Motion Estimation with Quadtree Splines

Richard Szeliski, *Member, IEEE Computer Society*, and Heung-Yeung Shum

Abstract—This paper presents a motion estimation algorithm based on a new multiresolution representation, the *quadtree spline*. This representation describes the motion field as a collection of smoothly connected patches of varying size, where the patch size is automatically adapted to the complexity of the underlying motion. The topology of the patches is determined by a quadtree data structure, and both split and merge techniques are developed for estimating this spatial subdivision. The quadtree spline is implemented using another novel representation, the *adaptive hierarchical basis spline*, and combines the advantages of adaptively-sized correlation windows with the speedups obtained with hierarchical basis preconditioners. Results are presented on some standard motion sequences.

Index Terms—Motion analysis, image registration, optical flow, splines, quadtrees, local parametric motion models, multiresolution analysis, image pyramids, hierarchical basis functions, motion segmentation.

1 Introduction

One of the fundamental tradeoffs in designing motion estimation and stereo matching algorithms is selecting the size of the windows or filters to be used in comparing portions of corresponding images. Using larger windows leads to better noise immunity through averaging and can also disambiguate potential matches in areas of weak texture or potential aperture problems. However, larger windows fail where they straddle motion or depth discontinuities, or in general where the motion or disparity varies significantly within the window.

Many techniques have been devised to deal with this problem, e.g., using adaptively-sized windows in stereo matching. In this paper, we present a technique for recursively subdividing an image into square patches of varying size and then matching these patches to subsequent frames in a way which preserves inter-patch motion continuity. Our technique is an extension of the *spline-based image registration technique* presented in [55], [56], and thus has the same advantages when compared to correlation-based approaches, i.e., lower computational cost and the ability to handle large image deformations.

As a first step, we show how using *hierarchical basis splines* instead of regular splines can lead to faster convergence and qualitatively perform a smoothing function similar to regularization. Then, we show how selectively setting certain nodes in the hierarchical basis to zero leads to an *adaptive hierarchical basis*. We can use this idea to build a spline defined over a quadtree domain, i.e., a *quadtree spline*. To determine the size of the patches in our adaptive basis, i.e., the shape of the quadtree, we develop both split and merge techniques based on the residual errors in the current optical flow estimates.

While this paper deals primarily with motion estimation (also known as *image registration* or *optical flow* computation), the techniques developed here can equally well be applied to stereo matching. In our framework, we view stereo as a special case of motion estimation where the *epipolar geometry* (corresponding lines) are known, thus reducing a two-dimensional search space at each pixel to a one-dimensional space. Our techniques can also be used as part of a *direct method* which simultaneously solves for projective depth and camera motion [55].

The adaptive hierarchical basis splines developed in this paper are equivalent to adaptively subdividing global parametric motion regions while maintaining continuity between adjacent patches. We can therefore implement a continuum of motion models ranging from a single global (e.g., affine) motion, all the way to a completely general local motion, as warranted by the data in a given image sequence.

The motion estimation algorithms developed in this paper can be used in a number of applications. Examples include motion compensation for video compression, the extraction of 3D scene geometry and camera motion, robot navigation, and the registration of multiple images, e.g., for medical applications. Feature tracking algorithms based on our techniques [57] can be used in human interface applications such as gaze tracking or expression detection, in addition to classical robotics applications.

The remainder of the paper is structured as follows. Section 2 presents a review of relevant previous work. Section 3 gives the general problem formulation for image registration. Section 4 reviews the spline-based motion estimation algorithm. Section 5 shows how hierarchical basis functions can be used to accelerate and regularize spline-based flow estimation. Section 6 presents our novel quadtree splines and discusses how their shape can be estimated using both split and merge techniques. Section 7 discusses the relationship of adaptive hierarchical basis splines to multiscale Markov Random Fields. Section 8 presents experimental results based on some commonly used motion test sequences. We close with a comparison of our approach to previous algorithms and a discussion of future work.

• *The authors are with Microsoft Corporation, One Microsoft Way, Redmond, WA 98052. E-mail: szeliski@microsoft.com.*

Manuscript received Mar. 31, 1995; revised Oct. 4, 1996. Recommended for acceptance by G. Medioni.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96099.

2 Previous Work

Motion estimation has long been one of the most actively studied areas of computer vision and image processing [2], [12]. Motion estimation algorithms include optical flow (general motion) estimators, global parametric motion estimators, constrained motion estimators (*direct methods*), stereo and multiframe stereo, hierarchical (coarse-to-fine) methods, feature trackers, and feature-based registration techniques. We will use this rough taxonomy to briefly review previous work, while recognizing that these algorithms overlap and that many algorithms use ideas from several of these categories.

The general motion estimation problem is often called *optical flow* recovery [24]. This involves estimating an independent displacement vector for each pixel in an image. Approaches to this problem include gradient-based approaches based on the *brightness constraint* [24], [30], [36], correlation-based techniques such as the *sum of squared differences* (SSD) [3], spatio-temporal filtering [1], [22], [18], [63], and regularization [24], [23], [43]. Nagel [36], Anandan [3], and Otte and Nagel [41] provide comparisons and derive relations between different techniques, while Barron et al. [6] provide some numerical comparisons.

Global motion estimators [29], [7] use a simple flow field model parameterized by a small number of unknown variables. Examples of global motion models include affine and quadratic flow fields. In the taxonomy of Bergen et al. [7], these fields are called parametric motion models, since they can be used locally as well (e.g., affine flow can be estimated at every pixel). The spline-based flow fields we describe in the next section can be viewed as local parametric models, since the flow within each spline patch is defined by a small number of control vertices.

Global methods are most useful when the scene has a particularly simple form, e.g., when the scene is planar. These methods can be extended to more complex scenes, however, by using a collection of global motion models. For example, each pixel can be associated with one of several global motion hypotheses, resulting in a *layered motion model* [62], [26], [17], [10]. Alternatively, a single image can be recursively subdivided into smaller parametric motion patches based on estimates of the current *residual error* in the flow estimate [35]. Our approach is similar to this latter work, except that it preserves inter-patch motion continuity, and uses both split and merge techniques.

Stereo matching [5], [45], [15] is traditionally considered as a separate sub-discipline within computer vision (and, of course, photogrammetry), but there are strong connections between it and motion estimation. Stereo can be viewed as a simplified version of constrained motion estimation where the *epipolar geometry* is given, so that each flow vector is constrained to lie along a known line. While stereo is traditionally performed on pairs of images, more recent algorithms use sequences of images (*multiframe stereo* or *motion stereo*) [11], [34], [39]. The idea of using adaptive window sizes in stereo [38], [40] is similar in spirit to the idea used in this paper, although their algorithm has a much higher computational complexity.

