of G. In the case of a triangular mesh, we in general have $K = 6$ and S enclosed by a hexagon as shown in Fig. 8. Thus, the name hexagonal search as originally was proposed by Nakaya *et al.* [7].

The inner node G may not be allowed to visit every location in S. This fact is demonstrated in Fig. 9 for cost polygons in a triangular (Fig. 9a) and a rectangular (Fig. 9c) mesh. The location of G in Fig. 9b results in an overlapping region O and causes a portion O' of a patch to lie outside the cost polygon. The location of G in Fig. 9d, on the other hand, results in a non-convex patch within the cost polygon. It can be shown that the collection of locations that G is allowed to visit within S forms a convex subset of S that can be easily found given S. This subset will be called as the allowed region. The allowed region for G is indicated by J in Figs. 9b and 9d (note that J is not necessarily the largest convex region in S). It is always true that when all the patches in S are quadrilaterals, the outline of the allowed region itself is given by a quadrilateral whose corners are positioned at the nodes of the mesh that are linked to G as in Figs. 9b and 9d.

Suppose that the current mesh is an nth level mesh, and that node G is an mth level node (we have $m \leq n$). In addition to the allowed region found for node G in the current mesh, we also find the allowed regions for node G considering only $n - 1$st and lower level nodes of the current mesh, then $n - 2$nd and lower level nodes, and so on, until mth and lower level nodes. In this paper, we restrict the displacement of G to the intersection of all the allowed regions for G and a square window of width w_i centered at G, as shown in Fig. 10a. In the following, this intersection region will be called as the search space for G and denoted by F.

B. Boundary Nodes. The cost polygon S for boundary node G is the union of patches that are connected to G. Thus, in Fig. 11, we have $S = S_1 \cup S_2 \cup S_3$. However, G may not visit every location in S, and in fact it is required to remain on the boundary of the polygon. The actual region that G is allowed to visit is given as the subset of the straight line segment joining the two neighboring nodes of G such that any location on this subset does not result in overlapping regions or nonconvex patches in the current and any lower-resolution levels of the hierarchy as previously explained for the inner nodes. This subset is intersected with a square window of width, say, w_b centered at G to form the search space for G as shown in Fig. 11.

C. Corner Nodes. The cost polygon S for corner G is defined as the union of those members of the triangular partitioning that have a vertex coincident with corner G. Such triangles are warped as corner G is moved during the search process. Figure 12 shows two different cost polygons for corner G, for different triangular partitioning. In Fig. 12a, $S = \Delta_1$, and in Fig. 12b, $S = \Delta_1 \cup \Delta_2$. Let K denote the number of triangles from the triangular partitioning that make up the cost polygon S of corner G and let $\Delta_{G,k}, k = 1, \ldots, K$ denote these triangles. Invoking the mild deformation assumption, the triangular partitioning is preserved, and for every triangle $\Delta_{G,k}$ within S there is a corresponding triangle in the reference polygon, denoted as $\Delta_{R,G,k}$. The search space F for a corner node G is defined as a square region with side length w_c given as some power of 2, and centered at the corner node G (see Fig. 12).

3.2.2. *Refining the Location of a Node*

A. Inner Nodes. Let $S_{R,1}, \ldots, S_{R,K}$ denote the patches in the reference frame that correspond to the $S_1, \ldots, S_K$ that are connected to G. The definitions of $S_1, \ldots, S_K$ vary with the displacement $\boldsymbol{d}_G$ of G as shown in Fig. 8. In order to find the optimum displacement $\boldsymbol{d}_G^*$ of G, we employ the MSE criterion defined as

$$E(\boldsymbol{d}_G) = \frac{1}{\sum_{k=1}^{K} N_k} \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in S_k(\boldsymbol{d}_G)} \mathrm{wfd}^2(\boldsymbol{T}_k, \boldsymbol{x}), \qquad (27)$$

where N_k denotes the number of pixels within the patch S_k, and $\boldsymbol{T}_k : S_k(\boldsymbol{d}_G) \rightarrow S_{R,k}$ is an affine mapping (3) if $S_{T,k}$

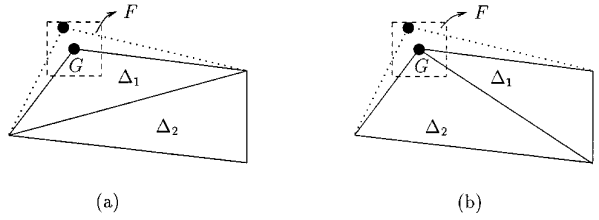


FIG. 12. A corner node moves inside a specified window F, of size w_c. The figure on the right shows how the reference half triangle changes when the corner is moved.

is a triangle or a bilinear mapping (5) if $S_{T,k}$ is a rectangle. The evaluation of the contrast and brightness terms that appear in $\text{wfd}(\boldsymbol{T}_k, \boldsymbol{x})$ is discussed in Section 3.2.3. The optimum displacement $\boldsymbol{d}_G^*$ is given by the following

$$\boldsymbol{d}_G^* = \begin{cases} \text{minimizer of } E(\boldsymbol{d}_G) & \text{in minimum } E(\boldsymbol{d}_G) < \rho E(\boldsymbol{0}), \\ \boldsymbol{0} & \text{otherwise} \end{cases}, \tag{28}$$

where $\boldsymbol{d}_G = \boldsymbol{0}$ implies zero displacement for G, and $0 < \rho \leq 1$. The value of ρ is usually set to 1 except when the ROI contains uniform regions with size comparable to that of the patches. In that case, the value of ρ is set to less than 1. The actual value of ρ is determined experimentally. We note that the smaller the value of ρ, the larger must be the reduction in the MSE in order for the location of G to be changed. The parameter ρ is thus introduced as a measure against false displacements of G in uniform image intensity regions because of the presence of noise.

Search within the search space for the optimum location of G can be performed exhaustively as suggested in [7]. In this paper, however, we employ a logarithmic search [19] process that results in significant computational savings especially when the position of G needs to be estimated at a fractional pixel accuracy. The logarithmic search process employed in this paper starts with a predetermined initial step size $\delta = \delta_0$ selected to be some power of two. Given δ_0 and the accuracy α of the search (i.e., $\alpha = 1/2$ for half pixel accuracy, $\alpha = 1/4$ for quarter pixel accuracy, and so on) the logarithmic search process will have $M = 1 + \log_2(\delta_0/\alpha)$ levels. The candidate locations for G in the first level of the logarithmic search are the samples of the search space F with a step size of δ_0 in both directions as demonstrated in Fig. 10b. At each subsequent level, the step size is halved and the search is performed only at eight neighboring points around the best location found in the previous level as in Fig. 10c. This process is repeated until the step size becomes equal to α at the Mth level.

