

New Kernels for Fast Mesh-Based Motion Estimation

Aria Nosratinia

Abstract—Mesh-based motion estimation—also known as *control grid interpolation* or *warping*—provides a smoother estimated intensity field compared to the traditional block-matching algorithm (BMA), resulting in most cases in a more realistic motion field and smaller estimation error. In mesh-based motion, unlike BMA, the computation of a motion vector is affected by its neighboring vectors. This interdependence necessitates a costly, iterative computation of motion vectors. The computational cost of mesh-based motion has been a main drawback of this otherwise powerful technique. We propose to use noniteratively computed motion vectors, such as BMA motion vectors, for node motions in the mesh model. However, we found that a straightforward insertion of BMA motion vectors in the deformable mesh leads to unpredictable and erratic results, and were thus motivated to carefully analyze the interaction of motion vectors and interpolation kernels in mesh models. This analysis leads to a methodology for computing optimal motion interpolation kernels for a given set of motion vectors (e.g., BMA motion vectors). We find a generalized orthogonality condition for these kernels; optimality is achieved only if the projections of vertex motions on the local intensity gradients are statistically orthogonal to mesh-based estimation errors. Experiments show that optimal kernels are often very different from the traditional bilinear kernels, and exhibit interesting variations. The new kernels benefit a variety of applications, including motion estimated interpolation, denoising, and compression.

Index Terms—Image sequence analysis, video coding.

I. INTRODUCTION

MOTION ESTIMATION and compensation through a deformable mesh (also known as *control grid interpolation* or *warping*) is one of the more advanced methods for the representation and processing of video. Compared to the better known—and widely used—block matching algorithm (BMA), warping is capable of producing a more accurate representation of motion fields, and thus is an attractive tool for many applications in video processing.

The key advantage of the mesh-based method is the ability to generate continuously varying motion fields. This is especially important because natural video sequences generally give rise to smooth or slowly varying motion fields. In particular, camera zoom, movement of the objects toward or away from the camera, and rotational motion of objects are much better described in a mesh-based model than in the traditional block-based motion model. The estimated intensity fields in these cases are smoother

than the blocky estimated fields generated by BMA, and the corresponding estimation error usually has smaller energy.¹

The earliest work on mesh-based geometrical transformation of digital images appeared in the field of remote sensing [1], where transformations were used to compensate for the distortions introduced by the imaging system. Image warping was also used to generate special effects in computer graphics [2], [3]. Applications in video signal processing quickly followed. Bruzewitz [4] and Sullivan and Baker [5] explored mesh-based systems for motion compensation and video compression. Others [6]–[8] also developed similar results. Huang and Hsu [9] introduced a hierarchical version of the motion mesh. Wang *et al.* [10] used a finite-element approach to the mesh problem, and explored its application in video coding [11]–[13]. Tekalp *et al.* [14], [15] used the mesh model for manipulation of synthetic objects as well as representations in joint natural-synthetic environments. Mesh-based methods have also been used with great success in processing medical image sequences [16]–[18].

The mesh motion model is based on a transformation that maps one set of polygons in the past frame to another set of polygons in the present frame. The pixels within the polygons are mapped from one frame to another through interpolation. To perform motion estimation, the present frame is covered by a regular tiling of polygons. Motion estimation consists of finding the vertices of the corresponding polygons in the past frame, such that the overall motion field results in the smallest possible error metric.

Fig. 1 shows an example of block-based and mesh-based interframe estimation, with rectangular tiling.² The pixel values inside each tile are estimated from the warped tile in the past frame. Fig. 1 does not show the correspondence of the individual pixels in the two frames, which is generated by interpolation (typically bilinear) between the vertex positions.

Motion meshes with triangular and rectangular tiling of the plane have attracted the most attention in the literature. Hexagonal tiling and nonuniform tiling of the plane are also possible, but introduce significant complexity, as well as problematic effects at the image boundaries. To our knowledge, these other tilings have not been actively pursued for motion models. This paper concentrates on the rectangular tiling, however, the results are general and can be directly extended to any uniform tiling.

Manuscript received September 18, 1998; revised May 19, 2000 and May 1, 2000. This work was supported in part by the National Science Foundation under Grant CCR-9985171.

The author was with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA. He is now with the Department of Electrical Engineering, University of Texas at Dallas, Richardson, TX 75083 USA.

Publisher Item Identifier S 1051-8215(01)00667-X.

¹Not all scenarios are equally well modeled by warping. For example, a continuous warping model cannot describe occlusions.

²There is another version of mesh-based motion estimation where the tiling is regular in the originating frame instead of the destination frame. This alternative form is known as forward tracking. Although our results extend to that case directly, we do not address it explicitly in this paper.

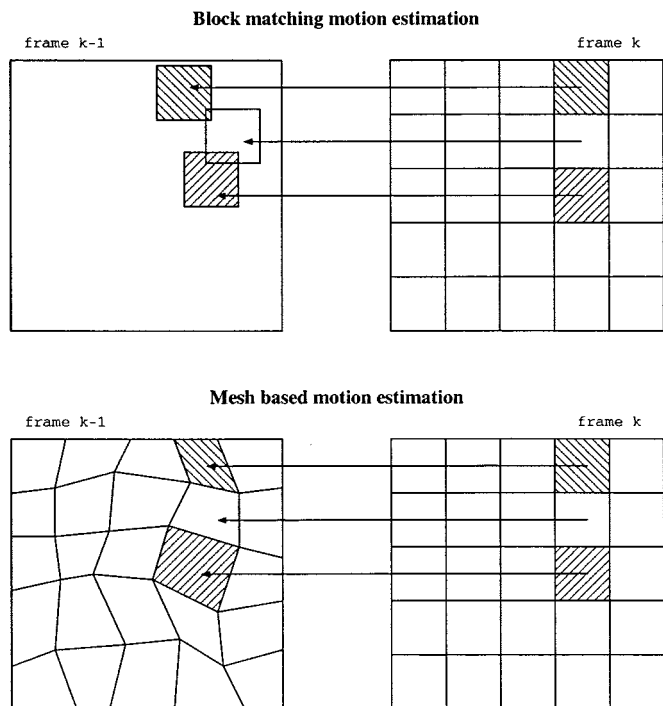


Fig. 1. Top: block-matching motion estimation. Bottom: Mesh-based motion estimation; new frame is tiled and corresponding tiles in the old frame are warped to give best match to current frame.

A. A Question of Complexity

Mesh-based motion estimation is a useful tool in describing motion fields, but is handicapped due to its computational complexity. To shed light on the mechanics of this handicap, and to search for ways of addressing it, we revisit model-based motion estimation. By “model based,” we refer to systems where pixel motions are not computed independently, but are generated through a relatively small number of parameters. This class includes both BMA and warping.

In model-based motion estimation, we wish to determine the best mapping parameters for a model that describes one frame in terms of another, such that some quality measure (often squared frame difference) is optimized. The choice of the model indicates one’s underlying beliefs regarding the properties of the motion fields, but also reflects the degree of willingness to accept complexity (conceptual or computational) in order to represent the true motion fields. BMA is one of the simplest of such models and has been widely applied. In BMA, motion estimation reduces to a simple correlation problem. The simplicity of BMA is due to the following facts: 1) motion vectors can be computed independently from one another (divide and conquer) and 2) the optimality of each vector depends only on a small subset of pixels in the frame.

In contrast, in a mesh-based model, the motion-compensated intensity estimate at each pixel depends on more than one node (see Figs. 1 and 2). Therefore, the computation of the optimal position for each node depends on the position of neighboring nodes. The inter-dependence necessitates a recursive computation of node positions, which can be costly. This effect is compounded by the fact that testing each candidate node position is expensive, because the motion of each pixel inside a polygon

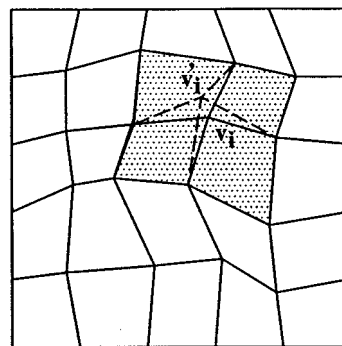


Fig. 2. Two candidates for node v_i are shown. Node position affects PSNR of the shaded area, therefore all pixels in that area are used to compute best position for v_i .