Hierarchical (coarse-to-fine) matching algorithms have a long history of use both in stereo matching [45], [64] and in motion estimation [16], [3], [51], [7]. Hierarchical algorithms first solve the matching problem on smaller, lower-resolution images and then use these to initialize higher-resolution estimates. Their advantages include both increased computational efficiency and the ability to find better solutions by avoiding local minima.

The algorithm presented in this paper is also related to patch-based feature trackers [30], [46], [61]. It differs from these previous approaches in that we use patches of varying size, we completely tile the image with patches, and we have no motion discontinuities across patch boundaries. Our motion estimator can be used as a parallel, adaptive feature tracker by selecting spline control vertices with low uncertainty in both motion components [57].

3 General Problem Formulation

The general motion estimation problem can be formulated as follows. We are given a sequence of images $I_t(x, y)$ which we assume were formed by locally displacing a reference image $I(x, y)$ with horizontal and vertical displacement fields[1] $u_t(x, y)$ and $v_t(x, y)$, i.e.,

$$I_t(x + u_t, y + v_t) = I(x, y). \tag{1}$$

Each individual image is assumed to be corrupted with uniform white Gaussian noise. We also ignore possible occlusions ("foldovers") and disocclusions in the warped images.

Given such a sequence of images, we wish to simultaneously recover the displacement fields (u_t, v_t) and the reference image $I(x, y)$. The maximum likelihood solution to this problem is well known and consists of minimizing the squared error

$$\sum_t \int\int \left[I_t(x + u_t, y + v_t) - I(x, y)\right]^2 dxdy. \tag{2}$$

In practice, we are usually given a set of discretely sampled images, so we replace the above integrals with summations over the set of pixels $\{(x_i, y_i)\}$.

If the displacement fields u_t and v_t at different times are independent of each other and the reference intensity image $I(x, y)$ is assumed to be known, the above minimization problem decomposes into a set of independent minimizations, one for each frame. For now, we will assume that this is the case, and only study the two frame problem, which can be rewritten as

$$E(\{u_i, v_i\}) = \sum_t \left[I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i)\right]^2. \tag{3}$$

This equation is called the *sum of squared differences* (SSD) formula [3]. Expanding I_1 in a first order Taylor series expansion in (u_i, v_i) yields the the *image brightness constraint* [24]

$$E(\{u_i, v_i\}) \approx \sum_i \left[\Delta I + I_x u_i + I_y v_i\right]^2,$$

1. We will use the terms *displacement field, flow field,* and *motion estimate* interchangeably.

where $\Delta I = I_1 - I_0$ and $\nabla I_1 = (I_x, I_y)$ is the intensity gradient.

The squared pixel error function (3) is by no means the only possible optimization criterion. For example, it can be generalized to account for photometric variation (global brightness and contrast changes), using

$$E'(\{u_i, v_i\}) \approx \sum_i \left[I_1(x_i + u_i, y_i + v_i) - cI_0(x_i, y_i) + b\right]^2,$$

where b and c are the (per-frame) brightness and contrast correction terms [29], [21], [19], [37]. Both of these parameters can be estimated concurrently with the flow field at little additional cost. Their inclusion is most useful in situations where the photometry can change between successive views (e.g., when the images are not acquired concurrently).

Another way to generalize the criterion is to replace the squaring function with a non-quadratic penalty function, which results in a *robust motion estimator* which can reject outlier measurements [8], [10], [9]. Another possibility is to weight each squared error term with a factor proportional to

$$\frac{1}{\sigma_I^2 + \sigma_u^2 |\nabla I|^2},$$

where σ_I^2 and σ_u^2 are the variances of the image and derivative noise, which can compensate for noise in the image derivative computation [50]. To further increase noise immunity, the intensity images used in (3) can be replaced by filtered images [13].

The above minimization problem typically has many local minima. Several techniques are commonly used to find a more globally optimal estimate. For example, the SSD algorithm performs the summation at each pixel over an $m \times m$ window (typically 5×5) [3]. More recent variations use adaptive windows [38] and multiple frames [39]. Regularization-based algorithms add smoothness constraints on the u and v fields to obtain good solutions [24], [23], [43]. Finally, multiscale or hierarchical (coarse-to-fine) techniques are often used to speed the search for the optimum displacement estimate and to avoid local minima.

The choice of representation for the (u, v) field also strongly influences the performance of the motion estimation algorithm. The most commonly made choice is to assign an independent estimate at each pixel (u_i, v_i), but global motion descriptors are also possible [29], [7], [55]. One can observe, however, that motion estimates at individual pixels are never truly independent. Both local correlation windows (as in SSD) and global smoothness constraints aggregate information from neighboring pixels. The resulting displacement estimates are therefore highly correlated. While it is possible to analyze the correlations induced by overlapping windows [34] and regularization [52], the procedures are cumbersome and rarely used. For these reasons, we have chosen in our work to represent the motion field as a spline, which is a representation which falls in between per-pixel motion estimates and purely global motion estimates.

4 Spline-Based Flow Estimation

Our approach is to represent the displacements fields $u(x, y)$ and $v(x, y)$ as two-dimensional *splines* controlled by a smaller number of displacement estimates $\hat{u}_j$ and $\hat{v}_j$ which lie on a coarser *spline control grid* (Fig. 1). Notice that we use j to refer to spline control vertices while we continue to use i to index pixels. The value for the displacement at a pixel i can be written as

$$u(x_i, y_i) = \sum_j \hat{u}_j B_j(x_i, y_i) \text{ or } u_i = \sum_j \hat{u}_j w_{ij}, \qquad (4)$$

where the $B_j(x, y)$ are called the *basis functions* and are only non-zero over a small interval (*finite support*). We call the $w_{ij} = B_j(x_i, y_i)$ *weights* to emphasize that the (u_i, v_i) are known linear combinations of the $(\hat{u}_j, \hat{v}_j)$.

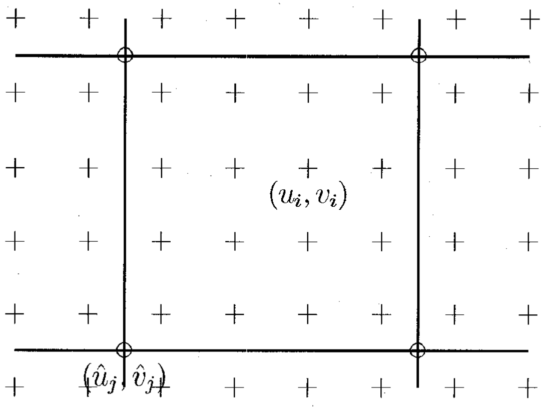


Fig. 1. Displacement spline: The spline control vertices $\{(\hat{u}_j, \hat{v}_j)\}$ are shown as circles (∘) and the pixel displacements $\{(u_i, v_i)\}$ are shown as pluses (+).