We note that, for an $N \times N$ search space, there are $((1/\alpha^2)N^2)$ candidates for G in an exhaustive search. In the proposed logarithmic search process, however, the number of candidates for G is given by $((1 + 2\lfloor N/2\delta_0 \rfloor)^2 + 8 \log_2(\delta_0/\alpha))$, where $\lfloor x \rfloor$ denotes the largest integer not greater than x. Thus, for example, for $N = 9$, $\delta_0 = 2$, and $\alpha = 1/8$, the proposed logarithmic search process is nearly 83 times faster than the exhaustive search process of [7].

B. Boundary Nodes. The refinement process for a boundary node is similar to that of an inner node. The candidate locations for boundary node G in the first level of the logarithmic search are the samples of the search space F with a step size of δ_0 along the boundary of the object polygon as demonstrated in Fig. 11. At each subsequent level, the step size is halved and the search is performed only at two neighboring points on either side of the best location found in the previous level as in Fig. 11.

C. Corner Nodes. The refinement of the location of a corner node G is performed using a logarithmic search process similar to the one employed for inner and boundary nodes. The initial step size δ_0 is set to $w_c/4$, and then halved at each subsequent level until it reaches α at the final level $M = 1 + \log_2(\delta_0/\alpha)$, where α denotes the predetermined accuracy. For each location of G the definition of each $\Delta_{G,k}$ changes. Let $\Delta_{G,k}(\boldsymbol{d}_G)$ denote the definition of $\Delta_{G,k}$ corresponding to the displacement of G by $\boldsymbol{d}_G$. If we define affine transformations $\boldsymbol{A}_k : \Delta_{G,k} \rightarrow \Delta_{G,k}(\boldsymbol{d}_G)$, $k = 1, \ldots, K$, then the location of $\boldsymbol{n}_c$ of a node of the current mesh inside one of $\Delta_{G,1} \cdots \Delta_{G,K}$ is changed to its new location $\boldsymbol{n}_c(\boldsymbol{d}_G)$ by using the appropriate affine transformation:

$$\boldsymbol{n}_c(\boldsymbol{d}_G) = \boldsymbol{A}_k \boldsymbol{n}_c, 1 \leq k \leq K \text{ such that } \boldsymbol{n}_c \in \Delta_{G,k}. \tag{29}$$

Let

$$\Delta_G(\boldsymbol{d}_G) = \cup_{k=1}^{K} \Delta_{G,k}(\boldsymbol{d}_G); \tag{30}$$

then, the error criterion to find the optimum displacement $\boldsymbol{d}_G^*$ of G, is defined as

$$E(\boldsymbol{d}_G) = \frac{1}{\sum_{S \cap \Delta_G(\boldsymbol{d}_G) \neq \emptyset} N_S} \sum_{S \cap \Delta_G(\boldsymbol{d}_G) \neq \emptyset} \sum_{\boldsymbol{x} \in S} \text{wfd}^2(\boldsymbol{T}, \boldsymbol{x}), \tag{31}$$

where S denotes a patch in the current mesh, N_S denotes the number of pixels in S, and $T : S \rightarrow S_R$ denotes the spatial transformation from S to the corresponding patch S_R in the reference frame. At the next level of the logarithmic search process, we consider the set of nine locations at and around the updated location. The above procedure is repeated for a total of M levels to find the optimum location of the corner G. Note that if, during the search, the corner moves to a location where the triangular partitioning, contrary to the mild deformation model, is no longer preserved, then the search is stopped and a new search using a new set of parameters (e.g., smaller w_c) is started.

3.2.3. *Accounting for Intensity Variations*

We account for intensity variations using the relationship $I_c = \gamma I_R + \eta$, where γ and η are the contrast and brightness parameters, as discussed in Section 2.3. The method that we suggest for finding the γ and η values for each node modifies the error criterion (27) used in the mesh refinement process that we have discussed in the previous sec-

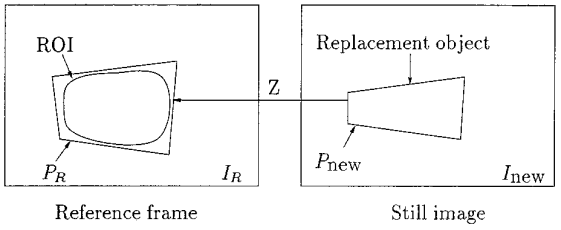


FIG. 13. Obtaining $I_{R,\text{new}}$ for transfiguration. The texture within the replacement object, is mapped into ROI, using the spatial transformation $\boldsymbol{Z}$, which is computed from the point correspondences of P_{new} and P_R.

tion. This method applies to inner and boundary nodes only. In this method, γ and η values are allowed to continuously vary for each pixel location inside the cost polygon during the refinement process. These values are obtained by bilinearly interpolating the parameters assigned to associated nodes, as described in Eqs. (8) and (10). Affine or bilinear interpolation method for triangular or quadrilateral patches, respectively, have been used for the luminance parameters in order to simplify the problem and get two linear equations in two unknowns. Also for triangular patches the interpolation can be performed very fast using scan line algorithm which only adds little to the overall computation. Let G denote the node of interest, and S_k, $k = 1, 2, \ldots, K$ denote the patches that are in the cost polygon. The error criterion is as defined in (27) for inner and boundary nodes and (31) for corner nodes.

For the kth patch, $\gamma_{\boldsymbol{x}}$ and $\eta_{\boldsymbol{x}}$, is interpolated from the parameters associated with the corners of the patch. For a triangular patch, whose corners are denoted by G, G_k, and G_{k+1}, if we denote the unknown contrast and brightness parameters for node G by (γ_G, η_G), and the parameters of other nodes of the patch by $(\gamma_{G_k}, \eta_{G_k})$ and $(\gamma_{G_{k+1}}, \eta_{G_{k+1}})$, $\gamma_{\boldsymbol{x}}$ and $\eta_{\boldsymbol{x}}$ are calculated as:

$$\begin{aligned}\gamma_{\boldsymbol{x}} &= ((1 - p_{\boldsymbol{x}} - q_{\boldsymbol{x}})\gamma_G + p_{\boldsymbol{x}}\gamma_{G_k} + q_{\boldsymbol{x}}\gamma_{G_{k+1}}) \\ \eta_{\boldsymbol{x}} &= ((1 - p_{\boldsymbol{x}} - q_{\boldsymbol{x}})\eta_G + p_{\boldsymbol{x}}\eta_{G_k} + q_{\boldsymbol{x}}\eta_{G_{k+1}}),\end{aligned} \tag{32}$$

where $p_{\boldsymbol{x}}$ and $q_{\boldsymbol{x}}$ are as defined in (9). Here, the contrast and brightness parameters $(\gamma_{G_k}, \eta_{G_k})$ and $(\gamma_{G_{k+1}}, \eta_{G_{k+1}})$ are assumed to be known *a priori*. This may not be the case at the very first iteration of the refinement process. We therefore initialize the nodes either with the default values of $\gamma = 1.0$ and $c = 0.0$, or with the final values of the mesh determined at the previous frame, when available. For a quadrilateral patch, bilinear interpolation of parameters associated with the corner nodes is used to determine $\gamma_{\boldsymbol{x}}$ and $\eta_{\boldsymbol{x}}$.