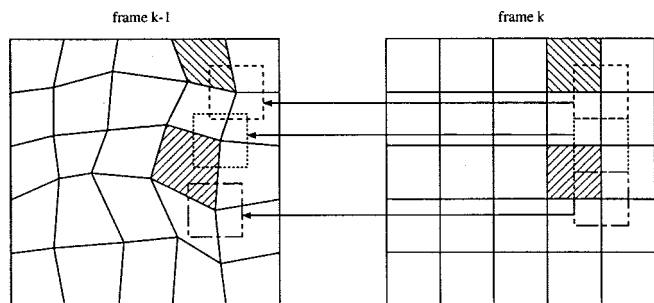


Fig. 3. Using block-matching motion vectors for node motions, without any refinement.

has to be computed via interpolation. Together, these two factors lead to a heavy computational load. In fact, the computational cost of mesh-based motion estimation is the primary drawback of this powerful technique.

While demanding a higher computational cost, mesh-based motion also offers smaller estimation error energy, and perhaps more importantly for many applications, a smooth estimated intensity field. It is then natural to ask: is it possible to achieve some of these desirable properties at a lower cost? One alternative would be to use the mesh-based motion model, but marry it to a simpler—perhaps suboptimal—motion estimation method. But is it possible to do so and yet maintain some of the advantages of mesh-based motion?

A candidate solution is to directly insert BMA motion vectors into the mesh model (see Fig. 3). Unfortunately, this direct approach, while simple and inexpensive, does not work very well. Experiments in Section V show that BMA motion vectors, when used in the traditional mesh model, generate erratic and unpredictable performance. Thankfully this is not the end of the road; significantly better results are possible if the motion interpolation of the mesh-based algorithm is matched to the statistics of the motion vectors and the video sequence. The analysis required for this matching is the subject of this paper.

We show that a careful design of motion interpolation kernels indeed makes it possible to use simplified motion estimation for mesh-based models. We call these motion interpolators *warping kernels*. We present a method to compute optimal warping kernels for any kind of node motion vectors (e.g., BMA motion vectors). Optimal warping kernels often look very different from the traditional bilinear interpolators. They perform better than

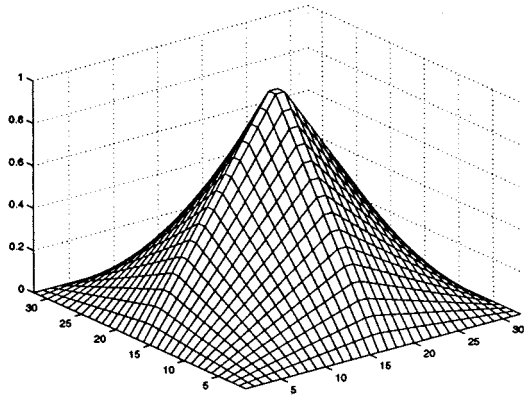


Fig. 4. Bilinear interpolation kernel, used in traditional mesh-based motion estimation.

BMA, while maintaining the motion-estimation complexity at the same level as BMA.

B. Interpolation Kernels and Intensity Fields

We use an analogy to describe some of the issues concerning motion interpolation kernels: one can think of mesh-based motion representation as a warping or stretching action with a rubber sheet. The rubber sheet has the image of one frame on it, and is stretched or warped until it matches the other frame. This warping is performed through the movement of a discrete number of control points at the vertices of the polygons (Fig. 2). The movement of these control points deforms the rubber sheet according to its elastic properties. The elasticity of the sheet is characterized by the warping kernel. The shape of these kernels will determine how the sheet will deform (motion estimation) as a function of the position of the control points (motion vectors).

Traditional warping uses a bilinear kernel (see Fig. 4), reflecting the belief that the rubber sheet should have uniform elasticity. The questions we raise here are: what is the optimal distribution of elasticity of the rubber sheet, and knowing the intensity sequence, how can we find this optimal distribution. These questions not only have an immediate impact on the computational complexity, as seen in Section I-A, they also give insight on the properties of the underlying (true) motion fields.

We will show that the answer to this question is nontrivial; that in fact the best distribution of elasticity for our fictitious rubber sheet is highly nonuniform. The best rubber sheet in most cases is rigid close to the control points (vertices of polygons) and more elastic away from them.

We also show that optimality for the warping kernel requires a generalized orthogonality condition. The warping kernel is optimal only if the motion estimated error is statistically orthogonal to the projection of vertex motions on the local intensity gradients.

C. Organization

The organization of this paper is as follows. Section II introduces the terminology used for mesh-based motion estimation, and presents optimality criteria for warping kernels. In some applications, the number of coefficients associated with the optimal kernel may be an impediment, and a constrained version

of the kernel with fewer parameters is desirable. Section III presents results for a constrained optimal kernel, with only one or two independent parameters. Section IV discusses computational aspects of the new method, comparing with the traditional mesh-based systems. Section V presents experimental and numerical results, and Section VI has concluding discussions.

II. OPTIMAL WARPING KERNELS

Let $I_k(s)$ represent the intensity of frame k at pixel s , and I_{k-1} the intensity of the past frame³. Let $v(s)$ be the (interpolated) motion at pixel s and $\{v_i\}$ the set of vertex (node) motion vectors. $g_k(s)$ represents the intensity gradient of frame k at location s . In our experiments, we use a rectangular tiling of the plane, and all references to “blocks” refer to a set of pixels in the present frame belonging to one of the rectangles, with four nodes at four corners. We note, however, that all developments are completely general and can be applied to arbitrary tilings. Blocks are enumerated by $B \in \mathcal{B}$, and the set of pixels belonging to a block B is denoted by \mathcal{S}_B . All vectors are column vectors, and the superscript t denotes vector transpose.

Following [19], the optimality condition for the warping kernel is given below.

Proposition 1: Given an n th order linear interpolation model for motion

$$v(s) = \sum_{i=1}^n a_i(s)v_i(B), \quad s \in \mathcal{S}_B \quad (1)$$

a set of necessary conditions for optimal interpolation parameters is

$$\mathcal{E}_B\{[I_k(s) - I_{k-1}(s - v(s))]g_{k-1}^t(s - v(s))v_i(B)\} = 0 \quad (2)$$

where $\mathcal{E}_B\{\cdot\}$ denotes the expected value, taken over $B \in \mathcal{B}$.

This condition is derived by setting the partial derivatives of the distortion cost function to zero. The distortion function is defined as

$$\begin{aligned} D(s) &= \mathcal{E}_B\{|I_k(s) - \hat{I}_k(s)|^2\} \\ &= \mathcal{E}_B\{|I_k(s) - I_{k-1}(s - v(s))|^2\} \end{aligned} \quad (3)$$

where the expectation is typically computed as an average over a training sequence of blocks

$$\frac{\partial D(s)}{\partial a_i} = -2\mathcal{E}_B\left\{[I_k(s) - I_{k-1}(s - v(s))] \cdot \frac{\partial I_{k-1}(s - v(s))}{\partial a_i}\right\}. \quad (4)$$

Consider the total differential

$$\begin{aligned} dI_{k-1}(s - v(s)) &= \frac{\partial I_{k-1}(s - v(s))}{\partial v_x} dv_x \\ &\quad + \frac{\partial I_{k-1}(s - v(s))}{\partial v_y} dv_y. \end{aligned}$$

³For coding and compression applications, replace I_{k-1} in all succeeding developments with I_{k-1} , the decoded past frame.