In our current implementation, the basis functions are spatially shifted versions of each other, i.e.,

$$B_j(x, y) = B(x - \hat{x}_j, y - \hat{y}_j).$$

We have studied five different interpolation functions:

1) block,
2) linear on squares,
3) linear on triangles,
4) bilinear, and
5) biquadratic [55].

In practice, we most often use the bilinear bases. We also impose the condition that the spline control grid is a regular subsampling of the pixel grid, $\hat{x}_j = mx_i, \hat{y}_j = my_i$, so that each set of $m \times m$ pixels corresponds to a single spline patch.

4.1 Function Minimization

To recover the local spline-based flow parameters, we need to minimize the cost function (3) with respect to the $\{\hat{u}_j, \hat{v}_j\}$. We do this using a variant of the Levenberg-Marquardt iterative non-linear minimization technique [44]. First, we compute the gradient of E in (3) with respect to each of the parameters $\hat{u}_j$ and $\hat{v}_j$,

$$g_j^u \equiv \frac{\partial E}{\partial \hat{u}_j} = 2\sum_i e_i G_i^x w_{ij}$$

$$g_j^v \equiv \frac{\partial E}{\partial \hat{v}_j} = 2\sum_i e_i G_i^y w_{ij}, \tag{5}$$

where

$$e_i = I_1(x_i + u_i, y_i + v_i) - I_0(x_i, y_i) \tag{6}$$

is the intensity error at pixel i,

$$\left(G_i^x, G_i^y\right) = \nabla I_1\left(x_i + u_i, y_i + v_i\right) \tag{7}$$

is the intensity gradient of I_1 at the displaced position for pixel i, and the w_{ij} are the sampled values of the spline basis function (4). Algorithmically, we compute the above gradients by first forming the displacement vector for each pixel (u_i, v_i) using (4), then computing the resampled intensity and gradient values of I_1 at $(x_i', y_i') = (x_i + u_i, y_i + v_i)$, computing e_i, and finally incrementing the g_j^u and g_j^v values of all control vertices affecting that pixel [55].

For the Levenberg-Marquardt algorithm, we also require the approximate Hessian matrix **A** where the second-derivative terms are left out. The matrix **A** contains entries of the form

$$a_{jk}^{uu} = 2\sum_i \frac{\partial e_i}{\partial \hat{u}_j}\frac{\partial e_i}{\partial \hat{u}_k} = 2\sum_i w_{ij}w_{ik}\left(G_i^x\right)^2$$
$$a_{jk}^{uv} = a_{jk}^{vu} = 2\sum_i \frac{\partial e_i}{\partial \hat{u}_j}\frac{\partial e_i}{\partial \hat{u}_k} = 2\sum_i w_{ij}w_{ik}G_i^x G_i^y$$
$$a_{jk}^{vv} = 2\sum_i \frac{\partial e_i}{\partial \hat{v}_j}\frac{\partial e_i}{\partial \hat{v}_k} = 2\sum_i w_{ij}w_{ik}\left(G_i^y\right)^2 \tag{8}$$

The entries of **A** can be computed at the same time as the energy gradients.

The Levenberg-Marquardt algorithm proceeds by computing an increment $\Delta\mathbf{u}$ to the current displacement estimate $\mathbf{u}$ which satisfies

$$(\mathbf{A} + \lambda\mathbf{I})\Delta\mathbf{u} = -\mathbf{g}, \tag{9}$$

where $\mathbf{u}$ is the vector of concatenated displacement estimates $\{\hat{u}_j, \hat{v}_j\}$, $\mathbf{g}$ is the vector of concatenated energy gradients $\{g_j^u, g_j^v\}$, and λ is a stabilization factor which varies over time [44]. To solve this large, sparse system of linear equations, we use preconditioned gradient descent

$$\Delta\mathbf{u} = -\alpha\mathbf{B}^{-1}\mathbf{g} = -\alpha\tilde{\mathbf{g}} \tag{10}$$

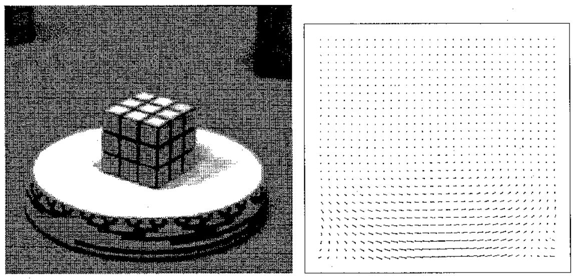
where $\mathbf{B} = \hat{\mathbf{A}} + \lambda\mathbf{I}$, and $\hat{\mathbf{A}}$ = block_diag(**A**) is the set of 2×2 block diagonal matrices defined in (9) with $j = k$, and $\tilde{\mathbf{g}} = \mathbf{B}^{-1}\mathbf{g}$ is called the *preconditioned residual vector*.[2] An optimal value for α can be computed at each iteration by minimizing

$$\Delta E(\alpha\,\mathbf{d}) \approx \alpha^2\mathbf{d}^T\mathbf{A}\mathbf{d} - 2\alpha\mathbf{d}^T\mathbf{g},$$

i.e., by setting $\alpha = (\mathbf{d}\cdot\mathbf{g})/(\mathbf{d}^T\mathbf{A}\mathbf{d})$, where $\mathbf{d} = \tilde{\mathbf{g}}$ is the *direction vector* for the current step. See [55] for more details on our algorithm implementation.

To handle larger displacements, we run our algorithm in a coarse-to-fine (hierarchical) fashion. A Gaussian image pyramid is first computed using an iterated separable 3-point (¼, ½, ¼) filter [13]. We then run the algorithm on one of the smaller pyramid levels, and use the resulting flow estimates to initialize the next finer level (using bilinear interpolation and doubling the displacement magnitudes).

Fig. 2 shows an example of the flow estimates produced by our technique. The input image is 256×240 pixels, and the flow is displayed on a 30×28 grid. We show the results of using a three-level pyramid, nine iterations at each level, and with three different patch sizes, $m = 64$, $m = 16$, and $m = 4$. As we can see, using patches that are too large result in flow estimates which are too smooth, while using patches that are too small result in noisy estimates. (This latter problem could potentially be fixed by adding regularization, but at the cost of increased iterations.) To overcome this problem, we need a technique which automatically selects the best patch size in each region of the image. This is the idea we will develop in the next two sections.