Because introduction of two new unknowns increases the search space by two dimensions in optimizing (27), we use a pseudo-optimum approach that is suggested in [13]. We first find the optimal γ_G and η_G by setting $\partial E/\partial_{\gamma G} = 0$ and $\partial E/\partial_{\eta G} = 0$. This yields two linear equations in γ_G and η_G that can be solved to find optimal γ_G^* and η_G^* as a function of $\boldsymbol{T}_k$. Then the optimal γ_G^* and η_G^* are substituted into the error criterion and the resulting function is minimized for $\boldsymbol{T}_G$ using the mesh refinement procedure. It can easily be verified that the optimal parameters have the following form:

$$\begin{aligned}\gamma_G &= \frac{1}{\mathscr{D}}(\mathscr{M}_1\mathscr{M}_3 - \mathscr{M}_2\mathscr{M}_4) \\ \eta_G &= \frac{1}{\mathscr{D}}(\mathscr{M}_2\mathscr{M}_5 - \mathscr{M}_1\mathscr{M}_4).\end{aligned} \tag{33}$$

Here I and $\tilde{I}$ denote $I_c(\boldsymbol{x})$ and $I_R(\boldsymbol{T}_k(\boldsymbol{x}))$, respectively, and

$$\begin{aligned}m_{\boldsymbol{x}} &= 1 - p_{\boldsymbol{x}} - q_{\boldsymbol{x}}, \\ r_{\boldsymbol{x}} &= I - (p_{\boldsymbol{x}}\gamma_{G_k} + q_{\boldsymbol{x}}\gamma_{G_{k+1}})\tilde{I} - (p_{\boldsymbol{x}}\eta_{G_k} + q_{\boldsymbol{x}}\eta_{G_{k+1}}) \\ \mathscr{M}_1 &= \sum_{k=1}^{K}\sum_{\boldsymbol{x}\in S_k} r_{\boldsymbol{x}}m_x\tilde{I}, \quad \mathscr{M}_2 = \sum_{k=1}^{K}\sum_{\boldsymbol{x}\in S_k} r_{\boldsymbol{x}}m_x, \\ \mathscr{M}_3 &= \sum_{k=1}^{K}\sum_{\boldsymbol{x}\in S_k} m_{\boldsymbol{x}}^2, \quad \mathscr{M}_4 = \sum_{k=1}^{K}\sum_{\boldsymbol{x}\in S_k} m_{\boldsymbol{x}}^2\tilde{I}, \\ \mathscr{M}_5 &= \sum_{k=1}^{K}\sum_{\boldsymbol{x}\in S_k} (m_{\boldsymbol{x}}\tilde{I})^2, \quad \mathscr{D} = \mathscr{M}_5\mathscr{M}_3 - \mathscr{M}_4^2.\end{aligned} \tag{34}$$

For corner nodes, a weighted average of γ and η values, calculated on the patches sharing the corner node, are used. The weights are determined by the respective area of each patch connected to the corner.

4. TRANSFIGURATION

The 2-D mesh model describes the motion and intensity variations of an ROI throughout the given image sequence.

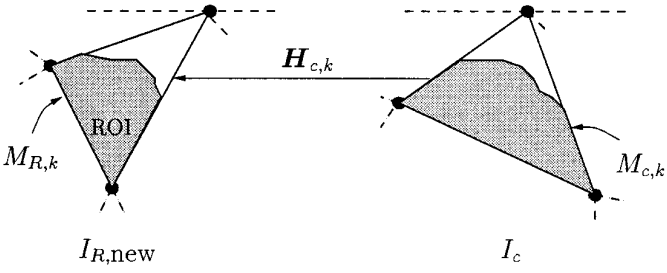


FIG. 14. The kth patch of M_c, $M_{c,k}$, and the corresponding patch in M_R, $M_{R,k}$, are depicted in the figure on the right and left, respectively. The intensity within the intersection of ROI, and the patch $M_{R,k}$ is texture-mapped into I_c using $\boldsymbol{H}_{c,k}$ to obtain the intensity of the current frame $I_{c,\text{new}}$ inside the patch $M_{c,k}$.

FIG. 15. (a) Reference frame, 9th frame, of Quark; (b) reference mesh superimposed on the reference frame; (c) original 1st frame; (d) original 22nd frame; (e) tracked mesh on the 1st frame; (f) tracked mesh on the 22nd frame; (g) self-transfiguration of the 1st frame; and (h) self-transfiguration of the 22nd frame.

FIG. 17. (a) Replacement polygon overlaid on the replacement object (b) 1st (c) 9th, and (d) 22nd frames of the generated transfiguration sequence.