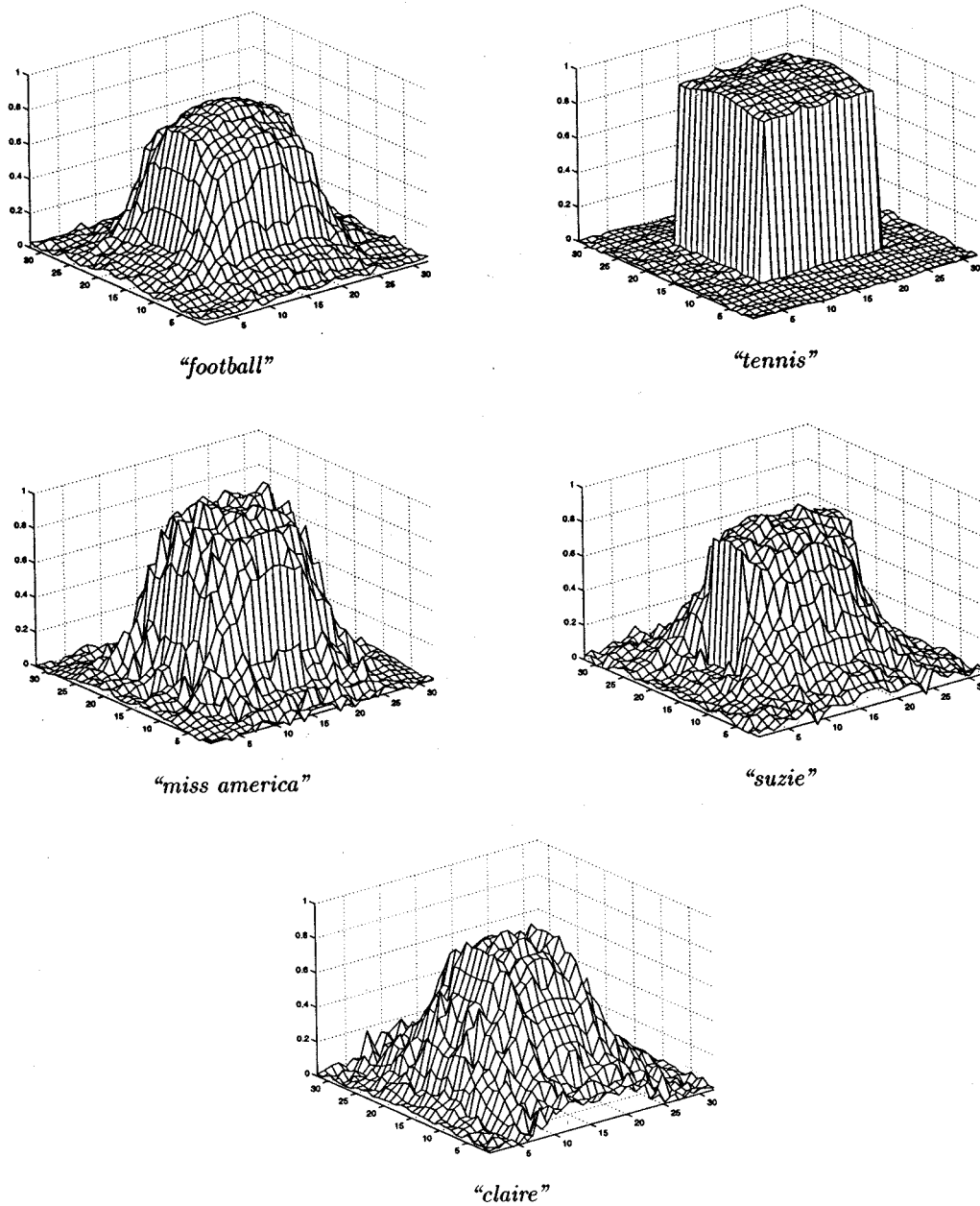


Fig. 5. Optimal kernel for various test sequences.

Then

$$\begin{aligned}
 & \frac{\partial I_{k-1}(s - v(s))}{\partial a_i} \\
 &= \frac{\partial I_{k-1}(s - v(s))}{\partial v_x} \frac{\partial v_x}{\partial a_i} + \frac{\partial I_{k-1}(s - v(s))}{\partial v_y} \frac{\partial v_y}{\partial a_i} \\
 &= -g_{k-1}^t(s - v(s)) \frac{\partial v(s)}{\partial a_i} \\
 &= -g_{k-1}^t(s - v(s)) v_i(B)
 \end{aligned} \tag{5}$$

where the last step makes use of (1). Substitution in (4) gives (2).

It is noteworthy that (2) is a generalized orthogonality condition. $g_{k-1}^t(s - v(s))v_i(B)$ is simply the (deterministic) projection of motion vectors of the vertices of block B onto the inten-

sity gradients of frame $k - 1$. Equation (2) states that optimality is achieved only if this projection is statistically orthogonal to estimation errors. Projections $g_{k-1}^t(s - v(s))v_i(B)$ are our net “observations” in this estimation problem, since only the component of motion vector along the intensity gradient contributes to the intensity estimate.

Fig. 5 demonstrates optimal warping kernels for five different video test sequences. These kernels were computed via gradient descent, using the gradient expressions of this section on the first 50 frames of each sequence (CIF resolution, 30 frames per second). The block motion vectors were computed using 16×16 blocks, 31×31 search area, and exhaustive search. Observe that the optimal kernel varies significantly from one video sequence to another. Details of the computation of these kernels are addressed in Section IV.

III. PARAMETRIC WARPING KERNELS

The warping kernel corresponding to 16×16 motion blocks, with quadrantal symmetry, has 256 degrees of freedom. These parameters could be updated typically every few seconds in a video stream. In some applications, such as motion estimated denoising or interpolation, the large number of parameters is not necessarily a problem. But in video compression applications, these parameters are transmitted to the decoder as overhead. This overhead cannot be very large, especially at low-bit-rate applications.

One can avoid the issue of overhead by a backward-looking computation of the kernels at the encoder and decoder, using information from the past frames. This removes the necessity of transmitting the parameters. Experiments show that the optimal kernels vary slowly within the same video sequence, thus a backward-looking approach is feasible as far as performance is concerned. Unfortunately, however, the backward-looking approach is not without its problems, such as loss of tracking (e.g., due to channel errors). This and other related issues motivate us to remain with a “forward” approach, where there are strong incentives to reduce the overhead as much as possible. In the following, we present a parametric version of our kernels which captures almost all the performance of optimal kernels with only a small (almost negligible) fraction of the overhead.

An efficient parameterization is possible because the optimal kernels have a recognizable regularity. A look at Fig. 5 indicates that warping kernels are far from arbitrary. Generally these kernels are dome shaped, with higher values in the middle (close to the polygon vertex, i.e., the motion vector), and smaller values away from the motion vector. This makes intuitive sense: a motion vector is more efficient in estimating close-by pixel values than those far away. The exact nature of this relationship depends on the characteristics of the video sequence, and the optimal strategy can be anywhere between completely discontinuous (e.g., block matching) to very smooth (e.g., bilinear warping). The constrained kernels presented below are parameterized by the degree of this “smoothness.”

A. One-Parameter Warping Kernels

To capture the variations of warping kernels, we propose to use the following function:

$$f(x) = \frac{1}{1 + e^x}. \quad (7)$$

The symmetry of this function and its derivative make it a computationally attractive choice for our optimization procedure

$$f'(x) = -f(x)f(-x) \quad (8)$$

$$f(-x) = 1 - f(x). \quad (9)$$

The simple closed-form derivative is especially useful in the development of descent algorithms. We introduce the smoothness parameter γ , which controls the effect of each motion vector on the motion field in its surrounding 32×32 -pixel region of influence. The function $h(\cdot)$ is designed to normalize to unity at origin and go to zero at the boundaries of the region

$$h_\gamma(x) = \frac{f(\gamma(2x - 1)) - f(\gamma)}{f(-\gamma) - f(\gamma)}. \quad (10)$$

This family of functions is depicted in Fig. 6. We use a separable construction to generate one quadrant of the warping kernel

$$h_\gamma(x, y) = h_\gamma(x)h_\gamma(y) \quad (x, y) \in [0, 1] \times [0, 1] \quad (11)$$

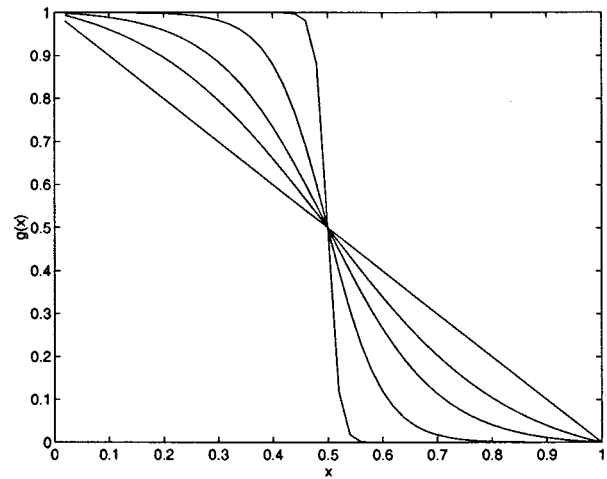


Fig. 6. Family of parametric interpolation functions $h_\gamma(x)$. From $\gamma = 1$ representing almost a straight line to $\gamma = 50$ representing almost a brick-wall. Intermediate values are $\gamma = 3, 5, 10$.