(a) (b)

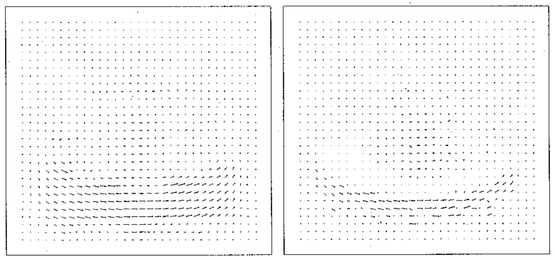

(c) (d)

Fig. 2. Example of general flow computation: (a) input image, (b)–(d) flow estimates for m = 64, 16, and 4.

2. Preconditioning means adjusting the descent direction to accelerate the convergence, e.g., by premultiplying it by an approximate inverse of **A** [4], [44].

5 Hierarchical Basis Splines

Regularized problems often require many iterations to propagate information from regions with high certainty (textures or edges) to regions with little information (uniform intensities). Several techniques have been developed to overcome this problem. Coarse-to-fine techniques [45], [3] can help, but often don't converge as quickly to the optimal solution as multigrid techniques [60]. Conjugate gradient descent can also be used, especially for non-linear problems such as shape-from-shading [49]. Perhaps the most effective technique is a combination of conjugate gradient descent with hierarchical basis functions [65], which has been applied both to interpolation problems in stereo matching [53] and to shape-from-shading [54].

Hierarchical basis functions are based on using a pyramidal representation for the data [13], where the number of nodes in the pyramid is equal to the original number of nodes at the finest level (Fig. 3). To convert from the hierarchical basis representation to the usual fine-level representation (which is called the *nodal basis* representation [65]), we start at the coarsest (smallest) level of the pyramid and interpolate the values at this level, thus doubling the resolution. These interpolated values are then added to the hierarchical representation values at the next lower level, and the process is repeated until the nodal representation is obtained.[3] This process can be written algorithmically as

```
procedure $S$
    for $l = L - 1$ down to 1
        for $j \in \mathcal{M}_l$
            $u_j^l = \hat{u}_j^l + \Sigma_{k \in \mathcal{N}_j} \tilde{w}_{jk} \hat{u}_k^{l+1}$
end $S$.
```

In this procedure, each node is assigned to one of the level collections $\mathcal{M}_l$ (the circles in Fig. 3). Each node also has a number of "parent nodes" $\mathcal{N}_j$ on the next coarser level that contribute to its value during the interpolation process. The $\tilde{w}_{jk}$ are the weighting functions that depend on the particular choice of interpolation function. For the examples shown in this paper, we use bilinear interpolation, since previous experiments suggest that this is a reasonable choice for the interpolator [53].

We can write the above process algebraically as

$$\mathbf{u} = \mathbf{S}\tilde{\mathbf{u}} = \mathbf{S}_1\mathbf{S}_2 \ldots \mathbf{S}_{L-1}\tilde{\mathbf{u}}, \qquad (11)$$

with

$$(\mathbf{S}_l)_{jk} = \begin{cases} 1 & \text{if } j = k \\ \tilde{w}_{jk} & \text{if } j \in \mathcal{M}_l \text{ and } k \in \mathcal{N}_j \\ 0 & \text{otherwise} \end{cases}$$

and $\tilde{\mathbf{u}}$ is the hierarchical basis representation.

Using a hierarchical basis representation for the flow field is equivalent to using $\mathbf{SS}^T$ as a preconditioner, i.e., $\tilde{\mathbf{g}} = \mathbf{SS}^T\mathbf{g}$ [4], [53]. The transformation $\mathbf{SS}^T$ can be used as a preconditioner because the influence of hierarchical bases at coarser levels (which are obtained from the $\mathbf{S}^T$ operation) are propagated to the nodal basis at the fine level through the $\mathbf{S}$ operation. To evaluate $\mathbf{S}^T$, i.e., to convert from the nodal basis representation to the hierarchical basis representation, we use the procedure

```
procedure $S^T$
    for $l = 1$ to $L - 1$
        for $k \in \mathcal{M}_{l+1}$
            $\hat{u}_k^{l+1} = u_k^{l+1} + \Sigma_{j: k \in \mathcal{N}_j} \tilde{w}_{jk} \hat{u}_j^l$
end $S^T$,
```

i.e., a bottom-up sweep is used to aggregate gradient values at coarser levels.

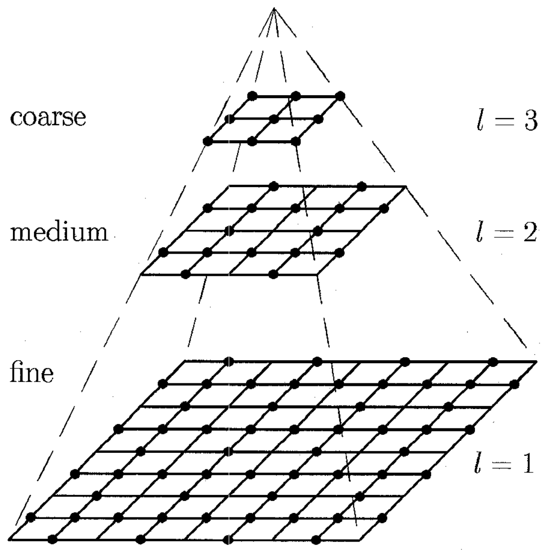


Fig. 3. Hierarchical basis representation. The multiple resolution levels are a schematic representation of the hierarchical basis spline. The circles indicate the nodes in the hierarchical basis. The total number of circles is equal to the number of variables at the finest (l = 1) level of the pyramid.

When combining hierarchical basis preconditioning with the block diagonal preconditioning in (10), we have several choices. We can apply the block diagonal preconditioning first, $\tilde{\mathbf{g}} = \mathbf{SS}^T\mathbf{B}^{-1}\mathbf{g}$, or second, $\tilde{\mathbf{g}} = \mathbf{B}^{-1}\mathbf{SS}^T\mathbf{g}$, or we can interleave the two preconditioners $\tilde{\mathbf{g}} = \mathbf{SB}^{-1}\mathbf{S}^T\mathbf{g}$, or $\tilde{\mathbf{g}} = \hat{\mathbf{B}}^{-T}\mathbf{SS}^T\hat{\mathbf{B}}\mathbf{g}$, where $\hat{\mathbf{B}} = \mathbf{B}^{\frac{1}{2}}$. The latter two operations correspond to well-defined preconditioners (i.e., optimization under a change of basis), while the first two are easier to implement. In our current work, we use the first form, i.e., we apply block preconditioning first, and then use sweep up and then down the hierarchical basis pyramid to smooth the residual. In future work, we plan to develop optimal combinations of block diagonal and hierarchical basis preconditioning.