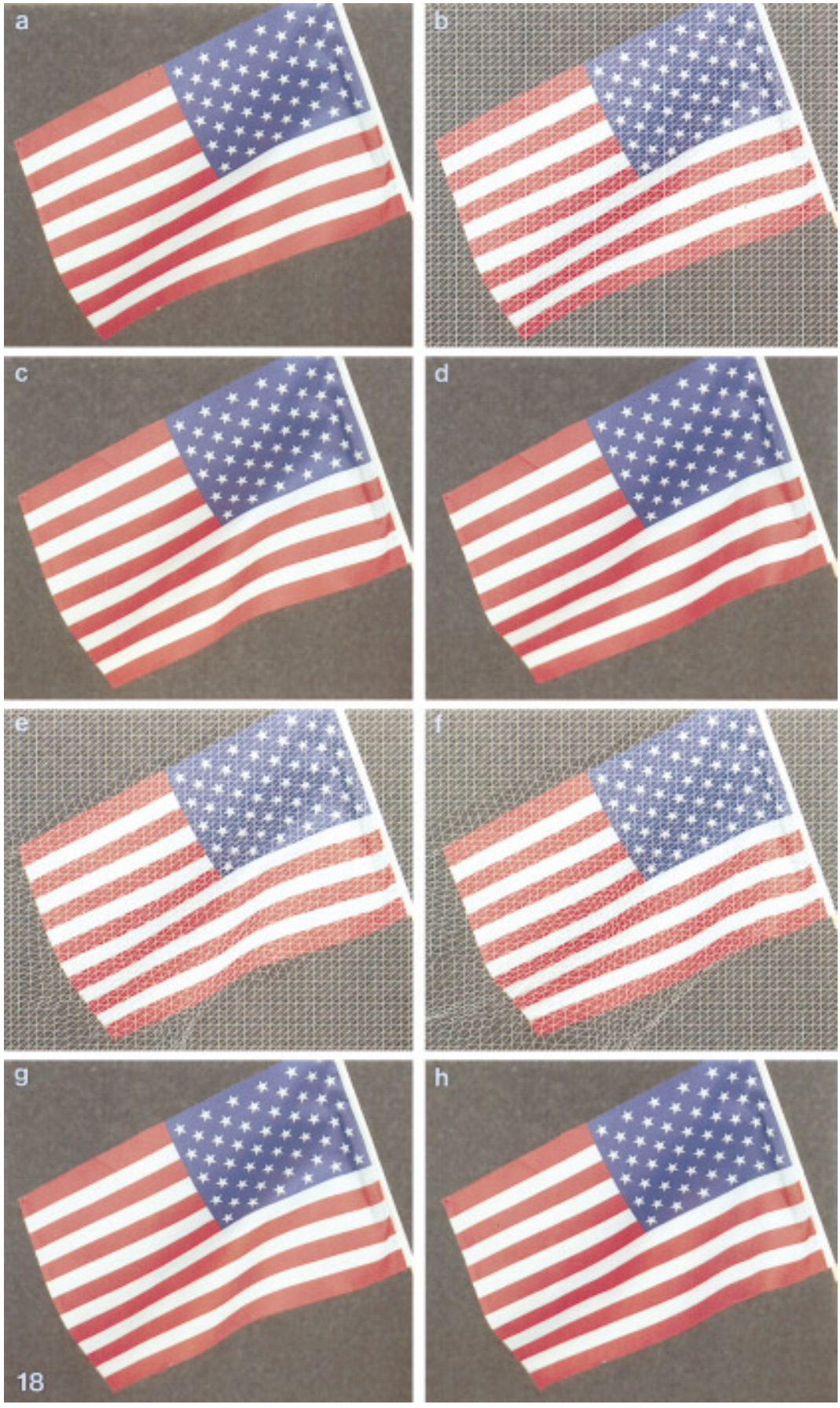
The ROI can be transfigured, i.e., replaced by a new object to obtain an image sequence containing the new object going through the same motion and intensity variations as the original region.

Suppose that the replacement object is enclosed by the replacement polygon P_{new} in the replacement image I_{new}, and P_{new} has the same number of sides as the reference polygon, P_R. First, a spatial transformation from the replacement polygon onto the reference polygon is determined. We will denote this transformation as $\boldsymbol{Z}:\mathrm{P}_{\text{new}} \rightarrow P_R$. In order to determine the reference frame $I_{\text{R,new}}$ of the new sequence, the replacement object is warped into I_R as

$$I_{R,\text{new}}(\boldsymbol{x}) = \begin{cases} I_{\text{new}}(\boldsymbol{Z}^{-1}\boldsymbol{x}), & \text{for } \boldsymbol{x} \in \text{ROI} \\ I_R(\boldsymbol{x}), & \text{elsewhere.} \end{cases} \tag{35}$$

In certain cases, the object to be replaced has a deforming surface, such as a waving flag (see Section 5). An ROI,

FIG. 18. (a) Reference frame, 1st frame, of the Flag sequence; (b) reference mesh superimposed on the reference frame; (c) original 5th frame; (d) original 10th frame; (e) tracked mesh superimposed on the 5th frame; (f) tracked mesh on the 10th frame; (g) self-transfiguration of the 5th frame; and (h) self-transfiguration of the 10th frame.


a
b
c
d
e
f
g
h
18

therefore, may have deforming boundaries and it may not be possible to exactly enclose it with a polygon. In such cases, a polygon that is large enough to contain the ROI is defined as P_R. In other words, ROI becomes a proper subregion of P_R. This is depicted in Fig. 13.

The transfiguration of the ROI in a current frame is based on the transformation between the current mesh M_c and the reference mesh M_R and is performed on a patch-by-patch basis. The transformation between the kth patch in M_c and the corresponding mesh in M_R is directly computed from node correspondences using the affine or the bilinear model, depending on the patch geometry. We refer to the transformation between the kth patch of M_c and the corresponding patch in M_R as $\boldsymbol{H}_{c,k}: M_{c,k} \rightarrow M_{R,k}$ (see Fig. 14). The intensity at position $\boldsymbol{x}$ within the kth patch in the transfigured current frame $I_{\text{c,new}}$ is thus given by

$$I_{\text{c,new}}(\boldsymbol{x}) = \begin{cases} \gamma_{\boldsymbol{x}} I_{R,\text{new}}(\boldsymbol{H}_{c,k}\boldsymbol{x}) + \eta_{\boldsymbol{x}}, & \text{for } \boldsymbol{H}_{c,k}\boldsymbol{x} \in \text{ROI} \\ I_C(\boldsymbol{x}), & \text{otherwise.} \end{cases} \tag{36}$$

where $\boldsymbol{x} \in M_{c,k}$, and the contrast and brightness parameters, $\gamma_{\boldsymbol{x}}$ and $\eta_{\boldsymbol{x}}$, are determined as described in Section 3.2.3. Once all patches within the current frame are transfigured, the process is then repeated for the remaining frames in the image sequence.

In certain augmented reality applications, a synthetic object, e.g., a text, is placed inside an ROI so that the placement object appears as if it is a part of the ROI (see Section 5 for an example). This is achieved by transfiguring only the part of the ROI where the synthetic object should appear. We illustrate this in Section 5 where we add synthetic text on a waving flag.

It is also possible to perform so-called *self-transfiguration*, which refers to rendering the ROI throughout the image sequence from the reference ROI rather than a new replacement object. In this case, the transfiguration can be performed by directly using the transformations determined during the mesh refinement process. Self-transfiguration facilitates motion and intensity-compensated prediction of an ROI, and can be used in content-based video compression. In the next section, we use the accuracy of self-transfiguration as a means for evaluating the accuracy of the proposed tracking method.

5. RESULTS

We have applied the proposed tracking algorithm to image sequences containing real-life motion. We report the results obtained in case of tracking (i) a rigid object with curved surface; and (ii) a mildly deforming object, in Sections 5.1 and 5.2, respectively. The first experiment is an example of replacing an object with a new object, where the sign ''QUARK, 3784'' is replaced by ''DAY 51, 2465'', while the second one is an example for augmentation reality, where the flag is augmented by the text ''USA''. In both experiments, we have used a triangular reference mesh. Mesh propagation, mesh refinement, and estimation of contrast and brightness parameters are all performed in the luminance domain. During transfiguration, the same set of estimated contrast and brightness parameters are applied to each color channel. For corner tracking, we used the affine motion model for all corners of the reference polygon, which was found to be satisfactory in all practical cases we dealt with.