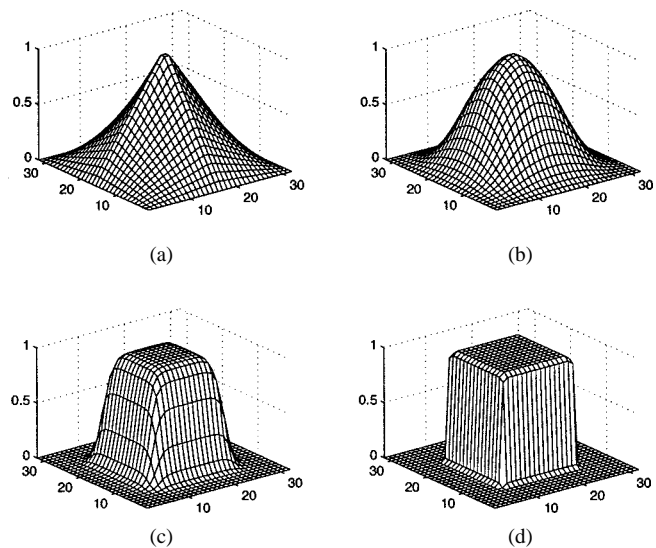


Fig. 7. Corresponding family of 2-D kernels. (a)–(d), respectively: $\gamma = 1, 3, 10, \text{ and } 50$.

$h_\gamma(x, y)$ is defined over the domain $[-1, 1] \times [-1, 1]$, using a symmetric extension of (11)

$$h_\gamma(x, y) = h_\gamma(-x, y) = h_\gamma(x, -y) = h_\gamma(-x, -y). \quad (12)$$

Fig. 6 shows $h_\gamma(x)$, and Fig. 7 shows examples of the corresponding 2-D kernels.

To find the optimal parameter γ , we use the same technique used in Section II, except this time the optimization is only over a single parameter γ . The cost function D representing the overall distortion is shown in (3), and the computation of optimality conditions follow the same lines as developed in Proposition 1. In particular

$$\frac{\partial D}{\partial \gamma} = -2\mathcal{E} \left\{ \sum_x [I_k(s) - I_{k-1}(s - v(s))] \cdot \frac{\partial I_{k-1}(s - v(s))}{\partial \gamma} \right\} = 0. \quad (13)$$

The derivative of the motion-compensated past frame is expressible in terms of the local intensity gradients and derivative of pixel-wise motion values

$$2\mathcal{E} \left\{ \sum_x [I_k(s) - I_{k-1}(s - v(s))] g_{k-1}^t(s - v(s)) \frac{\partial v(s)}{\partial \gamma} \right\} = 0. \quad (14)$$

Now we incorporate the parametric interpolator. Denoting the pixel location as the vector $s = [xy]^t$, and the motion vectors at the vertices of a given rectangle by $\{v_0, v_1, v_2, v_3\}$

$$\begin{aligned} v(x, y) &= v_0 h_\gamma(x, y) + v_1 h_\gamma(1 - x, y) \\ &\quad + v_2 h_\gamma(x, 1 - y) + v_3 h_\gamma(1 - x, 1 - y) \\ &= h_\gamma(x) [v_0 h_\gamma(y) + v_2 h_\gamma(1 - y)] \\ &\quad + h_\gamma(1 - x) [v_1 h_\gamma(y) + v_3 h_\gamma(1 - y)] \\ &= h_\gamma(x) [v_2 + (v_0 - v_2) h_\gamma(y)] \\ &\quad + (1 - h_\gamma(x)) [v_3 + (v_1 - v_3) h_\gamma(y)] \end{aligned} \quad (15)$$

where we have used the identity $h(\alpha) + h(1 - \alpha) = 1 \forall \alpha \in [0, 1]$. For notational convenience, let the prime denote differentiation with respect to the subscript, i.e., $h'_\gamma(\cdot) = \partial h_\gamma(\cdot) / \partial \gamma$. After some algebra

$$\begin{aligned} \frac{\partial v(x, y)}{\partial \gamma} &= h'_\gamma(x) [(v_2 - v_3) + (v_0 - v_1 - v_2 + v_3) h_\gamma(y)] \\ &\quad + h'_\gamma(y) [(v_1 - v_3) + (v_0 - v_1 - v_2 + v_3) h_\gamma(x)]. \end{aligned} \quad (16)$$

Substitution in (14) completes the derivation of the optimality conditions. Notice that h_γ and its derivative h'_γ do not depend on either the intensity or motion data. Therefore, in the optimization process, they can be pre-computed and stored in a lookup table. It is straightforward to show

$$\begin{aligned} \frac{\partial h_\gamma(x)}{\partial \gamma} &= \frac{(2x - 1)A(1 - A) - B(1 - B)}{1 - 2B} \\ &\quad + \frac{2B(1 - B)(A - B)}{(1 - 2B)^2} \end{aligned} \quad (17)$$

where

$$A = f(\gamma(2x - 1)), \quad B = f(\gamma). \quad (18)$$

The single-parameter optimization resulting from this approach is not only more convenient for communication purposes, it is also computationally easier and the computed optimal point is more reliable, due to the reduced dimensionality. It is worth noting again that this parameterization includes, as special cases, both the block-matching and the standard warping kernels. Application of this parametric optimization to test video sequences is shown in Fig. 8. We discuss the computational aspects of the parametric optimization in Section IV, and performance issues are addressed in Section V.

B. Two-Parameter Warping Kernels

A comparison of the parametric kernels computed in the last section, and the corresponding unconstrained kernels shown in

Fig. 5 indicates that the one-parameter family of kernels are a fairly good representation of the unconstrained kernels. It is only at the boundaries of the 32×32 -pixel area that some discrepancies are observed. More specifically, the one-parameter kernel goes to zero at these boundaries, while the unconstrained kernel may not (compare Figs. 5 and 8). In the following, we present a modification of the parametric kernel to account for this effect. We propose a two-parameter family of interpolation kernels

$$h_{\gamma, \delta} = \frac{f(\gamma(2x - 1)) - f(\gamma) + \delta}{f(-\gamma) - f(\gamma) + 2\delta}. \quad (19)$$

This allows one more degree of freedom. γ is the same ‘‘smoothness’’ parameter as before, and δ controls the value of the kernel at the boundaries, as shown in Fig. 9. The optimality conditions are

$$\frac{\partial D}{\partial \gamma} = 0, \quad \frac{\partial D}{\partial \delta} = 0. \quad (20)$$

They lead to a derivation virtually similar to Section III-A, which we do not repeat. The only difference is in the derivatives of $h_{\gamma, \delta}$, easily shown to be

$$\begin{aligned} \frac{\partial h_{\gamma, \delta}}{\partial \gamma} &= \frac{(2x - 1)A(1 - A) - B(1 - B)}{1 - 2B + 2\delta} \\ &\quad + \frac{2B(1 - B)(A - B + \delta)}{(1 - 2B + 2\delta)^2} \end{aligned} \quad (21)$$

$$\frac{\partial h_{\gamma, \delta}}{\partial \delta} = \frac{1 - 2A - 4B}{(1 - 2B + 2\delta)^2}. \quad (22)$$

Similar to the one-parameter case, the derivatives of $h_{\gamma, \delta}$ do not depend on the intensity and/or motion data, and thus can be pre-computed and stored in a lookup table. The second component ($\partial h / \partial \delta$) adds very little complexity compared to the one-parameter case, because its denominator is already computed in (21), and its numerator is easily calculated with no new function evaluations.