3. Hierarchical basis splines are therefore a degenerate (nonorthogonal) form of wavelets [32] with extremely compact support and inverses. The reconstruction process is also similar to the reconstruction of an image from a band-pass pyramid, except that the band-pass levels are not fully populated.

To summarize our algorithm (Fig. 4), we keep both the hierarchical and nodal representations, and map between the two as required. For accumulating the gradients required in (5) and (9), we compute the image flows and the derivatives with respect to the parameters in the nodal basis. We then use the hierarchical basis to smooth the residual (gradient) vector $\mathbf{g}$ before selecting a new conjugate direction and computing the optimal step size. Using this technique not only makes the convergence faster but also propagates local corrections over the whole domain, which tends to smooth the resulting flow significantly.

```
0.   $\beta_0 = 0,\ \mathbf{d}_{-1} = 0$
1.   $\mathbf{g}_n = -\nabla E(\mathbf{u})$
2.†  $\tilde{\mathbf{g}}_n = \mathbf{SZS}^T\mathbf{B}^{-1}\mathbf{g}_n$
3.   $\beta_n = \tilde{\mathbf{g}}_n \cdot \mathbf{g}_n \,/\, \tilde{\mathbf{g}}_{n-1} \cdot \mathbf{g}_{n-1}$
4.   $\mathbf{d}_n = \tilde{\mathbf{g}}_n - \beta_n \mathbf{d}_{n-1}$
5.   $\alpha_n = \mathbf{d}_n \cdot \mathbf{g}_n \,/\, \mathbf{d}_n^T \mathbf{A} \mathbf{d}_n$
6.   $\mathbf{u}_{n+1} = \mathbf{u}_n + \alpha_n \mathbf{d}_n$
7.   increment $n$, loop to 1.

†    $\mathbf{S}$ = mapping from hierarchical to nodal basis,
     $\mathbf{B}$ = block_diag($\mathbf{A}$),
     $\mathbf{Z}$ = 0/1 matrix for quadtree spline basis (Section 6).
```

Fig. 4. Hierarchical basis preconditioned conjugate gradient algorithm.

To demonstrate the performance improvements available with hierarchical basis functions, we use as our example the **Square 2** sequence, which is part of the data set used by Barron et al. [6]. Fig. 5a shows one image in the sequence, while Fig. 5b shows the convergence rates for regular gradient descent (GD), coarse-to-fine estimation (CTF, three levels), and preconditioning with hierarchical basis functions (HBPCG, three levels), with different amounts of regularization ($\lambda_1 = 0, 100, 1000$).[4] As we can see from these results, adding more regularization results in a more accurate solution (this is because the true flow is a single constant value), using coarse to fine is quicker than single-level relaxation, and hierarchical basis preconditioning is faster than coarse-to-fine relaxation. It is interesting to note that using hierarchical basis functions even without regularization quickly smooths out the solution and outperforms coarse-to-fine without regularization.

6 Quadtree (Adaptive Resolution) Splines

While hierarchical basis splines can help accelerate an estimation algorithm or even to add extra smoothness to the solution, they do not in themselves solve the problem of having adaptively-sized patches. For this, we will use the idea of *quadtree splines*, i.e., splines defined on a quadtree domain. A quadtree is a 2D representation built by recursively subdividing rectangles into four pieces (Fig. 6) [47]. The basic concept of a quadtree spline is to define a continuous function over a quadtree domain by interpolating numeric values at the corners of each spline leaf cell (square). However, because cells are non-uniformly subdivided, *cracks* or first-order discontinuities in the interpolated function may arise [47]. Fig. 6b shows an illustration of this, where each vertex in the a quadtree spline is allowed to move independently, thereby causing tears in the overall motion field.

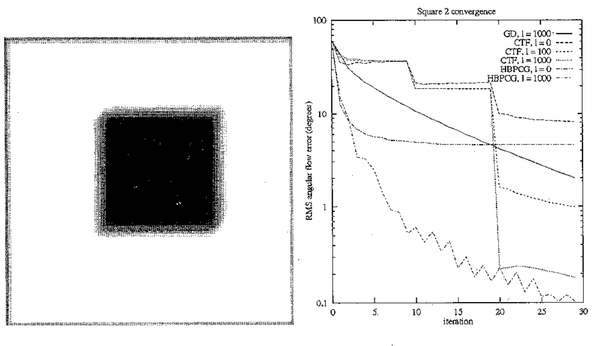


Fig. 5. **Square 2** sample image and convergence plot: (a) input image, (b) convergence plot (error vs. iteration number). The jumps in energy occur at changes in level in the coarse-to-fine (CTF) algorithm.

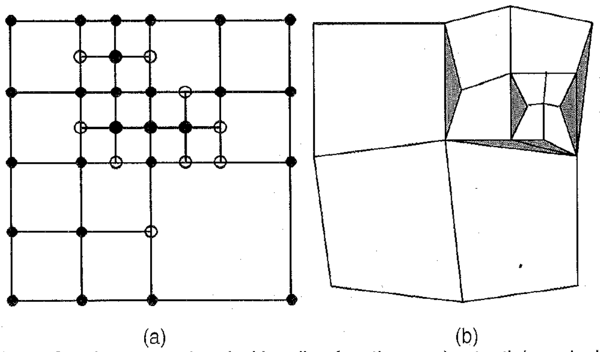


Fig. 6. Quadtree associated with spline function, and potential cracks in quadtree spline: (a) the nodes with filled circles (•) are free variables in the associated restricted hierarchical basis (Fig. 7), whereas the open circles (∘) must be interpolated from their ancestors; (b) potential cracks in a simpler quadtree spline are shown as shaded areas. If each vertex is allowed to move independently, discontinuities can appear in the motion field.

Several *crack-filling* strategies have been proposed [47]. The simplest strategy is to simply replace the values at the nodes along a crack edge (the white circles in Fig. 6) with the average values of its two parent nodes along the edge. This is the strategy we used in developing *octree splines* for the representation of multi-resolution distance maps in 3D pose estimation problems [28].

When the problem is one of iteratively *estimating* the values on the nodes in the quadtree spline, enforcing the crack-filling rule becomes more complicated. A useful strategy, which we developed for estimating 3D displacement fields in elastic medical image registration [58], is to use a hierarchical basis and to selectively zero out nodes in this basis. Fig. 7 shows the nodes in a hierarchical basis representation, where some nodes have been colored black (•) and others white (∘). When we set the values of the white nodes to zero in the hierarchical basis and then recompute

4. The errors are measured in degrees, as in [6].

the nodal basis using **S**, the resulting spline has the desired *continuity*, i.e., nodes along longer edges are the averages of their parents.