In order to speed up the numerical minimization process, (12) and (27) are computed over a subsampled version of S after blurring the reference and current frames. The blurring is employed to incorporate into the error computations the non-sampled pixel values and to *reduce the effects of noise* that may be present on the images. The amount of blurring is chosen to be the same for both the reference and current frames and to be at least as big as the subsampling factor. Because every patch is divided into four smaller patches in the hierarchical mesh refinement process, error calculations are done using larger subsampling factors (coarser resolution) at the earlier levels of the hierarchy than at the higher levels (finer resolution). The mesh refinement process is further speeded up by choosing larger values for the initial step size, δ, and search accuracy, α, at the earlier levels of the hierarchy than at the higher levels.

We have evaluated the tracking results on the basis of the root-mean-square error (RMSE) between the ROI at a certain frame and its rendering from the ROI at the reference frame, using the estimated motion, contrast, and brightness parameters (i.e., self-transfiguration), as well the visual quality of the transfigured sequence. For ROI at the current frame, the RMSE is defined as

$$\text{RMSE} = \left\{ \frac{1}{\sum_{k=1}^{K} \sum_{\boldsymbol{x} \in S_k} \delta(\boldsymbol{T}_k \boldsymbol{x})} \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in S_k} \text{wfd}^2(\boldsymbol{T}_k, \boldsymbol{x})\, \delta(\boldsymbol{T}_k \boldsymbol{x}) \right\}^{1/2}, \tag{37}$$

where K denotes the number of patches on the reference mesh, and $\boldsymbol{x}$ is an image sample point within S_k, the kth patch of the mesh. The $\delta(\boldsymbol{y})$ function is defined as follows:

$$\delta(\boldsymbol{y}) = 1 \text{ if } \boldsymbol{y} \in \text{ROI, and 0 otherwise.} \tag{38}$$

5.1. Rigid Object with Curved Surface

We have used the sequence, called "Quark," in testing the performance of our tracking algorithm in case of a highly curved, rigid surface. We have compared nonhierarchical versus hierarchical mesh refinement, and also observed the positive effects of accounting for intensity variations. Transfiguration results are also furnished. The Quark sequence is digitized from a video clip obtained by taping a highly curved, stationary, rigid poster board, containing the text "Quark 3784" against a textured background, by moving a Hi-8 mm consumer camcorder around the poster board. We have used 22 frames of the clip, where even fields are spatially interpolated to frame resolution (480 lines and 640 pixels) using directional spatial interpolation as discussed in [20]. The 9th frame of Quark, shown in Fig. 15a, is used as the reference frame. The 1st and the 22nd frames are shown in Figs. 15c and 15d, respectively. A rectangle of size 480×200, surrounding the text, is chosen as the reference polygon (and ROI). The reference polygon, and the reference mesh used in the single-level mesh refinement experiment, are shown in Fig. 15b. The patch width and height are 30 and 25 pixels, respectively.

We model the intensity variations throughout the sequence by means of only the brightness parameter η; that is we set $\gamma \equiv 1$. The corners of the reference polygon are numbered in a clockwise order such that the 1st corner is at the top right, and the 4th corner is the bottom right. Cost polygons for corners are chosen within the reference polygon as discussed in Section 3.1.1. Cost polygons for the 1st, 2nd, and 4th corners are of size 48×48. A 48 lines by 72 pixels rectangle is used as a cost polygon for the 3rd corner. The subpixel accuracy is $\alpha = 1/8$ for each corner, and the search parameters h_t and h_r are set to 6 and 4, respectively, corresponding to search windows, $\mathcal{S}_t$ and $\mathcal{S}_r$, of size 64×64 for translation, and 16×16 for rotation, zoom, and affine motion. For each cost polygon, the subsampling factor used during the search is equal to three, and the blur kernel used in blurring the frames is 7×7.

The parameters used in single- and three-level hierarchical mesh refinement are shown in Table 1, where parameters α, δ_0, w_i, w_b, and w_c are as defined in Section 3.2. The value of the parameter ρ is set to unity as the ROI did not contain uniform regions whose size is comparable to or larger than the size of the patches. For hierarchical refinement, the width and the height of the patches at the first (coarsest) level are 120 and 100, respectively. The size of the patches of the reference mesh created at the 3rd (finest) level of the hierarchy is equal to that of the reference mesh used in the single-level refinement. The average RMSE, defined as the average of RMSEs computed for all frames other than the reference frame, is monotonically decreasing with respect to the number of iterations and has saturated after six iterations of mesh refinement in the single-level case. We have compared the average RMSE and CPU performances of the single-level refinement with those for the three-level hierarchical refinement where the total number of iterations throughout the hierarchy is limited to six. In this and other experiments, we have chosen the limits for the number of iterations per hierarchy level from a set of possible combinations where the number of iterations monotonically increases from coarse to fine resolution levels, and the iteration limit for the coarsest level is less than or equal to $\lfloor \mathcal{N}/\lambda \rfloor$, where $\mathcal{N}$ is the total number of iterations and λ is the number of levels.