Optimizing the two-parameter family of kernels on the test sequences yields the shapes shown in Fig. 10. The gradient descent optimization is made relatively easy through the closed form gradients, as shown in the subsequent section on computational issues. Notice that differences between one-parameter and a two-parameter kernels depend on the video stream. While the two kernels are almost indistinguishable in the case of ‘‘tennis,’’ in other examples the kernels take a slightly different shape, especially close to the boundaries, where the two-parameter kernels does not go to zero.

IV. COMPUTATIONAL ISSUES

This section presents an analysis of the computational cost of various components of mesh-based motion estimation and compensation systems. We look into the complexity of block-matching motion search, fully iterative mesh-based motion search, as well as the cost of computing and using the new warping kernels.

We start with block matching. BMA motion vectors are used directly with the kernels introduced in this paper, and are also

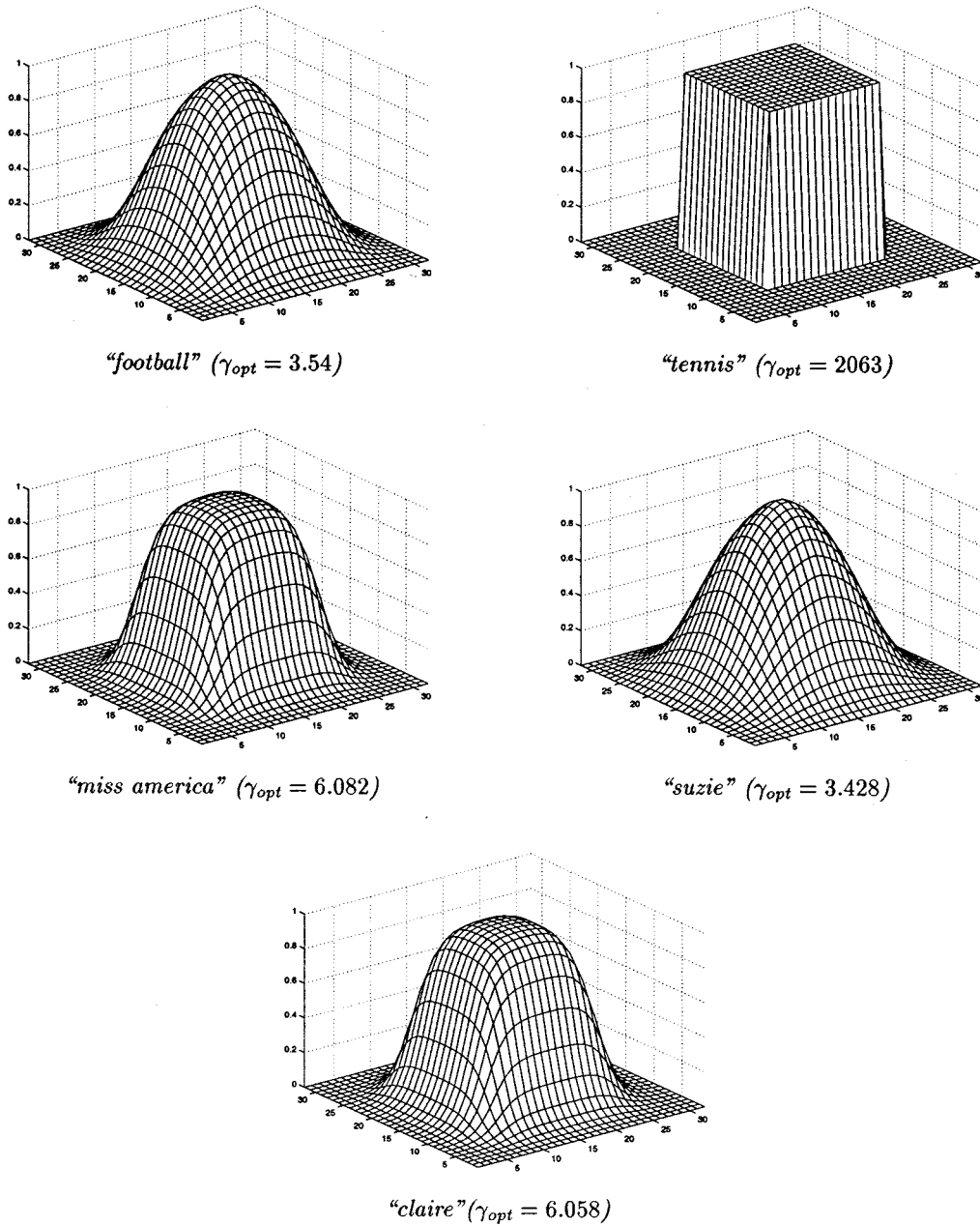


Fig. 8. One-parameter optimal kernels for several test sequences.

used as a seed or initial point in traditional (iterative) mesh-based motion estimation. We concentrate on simple full-search algorithms at single-pel accuracy, which we used in our experiments.⁴ Assume each frame is divided into $n \times n$ blocks. The search area is assumed to be approximately $2n \times 2n$. This is

⁴There exist various fast algorithms for computing motion vectors [20]. These fast algorithms rely on downsampling strategies, either in the intensity fields, or in the candidate motion fields. Such shortcuts achieve varying amounts of computational savings, at the cost of some level of distortion. Also, intelligent ordering and branching in the computation of frame differences reduces the computational complexity of the motion search, but makes it sequence-dependent. Analysis of these variations, with their associated heuristics, is beyond the scope of this paper. We limit ourselves to a comparison of exhaustive search methods for all algorithms in this study.

usually a good upper bound on the typical practical search areas. Using the mean absolute distance (MAD) measure

$$v^* = \arg \min_{v \in \mathcal{S}} \sum_{s \in \mathcal{B}} |I_k(s) - I_{k-1}(s - v)|. \quad (23)$$

The block-matching motion search requires on the order of $4n^4$ additions, $4n^2$ comparisons, and no multiplies, per motion vector. We note also that there are various methods of ordering and halting partial computation in (23), such that *on average* computational complexity is reduced. The exact effects of such strategies are not easily quantified, and we do not consider them in our comparisons.

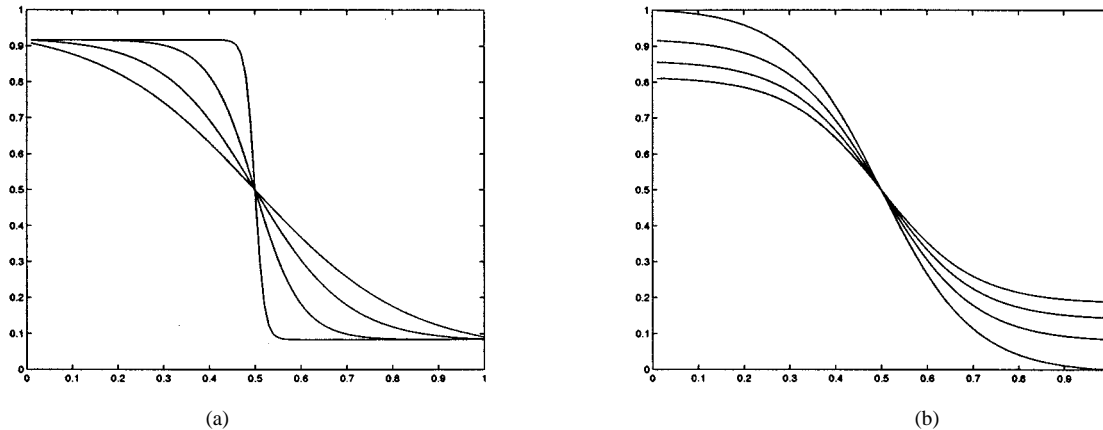


Fig. 9. Two-parameter family of functions $h_{\gamma,\delta}$ in 1-D. (a) With $\delta = 0.1$ and $\gamma = 3, 5, 10, 50$. (b) With $\gamma = 5$ and $\delta = 0, 0.1, 0.2, 0.3$.

The computational complexity of block matching was expressed above in terms of number of operations per motion vector. In what follows, we continue to normalize the computations per motion vector to hide extraneous parameters that have no direct bearing on our comparisons, e.g., frame size.