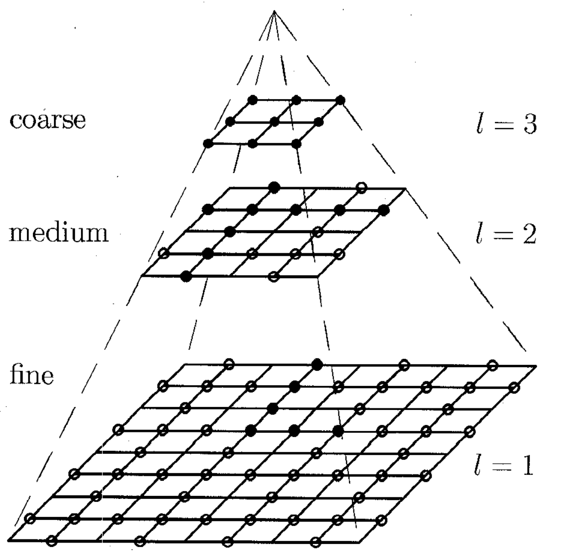


Fig. 7. Restricted (adaptive) hierarchical basis representation. The circles indicate the nodes in the hierarchical basis. Filled circles (•) are free variables in the quadtree spline, while open circles (○) must be zero (see Fig. 6).

The formulation of the quadtree spline in terms of an *adaptive hierarchical basis*, i.e., a basis in which some nodes are set to zero, has several advantages. First, it is very easy to implement, simply requiring a selective zeroing step between the $\mathbf{S}^T$ and **S** operations (algebraically, we write $\tilde{\mathbf{g}} = \mathbf{SZS}^T\mathbf{g}$, where **Z** is a diagonal matrix with 1s and 0s on the diagonal—see Fig. 4).[5] Second, it generalizes to splines of arbitrary order, e.g., we can build a C^1 quadtree spline based on quadratic B-splines using adaptive hierarchical basis functions. However, for higher-order splines, even more nodes have to be zeroed in order to ensure that finer level splines do not affect nearby coarser (undivided) cells. Third, as we will discuss in the next section, the adaptive hierarchical basis idea is even more general than the quadtree spline, and corresponds to a specific kind of multiresolution prior model.

The quadtree spline as described here ensures that the function within any leaf cell (square domain) has a simple form (single polynomial description, no spurious ripples). An alternative way of interpreting the quadtree in Fig. 6 is that it specifies the *minimum* degree of complexity in each cell, i.e., that each square is guaranteed to have its full degrees of freedom (e.g., all 4 corners have independent values in the bilinear case). In this latter interpretation, the open circles in the hierarchical basis are not zeroed, and only the circles actually not drawn in Fig. 6 are zeroed. In this approach, large squares can have arbitrarily-detailed ripples inside their domain resulting from fine-level basis functions near the square's boundaries. To date, we have not investigated this alternative possibility.

6.1 Subdivision Strategy

The quadtree spline provides a convenient way to use adaptively-sized patches for motion estimation, while maintaining inter-patch continuity. The question remains how to actually determine the topology of the patches, i.e., which patches get subdivided and which ones remain large. Ideally, we would like each patch to cover a region of the image within which the parametric motion model is valid. In a real-world situation, this may correspond to planar surface patches undergoing rigid motion with a small amount of perspective distortion (bilinear flow is then very close to projective flow). However, usually we are not *a priori* given the required segmentation of the image. Instead, we must deduce such a segmentation based on the adequacy of the flow model within each patch.

The fundamental tool we will use here is the concept of *residual flow* [25], recently used by Müller et al. [35] to subdivide affine motion patches (which they call *tiles*). The residual flow is the per-pixel estimate of flow required to register the two images in addition to the flow currently being modeled by the parametric motion model. At a single pixel, only the normal flow can be estimated,

$$\mathbf{u}_i^{\mathrm{N}} = \frac{\left(e_i G_i^x, e_i G_i^y\right)}{\left\|\left(G_i^x, G_i^y\right)\right\| + \epsilon} \tag{12}$$

where the intensity error e_i and the gradient $\nabla I_1 = \left(G_i^x, G_i^y\right)$ are given in (6) and (7). This measure is different from that used in [25], [35], who sum the numerator and denominator in (12) over a small neighborhood around each pixel.

To decide whether to split a spline patch into four smaller patches, we sum the magnitude of the residual normal flow $\left\|\mathbf{u}_i^{\mathrm{N}}\right\|$ over all the pixels in the patch and compare it to a threshold θ_{u}. In fact, we use a p-norm, $\left(\Sigma_i \left\|\mathbf{u}_i^{\mathrm{N}}\right\|^p\right)^{1/p}$, which can model a max operation as $p \to \infty$. Patches where the motion model is adequate should fall below this threshold, while patches which have multiple motions should be above. Starting with the whole image, we subdivide recursively until either the p-norm residual falls below an acceptable value or the smallest patch size considered (typically four to eight pixels wide) is reached. The actual patch size and number of levels are currently set by the user. Setting these automatically would make an interesting topic for future research.

Figs. 8a to 8c show an example of a quadtree spline motion estimate produced with this splitting technique for a simple synthetic example in which two central disks are independently moving against a textured background. The quadtree boundaries are warped to show the extent of the estimated image motion (up and left for the top disc, down and right for the bottom disc). Note how the subdivision occurs mostly at the object boundaries, as would be ex-

5. Whenever the **Z** matrix changes, we also have to recompute the quadtree spline using $\mathbf{u} \leftarrow \mathbf{SZS}^{-1}\mathbf{u}$. The $\mathbf{S}^{-1}$ procedure is similar to $\mathbf{S}^T$, but now $\hat{\mathbf{u}}_j \leftarrow \hat{\mathbf{u}}_j - \tilde{w}_{jk}\hat{\mathbf{u}}_k$.

pected. The most visible error (near the upper right edge of the lower disc) occurs in an area of little image contrast and where the motion is mostly parallel to the region contour.

An alternative to the iterative splitting strategy is to start with small patches and to then *merge* adjacent patches with compatible motion estimates into larger patches (within the constraints of allowable quadtree topologies). To test if a larger patch has consistent flow, we compare the four values along the edge of the patch and the value at the center with the average values interpolated from the four corner cells (look at the lower left quadrant of Fig. 6a to visualize this). The relative difference between the estimated and interpolated values,

$$d = \frac{\left\|\hat{\mathbf{u}}_j - \overline{\mathbf{u}}_j\right\|}{\sqrt{\left\|\hat{\mathbf{u}}_j\right\|^2 + \left\|\overline{\mathbf{u}}_j\right\|^2}} < \theta_d$$

where $\overline{\mathbf{u}}_j$ is the interpolated value, must be below a threshold θ_d (typically 0.25-0.5) for all five nodes before the four constituent patches are allowed to be merged into a larger patch. Notice that the quantity $\hat{\mathbf{u}}_j - \overline{\mathbf{u}}_j$ is exactly the value of the hierarchical basis function at a node (at least for bilinear splines), so we are in effect replacing small hierarchical basis values with zero values (this has a Bayesian interpretation, as we will discuss in the next section). Note also that this consistency criterion may fail in regions of little texture where the flow estimates are initially unreliable, unless regularization is applied to make these flow fields more smooth.