TABLE 1
Parameters Used in Mesh Refinement for Quark

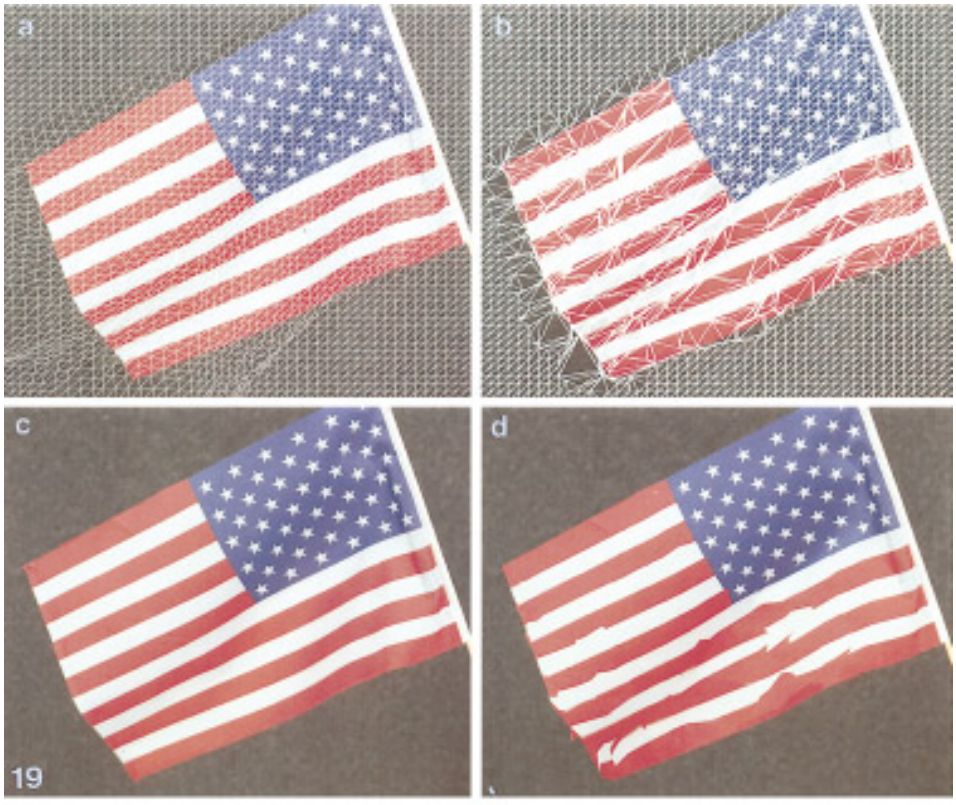
	Level	α	δ_0	w_i	w_b	w_c	Subsampling factor	Blur size
Single level	1	1/8	2	9	9	9	2	3
3-Level hier.	1	1	4	11	11	5	8	11
	2	1	2	7	7	5	4	7
	3	1/8	2	3	3	9	2	3

TABLE 2
Results of Experiments for Quark

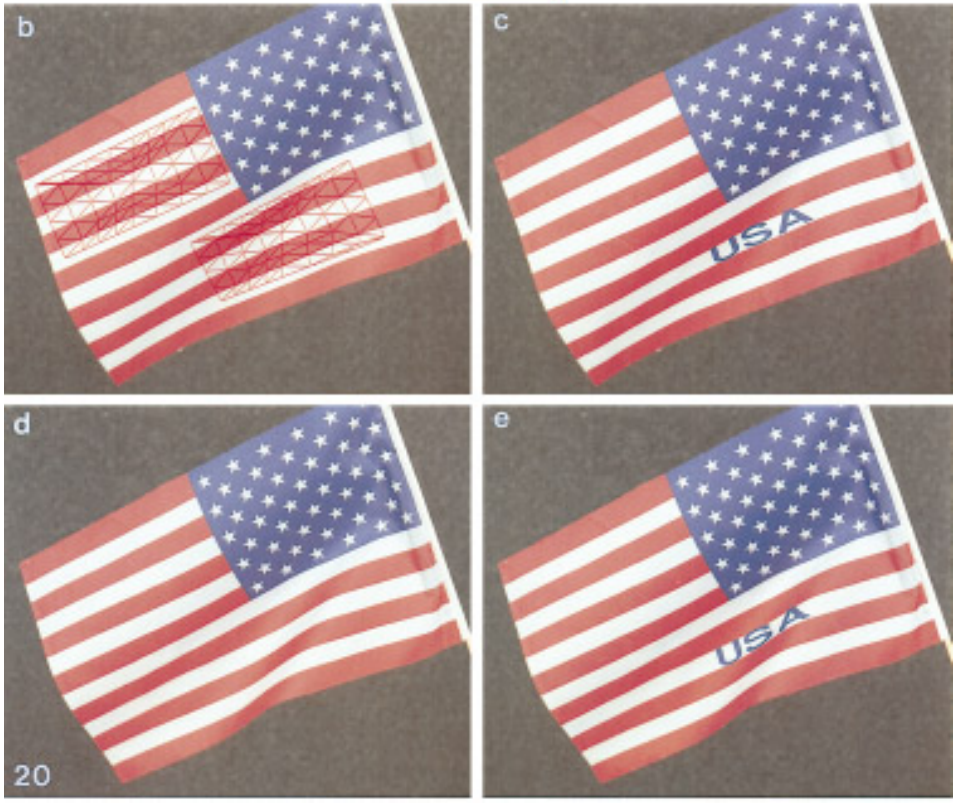
Experiment	Average RMSE	CPU time (s)
SINGLE-6	5.82458	2345
HIER-114	5.83108	1435
HIER-123	5.84563	1107
HIER-222	5.89567	764

The average RMSE and CPU performances of the single-level and three different implementations of three-level hierarchical refinement are given in Table 2, where SINGLE-6 denotes the single-level refinement and HIER-*xyz* denotes a three-level hierarchical refinement where number of iterations are limited to x, y, and z, in the first (coarsest), second, and third (finest) resolutions. Although the average RMSE performances are quite similar, the CPU advantage of hierarchical refinement is evident from the results. The insignificant RMSE advantage of the hierarchical refinement is because of the lack of local deformations in the actual surface in 3-D. The single-level results, at the end of six iterations of refinement, are shown in Fig. 15. The tracked meshes at the 1st and the 22nd frames, and the self-transfiguration of these frames are shown in Figs. 15e, 15f, 15g, and 15h, respectively. The visual difference between the single level and hierarchical results is insignificant.

To show the positive effect of taking the brightness term into account, we have repeated the SINGLE-6 experiment with $\gamma \equiv 1$ and $\eta \equiv 0$. Comparison of RMSE performance


a
b
c
d
19


a
USA


b
c
USA
d
e
USA
20

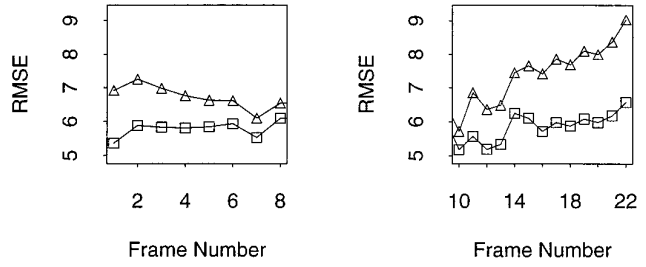


FIG. 16. The improvement achieved by taking the intensity variations into account in Quark: RMSE vs frame number, where ''triangles'' and ''squares'' denote the cases where intensity variations are and are not taken into account, respectively.

versus frame number is presented in Fig. 16 for frames 1 through 8 and 10 through 22.

Finally, the text ''Quark 3784'' is transfigured and replaced by ''Day 51, 2465'' using the results of tracking obtained in the case of SINGLE-6 experiment. The replacement polygon on the new object is depicted in Fig. 17a. The transfigured sequence was obtained as explained in Section 4. The results of the transfiguration of 1st, 9th, and 22nd frames are presented in Figs. 17b, 17c, and 17d, respectively.