A. Complexity of Iterative Mesh-Based Motion Estimation

In mesh-based motion search, the candidate positions for each node in the past frame constitute an area bound by neighboring nodes, such that the resulting polygons do not overlap. A typical example is shown in Fig. 2. It is not difficult to see that, over all nodes and disregarding boundary effects, the shaded area averages to $2n \times 2n = 4n^2$, since the sum of all such shaded areas is four times the area of the frame.

At each of these $4n^2$ candidate points, one needs to find the minimum absolute distance computed by

$$D = \sum_{s \in P} |I_k(s) - I_{k-1}(s - v(s))|$$

where P is the set of pixels in the four blocks corresponding to the shaded search area, and

$$v(s) = \sum_{i=0}^3 a_i(s)v_i.$$

Computation of the effective motion vector $s - \sum a_i v_i$ at each pixel requires eight multiplications and eight additions. However, because of the structure of bilinear kernel, one can use a shortcut in computation known as the *scanline algorithm* [3]. This method utilizes the linearity of bilinear kernel along the horizontal and vertical directions, and thus needs only two additions per pixel (plus a few overhead operations).

Because $v(s)$ is generally not integer valued, interpolation between pixels is necessary to compute the luminance estimate from the past frame. An intelligent implementation of bilinear interpolation (to a given accuracy, and using table lookup for coefficients) requires four multiplications and three additions per pixel. Then, one more addition is needed to form the frame difference at this pixel. The total computation up to this point is six additions and four multiplications per pixel.

The frame difference must be computed over the shaded area of Fig. 2, which has $4n^2$ pixels (on average). To combine all the pixel values to form the cost function, we need another $4n^2 - 1$ addi-

tions. This means $16n^2$ multiplications and $28n^2 - 1$ additions to calculate the cost function at each candidate node position.

To complete the motion search, this computation has to be repeated $4n^2$ times (once for each candidate node position), giving a total of $64n^4$ multiplies and $112n^4 - 4n^2$ additions, per motion vector.

Then the whole process has to be repeated iteratively for reasons mentioned in Section I. Typically, two to four repetitions are needed for convergence.

Once the optimal motion vectors are found, they are used to find the estimated frame intensity $\hat{I}_k(s) = I_{k-1}(s - \sum a_i v_i)$. At each pixel, this requires four multiplications and five additions, totaling $4n^2$ multiplies and $5n^2$ additions per motion vector.

B. Complexity of Computing and Using the New Kernels

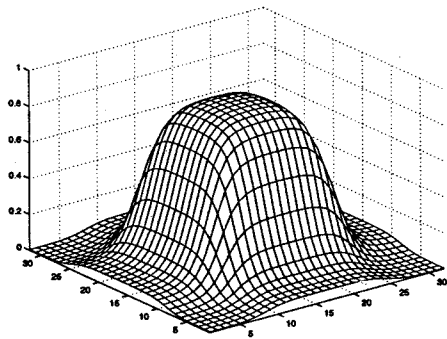
For the general warping kernel, scanline algorithms can no longer be used. Therefore, the computation of $s - v(s)$ requires eight additions and multiplies, and considering the inter-pixel interpolation, the cost of computing the estimated intensity is $12n^2$ multiplications and $11n^2$ additions per motion vector. The other issue is the cost of computing the optimal kernels. In particular, one is interested to know if the cost of finding suitable kernels is small compared to the cost of iterated mesh-based motion search, which is $O(n^4)$.

The computation of optimal kernels is performed through an iterative optimization process, with two dominant elements in each iteration: calculation of the cost function and its gradient. The cost function is nothing but the displaced frame difference, whose computation is essentially the same as the intensity estimate, with one more operation. Therefore, the cost function requires $12n^2$ multiplications and $12n^2$ additions per motion vector.

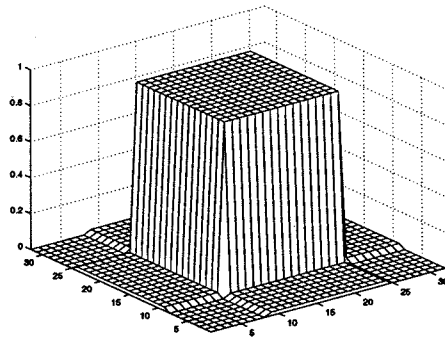
The computational complexity of the cost gradient depends on the kernel. First, we consider the general unconstrained kernel developed in Section II. Recall that the cost gradient is calculated through

$$\frac{\partial D}{\partial a_i} = 2\mathcal{E}_B\{[I_k(s) - I_{k-1}(s - v(s))]g_{k-1}^t(s - v(s))v_i(B)\}.$$

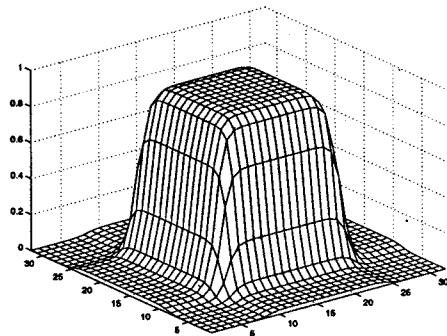
The computation of $s - v(s)$ requires eight addition and multiplications. Calculating intensity gradients g_k^t and intensity difference $I_k(s) - I_{k-1}(s - v(s))$ would require interpolation, but



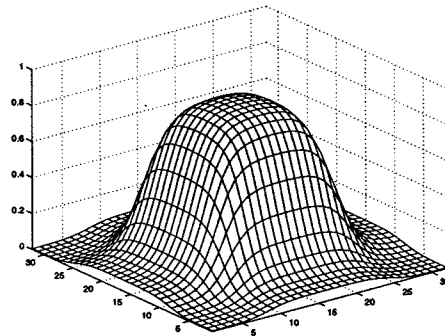
“football” ($\gamma_{opt} = 6.279$, $\delta_{opt} = 0.0748$)



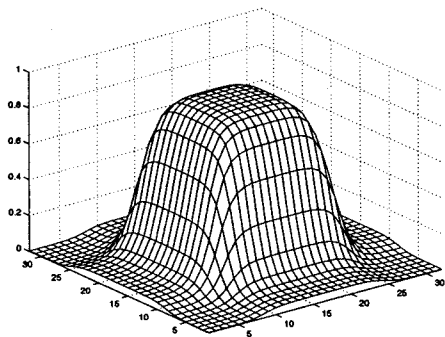
“tennis” ($\gamma_{opt} = 4582$, $\delta_{opt} = 0.0257$)



“miss america” ($\gamma_{opt} = 13.40$, $\delta_{opt} = 0.0281$)



“suzie” ($\gamma_{opt} = 5.274$, $\delta_{opt} = 0.0671$)



“claire” ($\gamma_{opt} = 7.816$, $\delta_{opt} = 0.0545$)

Fig. 10. Two-parameter optimal kernels for several test sequences.

we find that for the purposes of optimization, a nearest pixel approximation is sufficient, resulting in two and one more additions, respectively. The expression inside the expected value contains three more multiplications and one addition, and the computation of the expectation on average requires one addition per motion vector; a total of 13 additions and 11 multiplications. These operations have to be repeated for each $a_i(s)$ $i = 0, 1, 2, 3$, and for each pixel over the support of the kernel, of size $4n^2$. This means a total of $188n^2$ multiplications and $220n^2$ additions for both the cost function and its derivative.

For the parametric kernel, recall that the cost function gradient is computed through

$$\frac{\partial D}{\partial \gamma} = 2\mathcal{E} \left\{ \sum_x [I_k(s) - I_{k-1}(s - v(s))] g_{k-1}^t(s - v(s)) \frac{\partial v(s)}{\partial \gamma} \right\}$$

where $\partial v(s)/\partial \gamma$ is given in (16). With suitable arrangement of terms, and noting that $h(\cdot)$ and $h'(\cdot)$ are computed off-line, (16) requires 8 multiplications and 14 additions. Along the same lines as the previous case, it is easily verified that the cost function gradient requires 11 multiplications and 18 additions for the expression inside the expected value. In contrast to the previous case, however, this expectation is computed once over all pixels, not individually for each point in the support of the kernel, resulting in $11n^2$ multiplications and $18n^2 + n^2 = 19n^2$ additions per motion vector for the gradient. The cost function requires $12n^2$ operations per motion vector, thus the total comes up to $23n^2$ multiplications and $31n^2$ additions.