Fig. 8d shows an example of a quadtree spline motion estimate produced with this merging technique. The results are qualitatively quite similar to the results obtained with the split technique, although more fragmentation (splitting) is evident. It should be noted that changing the threholds $\theta_{\mathbf{u}}$ and θ_d will result in more or less segmentation. At the moment, we do not have any automatic way to set these thresholds.

7 A Bayesian Interpretation

The connection between energy-based or regularized low-level vision problems and Bayesian estimation formulations is well known [27], [33], [52]. In a nutshell, it can be shown that the energy or cost function being minimized can be converted into a probability distribution over the unknowns using a Gibbs or Boltzmann distribution, and that finding the minimum energy solution is equivalent to *maximum a posteriori* (MAP) estimation. The Bayesian model nicely decomposes the energy function into a measurement model (typically the squared error terms between the measurements and their predicted values) and a prior model (which usually corresponds to the stabilizer or smoothing term), i.e.,

$$\begin{aligned} &\mathcal{E}(\mathbf{u}) = \mathcal{E}_d(\mathbf{u},\mathbf{d}) + \mathcal{E}_p(\mathbf{u}) \Leftrightarrow \\ &p(\mathbf{u}) \propto e^{-\mathcal{E}(\mathbf{u})} = e^{-\mathcal{E}_d(\mathbf{u},\mathbf{d})}e^{-\mathcal{E}_p(\mathbf{u})} \propto p(\mathbf{u},\mathbf{d})p(\mathbf{u}) \end{aligned} \tag{13}$$

It then becomes straightforward to make use of robust statistical models by simply modifying the appropriate energy terms [8], [9].

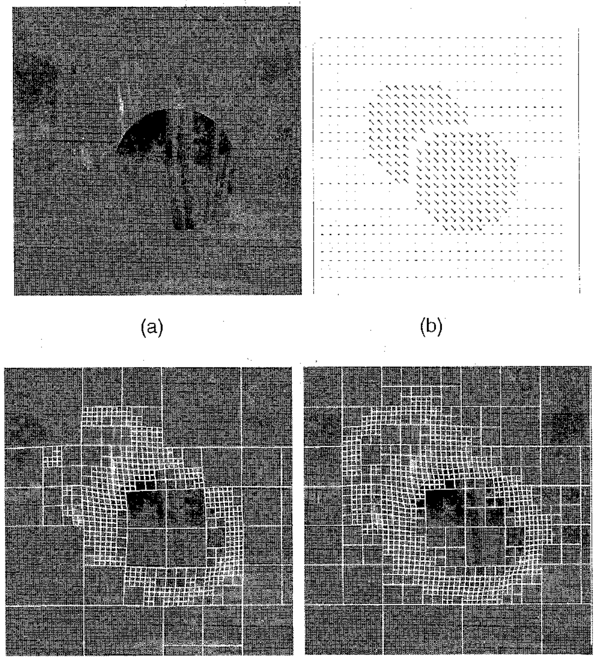

(a) (b)
(c) (d)

Fig. 8. Quadtree spline motion estimation (**Two Discs (SRI Trees)** sequence): (a) input image, (b) true flow, (c) split technique, (d) merge technique.

The basic spline-based flow model introduced in [55] is already a valid prior model, since it restricts the family of functions to the smooth set of tensor-product splines. In most cases, a small amount of intensity variation inside each spline patch is sufficient to ensure that a unique, well-behaved solution exists. It is also easy to add a small amount of regularization with quadratic penalty terms on the $\hat{\mathbf{u}}_j$s and their finite differences to handle possibly featureless areas in the image.

Hierarchical basis splines, as well as other multilevel representations such as overcomplete pyramids can be viewed as multiresolution priors [59]. There are two basic approaches to specifying such a prior. The first, which we use in our current work, is to simply view the hierarchical basis as a preconditioner, and to define the prior model over the usual nodal basis [53]. The alternative is to define the prior model directly on the hierarchical basis, usually assuming that each basis element is statistically independent from the others (i.e., that the covariance matrix is diagonal) [59], [42]. An extreme example of this is the *scale-recursive* multiscale Markov Random Fields introduced in [14], whose special structure makes it possible to recover the field in a single sweep through the pyramid. Unfortunately, their technique is based on a piecewise-constant model of flow, which results in recovered fields that have excessive "blockiness" [31].

Within this framework, adaptive hierarchical basis splines can be viewed as having a more complex multiresolution prior where each hierarchical node has a nonzero prior probability of being exactly zero. The split and merge algorithms can be viewed as simple heuristic techniques designed to recover the underlying motion field and to decide which nodes are actually zero. More sophisticated techniques to solve this problem would include simulated annealing [33] and mean-field annealing [20].

Quadtree splines have an even more complicated prior model, since the existence of zeros at certain levels in the pyramid implies zeros at lower levels as well as zeros at some neighboring nodes (depending on the exact interpretation of the quadtree spline). We will not pursue these models further in this paper, and leave their investigation to future work.

8 Experimental Results

To investigate the performance of our quadtree spline-based motion estimator, we use the synthetically generated **Two Discs (SRI Trees)** sequence shown in Fig. 8, for which we know the true motion (Fig. 8b). The results of our spline-based motion estimator for various choices of window size s, as well as the results with both the split and merge techniques, are shown in Table 1. The experiments show that the optimal fixed window size is $s = 8$, and that both split and merge techniques provide slightly better results. The relatively small difference in error between the various techniques is due to most of the error being concentrated in the regions where occlusions occur (Fig. 9). Adding an occlusion detection process to our algorithm should help reduce the errors in these regions.

TABLE 1
Summary of Two Discs (SRI Trees) Results

Technique	Pixel Error	Std. Dev.	Avg. Ang. Error	Std. Dev.	Density
regular spline ($s = 16$)	0.95	1.71	12.41°	18.95°	100%
regular spline ($s = 8$)	0.89	1.64	11.78°	17.75°	100%
regular spline ($s = 4$)	0.95	1.68	14.81°	17.80°	100%
quadtree spline (merge, $s = 4$)	0.85	1.58	11.04°	16.66°	100%
quadtree spline (split, $s = 4$)	0.95	1.63	14.41°	18.11°	100%

We also tested our algorithm on some of the standard motion sequences used in other recent motion estimation papers [6], [62], [41]. For each of these sequences, the same global parameters were used as in [56], e.g., 10 iterations per level, three levels in the coarse-to-fine or hierarchical basis pyramid, etc. The computation times on a DEC 3000 Model 500 AXP for these algorithms were all under 30 seconds.