5.2. Mildly Deforming Object

We demonstrate the performance of the proposed approach in case of mild deformations. The test sequence is called ''Real Flag'' and is obtained by digitizing a part of a clip obtained by shooting a waving flag with a stationary motion picture camera. Real Flag contains 10 frames with 720 lines and 870 pixels for each line. The first frame, shown in Fig. 18a, is taken as the reference frame. Because the flag has deformable boundaries, the reference polygon is chosen as the entire frame. The ROI in the reference polygon is taken as the region containing the flag; i.e., the outline of the flag coincides with the outline of the ROI. The outline is identified by an interactive contour drawing tool.

The patches of the reference mesh has horizontal and vertical dimensions of 18.125 and 15 pixels, respectively, in the case of single-level refinement. For hierarchical refinement, the width and the height of the patches at the first level is set to 290 and 240 pixels, respectively. The mesh created at the 5th level of the hierarchy is equal in patch size to the single-level mesh. The values of the parameters used in mesh refinement for single level and five-level hierarchical refinement are shown in Table 3. We again model the intensity variations throughout the sequence by means of only the brightness parameter η; that is we set $\gamma \equiv 1$. The value of the parameter ρ is set to 0.975 as the ROI in this case did contain uniform regions (stripes of the flag) with size comparable to that of the patches.

TABLE 3
Parameters Used in Mesh Refinement for Flag

	Level	α	δ_0	w_i	w_b	Subsampling factor	Blur size
Single level	1	1/8	8	31	31	2	3
5-Level hier.	1	1	16	49	49	10	17
	2	1	8	24	24	8	13
	3	1	4	12	12	6	9
	4	1	2	5	5	4	5
	5	1/8	1	3	3	2	3

The average RMSE performance, computed over the ROI according to (37), in single-level refinement is monotonically decreasing and has saturated after five iteractions. In the five-level hierarchical refinement, we limited the number of iterations to a total of 11. In particular, we have limited the number of iterations to 2–2–2–2–3 at first, second, third, fourth, and the fifth levels of the hierarchy, respectively, adding up to a maximum of 11 iterations. The average RMSE results are summarized in Table 4, for single- versus five-level hierarchical refinement, and with and without intensity variation modeling. The CPU time for five-level hierarchical refinement (not shown in the table) with a total number of eleven iterations is significantly less (by a factor of 3) than that of single-level refinement with five iterations. The RMSE performance of the hierarchical implementation is clearly superior to that of the single-level implementation. Mesh tracking and self-transfiguration results for the five-level hierarchical refinement are shown in Fig. 18. Self-transfiguration (over the ROI) of the 5th frame using the hierarchical refinement is compared to that obtained using the single-level refinement in Fig. 19.

Next, we superimpose the text ''USA'' on the flag via partial transfiguration to implement ''augmented reality.'' The goal is to have the text follow the deformation and intensity variations on the flag so that the text appears as if it were actually on the flag. The replacement text USA, centered within a rectangle, is shown in Fig. 20a. We refer

FIG. 19. Comparison of self transfiguration of the 5th frame of the Flag sequence using hierarchical and single-level mesh refinement: (a) hierarchically refined mesh, (b) single-level refined mesh, (c) self-transfiguration using (a), and (d) self-transfiguration using (b).

FIG. 20. Results of transfiguring Flag: (a) The replacement text ''USA''; (b) the quadrilateral Q_1, at the lower right, and Q_2, at the upper right, shown with corresponding mesh structures within; (c) the new reference frame; (d) original 4th frame; and (e) transfigured version of the 4th frame.

TABLE 4
Summary of Results for Flag

	Single level (SINGLE-5) with $\eta = 0$	Single level (SINGLE-5) with varying η	5-Level (HIER-22223) with $\eta = 0$	5-Level (HIER-22223) with varying η
Average RMSE	25.5649	14.0429	17.62	8.94364

to this rectangle as Q_0. We particularly want to place the text on a highly deforming region of the flag, namely the lower part of the flag in this case. Such a region, shown in Fig. 20b, is a quadrilateral and will be referred to as Q_1. If the test sequence had contained a frame where an almost flat (nondeforming) version of this region were available, we would choose that frame as the reference frame and we would use a spatial transformation from the rectangle containing USA into Q_1 and texture map only the USA text into Q_1. In this case, however, a flat version of this region is not available in the test sequence. To alleviate the problem, we identified a region in the upper left part of the flag that is almost flat in the 1st frame (the reference frame). We refer to this quadrilateral region as Q_2 (see Fig. 20b). We want to place the USA text onto the white stripe inside Q_1. Therefore, we centered Q_2 on a white stripe and aligned the upper and lower boundaries of Q_2 with the stripe boundaries in the neighborhood. We have then estimated the parameters of the global perspective transformation between the quadrilateral Q_2 and the rectangle Q_0, denoted as $\mathbf{Z}: Q_2 \rightarrow Q_0$. We next estimate the local transformations that map the reference mesh within Q_1 onto Q_2, using three-level hierarchical mesh refinement with parameters shown in Table 5. Using these transformations, and the transformation $\mathbf{Z}$, we place the replacement text USA in Q_1, as shown in Fig. 20c. The flag in the 1st frame, containing the new text, is taken as the reference ROI and the new sequence is obtained by performing transfiguration of the flag using the tracking parameters estimated earlier. The results are shown in Figs. 20d and 20e, where the 4th frame of the original sequence and its transfigured version are shown, respectively.

TABLE 5
Parameters Used in Mesh Refinement in Transfiguration of Flag

Level	α	δ_0	w_i	w_b	Subsampling factor	Blur size
1	1	4	15	15	2	7
2	1	4	9	9	2	3
3	1/8	2	5	5	1	3

6. CONCLUSION

This paper introduced a new algorithm for object tracking and synthetic transfiguration, where curved surfaces, mildly deforming objects, and frame-to-frame contrast and brightness variations are allowed. Variations in both the motion (because of curved surfaces and mild deformations) and the intensity within the ROI have been tracked by means of a hierarchical 2-D mesh model. Spatial transformations-based motion estimation is shown to be effective in the case of mildly deforming objects and highly curved surfaces. Geometric constraints have been introduced to ensure the connectivity of the 2-D deformable mesh model. A new interpolation-based model is proposed to compensate for intensity variations over long sequences, which is shown to be very effective. Incorporating the intensity variations has resulted in superior tracking of the object. The computational load of the tracking algorithm has been significantly reduced by the proposed logarithmic search.

The proposed hierarchical approach to local motion estimation is efficient because larger areas with insignificant variations in the governing motion model can be tracked in less time using larger patches. Those regions containing significant local variations are tracked using smaller patches in a more efficient manner because the estimates obtained by the larger patches are propagated as initial estimates to the smaller patches.