The cost function and its gradient have to be recomputed in each iteration of the optimization process. Typically three to four iteration steps are sufficient for convergence. Table I

TABLE I
COMPARISON OF COMPUTATIONAL COMPLEXITIES. THE NEW OPTIMAL AND PARAMETRIC KERNELS HAVE MUCH SMALLER COMPUTATIONAL COMPLEXITIES THAN THE TRADITIONAL MESH-BASED METHOD (ITERATIVE MESH-BASED)

	Multiplications	Additions
Block Matching	0	$4n^4$
Iterative Mesh-based	$64Kn^4 + 4n^2$	$4n^4 + K(112n^4 - 4n^2) + 5n^2$
Optimal Kernel	$188K'n^2 + 12n^2$	$4n^4 + 220K'n^2 + 12n^2$
Parametric Kernel	$23K'n^2 + 12n^2$	$4n^4 + 31K'n^2 + 12n^2$

TABLE II
ESTIMATION GAIN (dB) OVER BLOCK MATCHING, FOR 50 FRAMES OF TEST SEQUENCES USING VARIOUS KERNELS. ALL TESTS USE BLOCK-MATCHING MOTION VECTORS, EXCEPT THE LAST COLUMN WHERE ITERATIVE WARPING MOTION VECTORS ARE USED

Sequence	Bilinear	Optimal	1-parameter	2-parameter	Iterative
football	0.5444	0.7186	0.6435	0.6938	0.1124
tennis	-1.0011	0.1745	0.0000	0.1717	-0.0776
claire	-0.3603	0.6269	0.3766	0.4161	1.8983
miss america	0.0600	0.2468	0.2023	0.2392	0.9884
suzie	0.4111	0.5454	0.4933	0.5402	1.6249

presents a comparison of the computational complexity of various methods discussed in this section. “Full mesh” refers to traditional, iteratively computed mesh-based motion, and the numbers in that column show the cost of one iteration of motion estimation. K is the number of iterations in traditional full-mesh motion search, and K' is the number of iterations in the optimization of our new kernels. Both K and K' take typical values of three to four. The numbers in Table I show that computation and usage of new kernels is dominated by the cost of block-matching motion search, and is far more economical than fully iterative mesh-based motion estimation. The $4n^4$ additions appearing in all cases reflect the cost of block matching, which is used in all methods to compute either the motion vectors themselves, or to compute the initial condition in the case of fully iterative mesh-based motion.

In practical situations, the new kernels are even more economical than implied by Table I for the following reason. When the motion vectors at the vertices of a polygon are equal, the motion inside the polygon will be constant (translational). This is due to the symmetries of the kernel. It follows that when the vertex motions are equal, the shape of the kernel has no effect on the pixel-wise motion values, and hence it does not affect the intensity estimate or the estimation error. Therefore, all blocks that have equal motion vectors at the vertices can be removed from the computation of the cost function and its gradient. In practice, as many as one half of the blocks may be inactive, leading to a large computational saving.

V. NUMERICAL RESULTS

We computed general and parametric warping kernels for five video test sequences “football,” “tennis,” and “suzie,” (352×240) “claire,” and “miss america,” (352×288) all at 30 frames per second. Motion vectors were computed using a full search block-matching algorithm with 16×16 blocks and 31×31 search area. In each case the first 50 frames of the sequence were used for our tests. The results appear in Table II.

The numbers in the table show average estimation gain (decibels) in each case, over the simple block-matching motion compensated estimator. The motion vectors in all cases are found using the block-matching algorithm.

The first column, labeled “Bilinear,” shows the performance of the bilinear kernel. The results are very erratic: in some cases there is some gain with respect to block matching, but in other cases there is significant loss, e.g. for “tennis” and “claire.” The unreliable performance of the bilinear kernel was the primary motivation for finding the new kernels.

The next three columns, respectively, show the performance of the unconstrained optimal kernel, and the one- and two-parameter kernels. The kernel designs for each case are shown in Figs. 5, 8 and 10. The gains over block matching vary, but the performance is always superior to the bilinear kernel. There is a slight loss of performance when going from optimal to parametric kernels, but depending on the application, the simplicity of the parametric version may be an overriding factor. Except for “claire,” the two-parameter kernel is within 0.02 dB of the optimal kernel.

All experiments mentioned so far use the block-matching motion vectors. For completeness, we offer one more set of experiments. The last column of Table II shows PSNR results with fully iterative warping. This method, as seen in the last section, is much more computational than the ones represented in the other columns, therefore a direct comparison of results may not be appropriate. With that caveat, we note that the performance of iterative warping is sometimes better (“claire,” “miss america,” and “suzie”) but also sometimes worse than the new kernels (“football” and “tennis”).

The main result of our experiments is that with proper kernels, one can capture some of the gain of warping, while using computationally cheap, noniterative motion vectors.

Figs. 11 and 12 show traces of estimation gain for 150 frames of “tennis” and “claire.” These figures show experiments with the nonparametric kernels; the results for parametric kernels are similar (see Table II). The new kernel in each case is trained only

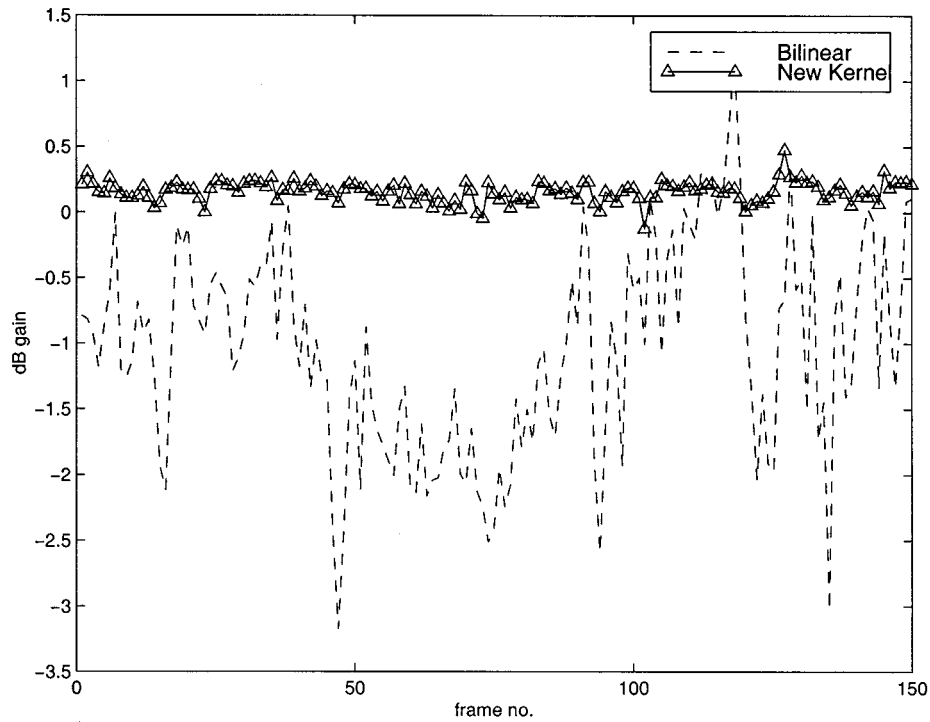


Fig. 11. Performance of various kernels on the "tennis" sequence.

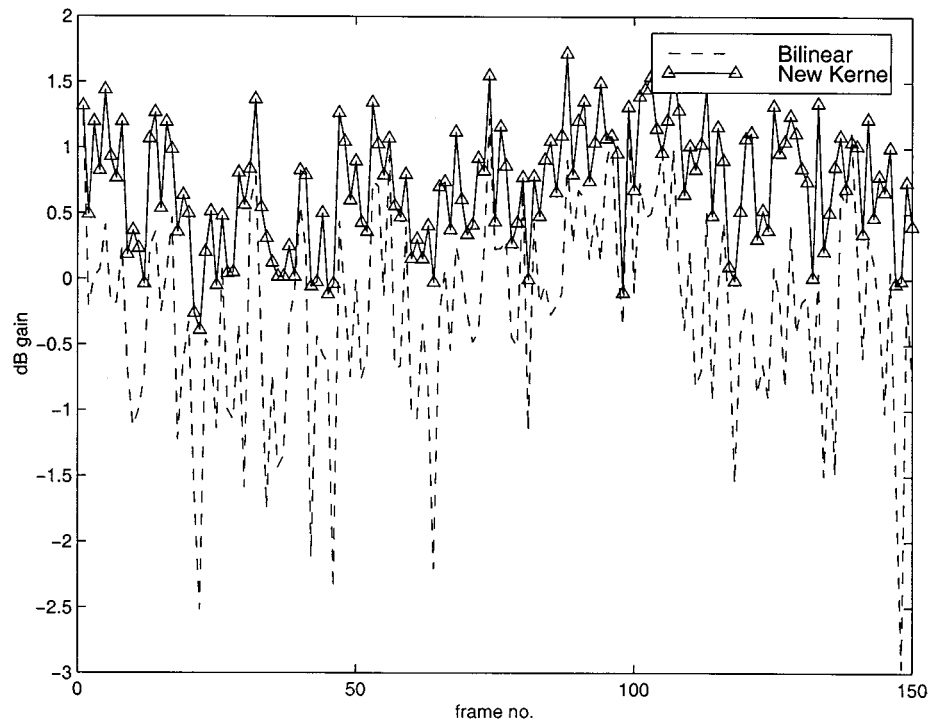


Fig. 12. Performance of various kernels on the "claire" sequence.

over the first 50 frames of the sequence. The sustained performance after frame 50 suggests that, while optimal kernels can vary significantly between different video sequences, they generally vary only slowly within the same sequence. This observation points to further computational advantages, since one may be able to recompute optimal kernels less often, or use fewer frames in their computation.

VI. DISCUSSION AND CONCLUSION

This paper presented a new method for the design of interpolation kernels for mesh-based motion estimation. Traditional mesh-based motion estimation uses a bilinear interpolator, and requires an iterative process for the estimation of motion vectors. This process is typically very cumbersome, leading us

to ask if we can use BMA motion vectors in a mesh-based system.

It was shown that such a direct simplification is not possible within the confines of traditional warping kernels. However, it is possible to use BMA motion vectors with appropriately chosen warping kernels. We developed optimality conditions for these kernels and analyzed their computational complexity. It was shown that, even with optimization, the complexity is still smaller than the fully iterative (traditional) mesh-based motion estimation, and is comparable to the order of computation of the block-matching algorithm.

The new warping kernels depend on the statistics of the video sequence, and vary from one sequence to another. Using a rubber sheet analogy for warping motion estimation, the optimal warping rubber sheet is nonuniform; it is more rigid close to the control points (motion vectors) and more elastic away from them. The necessary condition for optimality can be characterized as a generalized orthogonality condition. It requires that the projection of motion vectors on image intensity gradients be statistically orthogonal to estimation errors.

REFERENCES

- [1] R. M. Haralick, "Automatic remote sensor image processing," in *Topics in Applied Physics, Vol. 11: Digital Picture Analysis*. New York: Springer-Verlag, 1976, pp. 5–63.
- [2] G. J. Holzmann, *Beyond Photography—The Digital Dark-room*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
- [3] G. Wolberg, *Digital Image Warping*. Los Alamos, CA: IEEE Computer Society Press, 1990.
- [4] H. Bruzewitz, "Motion compensation with triangles," in *Proc. 3rd Int. Conf. 64 kbit Coding of Moving Video*, Rotterdam, The Netherlands, Sept. 1990.
- [5] G. J. Sullivan and R. L. Baker, "Motion compensation for video compression using control grid interpolation," in *Proc. IEEE ICASSP*, vol. 4, May 1991, pp. 2713–2716.
- [6] Y. Nakaya and H. Harashima, "An iterative motion estimation method using triangular patches for motion compensation," in *Proc. SPIE Visual Communications and Image Processing*, vol. 1605, Nov. 1991, pp. 546–557.
- [7] Y. Nakaya and H. Harashima, "Motion compensation based on spatial transformations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 339–356, June 1994.
- [8] J. Nieweglowski, T. Campbell, and P. Haavisto, "A novel video coding scheme based on temporal prediction using digital image warping," *IEEE Trans. Consumer Electron.*, vol. 39, pp. 141–150, Aug. 1993.
- [9] C. L. Huang and C. Y. Hsu, "A new motion compensation method for image sequence coding using hierarchical grid interpolation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 42–52, Feb. 1994.
- [10] Y. Wang and O. Lee, "Active mesh—A feature seeking and tracking image sequence representation scheme," *IEEE Trans. Image Processing*, vol. 3, pp. 610–624, Sept. 1994.

- [11] —, "Use of two-dimensional deformable mesh structures for video coding, Part I—The synthesis problem: Mesh based function approximation and mapping," *IEEE Tran. Circuits Syst. Video Technol.*, vol. 6, pp. 636–646, Dec. 1996.
- [12] Y. Wang, O. Lee, and A. Vetro, "Use of two-dimensional deformable mesh structures for video coding, Part II—The analysis problem and a region-based coder employing an active mesh representation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 647–659, Dec. 1996.
- [13] Y. Wang and J. Osterman, "Evaluation of mesh-based motion estimation in h.263 like coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 243–252, June 1998.
- [14] A. M. Tekalp, P. J. L. van Beek, C. Toklu, and B. Gunsel, "2nd mesh-based visual object representation for interactive synthetic/natural video," *Proc. IEEE*, vol. 86, pp. 1029–1051, June 1998.
- [15] C. Toklu, A. Erdem, M. I. Sezan, and A. M. Tekalp, "Tracking motion and intensity variations using hierarchical 2-D mesh modeling for synthetic object transfiguration," *Graphic Models and Image Processing*, vol. 58, pp. 553–573, Nov. 1996.
- [16] A. Nosratinia, N. Mohsenian, M. T. Orchard, and B. Liu, "Interslice coding of magnetic resonance images using deformable triangular patches," in *Proc. IEEE ICIP*, vol. 2, Austin, TX, Nov. 1994, pp. 898–892.
- [17] —, "Interframe coding of magnetic resonance images," *IEEE Trans. Medical Imaging*, vol. 15, pp. 639–647, Oct. 1996.
- [18] N. Mohsenian, A. Nosratinia, B. Liu, and M. T. Orchard, "Adaptive entropy constrained transform coding of magnetic resonance image sequences," *IEEE Trans. Nuclear Sci.*, vol. 42, pp. 2309–2316, Dec. 1995.
- [19] A. Nosratinia and M. T. Orchard, "Optimal warping prediction for video coding," in *Proc. ICASSP*, vol. IV, Atlanta, GA, May 1996, pp. 1986–1989.
- [20] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 148–157, Apr. 1993.



Aria Nosratinia received the B.S. degree in 1988 from the University of Tehran, Iran, and the M.S. degree in 1991 from University of Windsor, Canada, both in electrical engineering, and the Ph.D. degree from the University of Illinois at Urbana-Champaign in 1996, in electrical and computer engineering.

During the academic year 1995–1996, he was with Princeton University, Princeton, NJ. From 1996 to 1999, he was a Visiting Professor and Faculty Fellow at Rice University, Houston, TX. Since July 1999, he has been on the faculty of the University of Texas at

Dallas, where he is currently Assistant Professor of Electrical Engineering. His general research interests are in digital signal processing, image processing, and coding of images and video. In particular, he is interested in problems related to processing and transport of multimedia signals (audio, images, video) in packet switched networks and wireless channels. He has published in various IEEE and European journals, and contributed to two books: *Wavelets, Subbands, and Block Transforms in Communications and Multimedia* (Norwell, MA: Kluwer, 1999) and *Applied and Computational Control, Signals, and Circuits* (Cambridge, MA: Birkhauser, 1999).

Dr. Nosratinia was the recipient of a National Science Foundation Career award in 2000, and a Texas Higher Education Coordinating Board Advanced Research Program Award in 1999.