The proposed mesh-based motion tracking method can also be used in object-based video coding. It is well-known that mesh-based motion compensation overcomes the blocking artifacts observed in international coding standards at very low bit-rates. In this case, the boundaries of the moving objects, the displacement vectors, and the contrast and brightness parameters at the nodes of the reference mesh need to be sent. The receiver can then reconstruct subsequent frames given this information and the reference frame. Work is under progress to detect and handle self-occlusion with the ROI, and motion estimation and tracking in the presence of more than mild deformations, which are essential in object-based video coding.

REFERENCES

1. F. Leymarie and M. Levine, Tracking deformable objects in the plane using an active contour model, *IEEE Trans. Pattern Anal. Mach. Intell.* **15,** June 1993, 617–634.
2. K. Fujimura, N. Yokoya, and K. Yamamoto, Motion tracking of deformable objects by active contour models using multiscale dynamic programming, *J. Visual Commun. Image Represent.* **4,** Dec. 1993, 382–391.
3. B. Bascle *et al.*, Tracking complex primitives in an image sequence, in *Int. Conf. Pattern Recog.*, *Israel*, *Oct. 1994*, pp. 426–431.
4. M. Kass, A. Witkin, and D. Terzopoulos, Snakes: Active contour models, *Int. J. Comput. Vision* 1(4), 1988, 321–331.
5. C. Kervrann and F. Heitz, Robust tracking of stochastic deformable models in long image sequences, in *IEEE Int. Conf. Image Proc.*, *Austin*, *TX*, *Nov. 1994*.
6. G. Sullivan and R. Baker, Motion compensation for video compression using control grid interpolation, in *IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, *Toronto*, *Canada*, *May 1991*, pp. 2713–2716.
7. Y. Nakaya and H. Harashima, Motion compensation based on spatial transformations, *IEEE Trans. Circuits and Systems Video Technol.* **4,** June 1994, 339–357.
8. M. Dudon, O. Avaro, and G. Eude, Object-oriented motion estimation, in *Picture Coding Symposium*, *California*, *Sept. 1994*, pp. 284–287.
9. Y. Wang and O. Lee, Active mesh-a feature seeking and tracking image sequence representation scheme, *IEEE Trans. Image Process.* **3,** Sept. 1994, 610–624.
10. J. Nieweglowski, T. Campbell, and P. Haavisto, A novel video coding scheme based on temporal prediction using digital image warping, *IEEE Trans. Consumer Electron.* **39,** Aug. 1993, 141–150.
11. C.-L. Huang and C.-Y. Hsu, A new motion compensation method for image sequence coding using hierarchical grid interpolation, *IEEE Trans. Circuits and Systems Video Technol.* **4,** Feb. 1994, 42–52.
12. R. Szeliski and H.-Y. Shum, *Motion Estimation with Quadtree Splines*, Tech. Rep., 95/1, Digital Equipment Corp., Cambridge Research Lab, Mar. 1995.
13. C.-S. Fuh and P. Maragos, Affine models for image matching and motion detection, in *IEEE Int. Conf. Acoust., Speech, and Signal Proc., Toronto, Canada, May 1991*, pp. 2409–2412.
14. H. Li and R. Forchheimer, A transformed block-based motion compensation technique, *IEEE Trans. Commun.* **43,** Feb./Mar./Apr. 1995, 1673–1676.
15. G. Wolberg, Spatial transformations, in *Digital Image Warping*, IEEE Comput. Soc. Press, Los Alamitos, CA, 1992.
16. A. M. Tekalp, in *Digital Video Processing*, Prentice–Hall, Englewood Cliffs, NJ, 1995.
17. M. Buck and N. Diehl, Model based image sequence coding, in *Motion Analysis and Image Sequence Processing* (M. I. Sezan and R. L. Lagendijk, Eds.), Kluwer, Boston, 1993.
18. V. Seferidis and M. Ghanbari, General approach to block-matching motion estimation, *Opt. Eng.* **32,** July 1993, 1464–1474.
19. T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, Motion compensated interframe coding for video conferencing, in *Proc. of the Nat. Telecom. Conf.*, *New Orleans*, *LA*, *Nov. 29–Dec. 3*, *1981*, pp. G5.3.1–5.3.5.
20. A. J. Patti, M. I. Sezan, and A. M. Tekalp, Robust methods for high-quality stills from interlaced video in the presence of dominant motion, *IEEE Trans. Circuits and Systems Video Technol.* to appear.

Statement of ownership, management, and circulation required by the Act of October 23, 1962, Section 4369, Title 39, United States Code: of

GRAPHICAL MODELS AND IMAGE PROCESSING

Published bimonthly by Academic Press, Inc., 6277 Sea Harbor Drive, Orlando, FL 32887-4900. Number of issues published annually: 6. Editor: Dr. N. I. Badler, Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104-6389; and Dr. R. Chellappa, Center for Automation Research, University of Maryland, College Park, MD 20742.

Owned by Academic Press, Inc., 525 B Street, Suite 1900, San Diego, CA 92101-4495. Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, and other securities: None.

Paragraphs 2 and 3 include, in cases where the stockholder or security holder appears upon the books of the company as trustee or in any other fiduciary relation, the name of the person or corporation for whom such trustee is acting, also the statements in the two paragraphs show the affiant's full knowledge and belief as to the circumstances and conditions under which stockholders and security holders who do not appear upon the books of the company as trustees, hold stock and securities in a capacity other than that of a bona fide owner. Names and addresses of individuals who are stockholders of a corporation which itself is a stockholder or holder of bonds, mortgages, or other securities of the publishing corporation have been included in paragraphs 2 and 3 when the interests of such individuals are equivalent to 1 percent or more of the total amount of the stock or securities of the publishing corporation.

Total no. copies printed: average no. copies each issue during preceding 12 months: 1759; single issue nearest to filing date: 1750. Paid circulation (a) to term subscribers by mail, carrier delivery, or by other means: average no. copies each issue during preceding 12 months: 465; single issue nearest to filing date: 453. (b) Sales through agents, news dealers, or otherwise: average no. copies each issue during preceding 12 months: 754; single issue nearest to filing date: 709. Free distribution (a) by mail: average no. copies each issue during preceding 12 months: 46; single issue nearest to filing date: 46. (b) Outside the mail: average no. copies each issue during preceding 12 months: 10; single issue nearest to filing date: 10. Total no. of copies distributed: average no. copies each issue during preceding 12 months: 1275; single issue nearest to filing date: 1218. Percent paid and/or requested circulation: average percent each issue during preceding 12 months: 96%; single issue nearest to filing date: 95%.

(Signed) Janice M. Peterson, Director, Fulfillment and Special Projects