The results on the **Hamburg Taxi** sequence are shown in Fig. 10, where the independent motion of the three moving cars can be clearly distinguished. Notice that the algorithm was also able to pick out the small region of the moving pedestrian near the upper left corner.

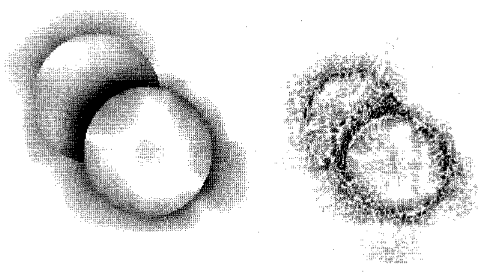


Fig. 9. Flow error $\|\mathbf{u} - \mathbf{u}^*\|$ and residual normal flow $\left\|\mathbf{u}_i^N\right\|$ for **Two Discs (SRI Trees)** sequence. Note how most of the errors are concentrated near the motion discontinuities and especially the disoccluded region in the center.

The results on the **Flower Garden** sequence are shown in Fig. 11. Here, the independent motion of the trunk of the tree has clearly been separated from the rest of the scene. The top of the flower garden, on the other hand, is not clearly separated from the house and sky, since it appears that the C^0 continuous motion field represented by the splines is an adequate description.[6]

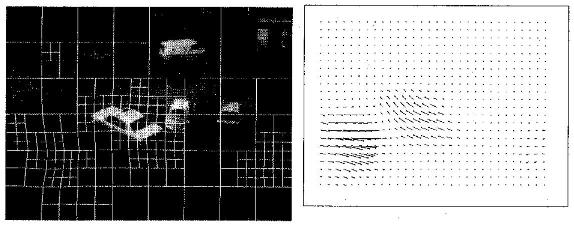


Fig. 10. **Hamburg Taxi** sequence: estimated quadtree and estimated flow, merging $s = 8$ patches in a 3-level pyramid.

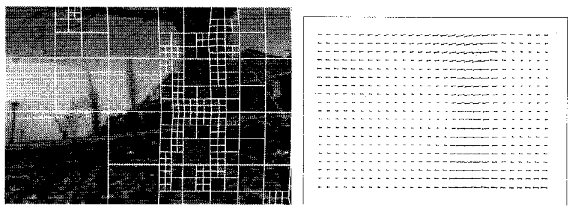


Fig. 11. **Flower Garden** sequence: estimated quadtree and estimated flow, merging $s = 4$ patches in a 4-level pyramid.

The results on the **Yosemite** sequence are shown in Fig. 12, and the tabulated error figures are shown in Table 2. As can be seen from these numbers, the results of using the quadtree spline are only slightly better than the already

6. Unlike the global motion estimates used in [62], we do not require that the motion be a combination of a few global affine motions.

very good results obtained with our regular spline-based motion estimator. As expected, the patch sizes are smaller in the vicinity of the motion discontinuities (e.g., the top edge of foreground peak).

TABLE 2
SUMMARY OF YOSEMITE RESULTS

Technique	Average Error	Standard Deviation	Density
Lucas and Kanade ($\lambda_2 \geq 5.0$)	3.22°	8.92°	8.7%
Fleet and Jepson ($\tau = 1.25$)	5.28°	14.34°	30.6%
local flow ($s = 16$, $T_e = 3000$)	2.20°	5.87°	23.1%
local flow ($s = 16$, $T_e = 2000$)	3.09°	7.59°	39.6%
merged flow (merge, $s = 16$)	3.00°	7.08°	39.4%

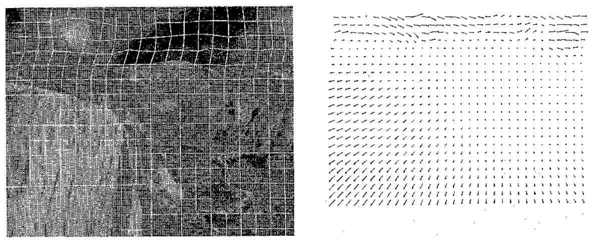


Fig. 12. **Yosemite** sequence: estimated quadtree and estimated flow, merging $s = 8$ patches in a 3-level pyramid.

The final sequence which we studied is the table of marble blocks acquired by Michael Otte [41] (Fig. 13). In this scene, the camera is moving forward and left while all of the blocks are stationary, except for the short central block, which is independently moving to the left. The quadtree segmentation of the motion field has separated out the tall block in the foreground and the independently moving block, but has not separated the other blocks from the table or the checkered background. Changing the thresholds on the merge algorithm could be used to achieve a greater segmentation, but this does not appear to be necessary to adequately model the motion field.

9 EXTENSIONS

We have recently extended the algorithm described in this paper in a number of directions, which include better multiframe flow estimation, parallel feature tracking, and local search.

When given more than two frames, we must assume a model of motion coherency across frames to take advantage of the additional information available. The simplest assumption is that of *linear flow*, i.e., that displacements between successive images and a base image are known scalar multiples of each other, $\mathbf{u}_t = s_t\mathbf{u}_1$.[7] Flow estimation can then be formulated by summing the intensity differences between the base frame and all other frames [55], which is similar to the *sum of sum of squared-distance* (SSSD) algorithm of [39]. We have found that in practice this works well, although it is often necessary to *bootstrap* the motion estimate by first computing motion estimates with fewer frames (this is because gradient descent gets trapped in local minima when the inter-frame displacements become large).

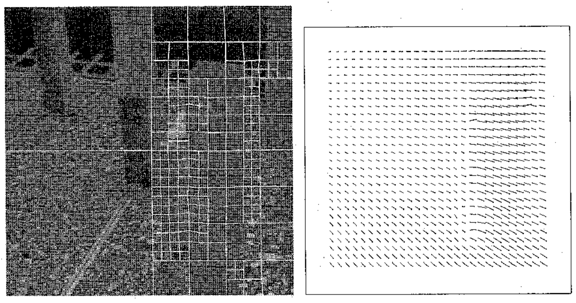


Fig. 13. **Michael Otte's** sequence: estimated quadtree and estimated flow, merging $s = 8$ patches in a 4-level pyramid.

When the motion is not linear, i.e., we have a non-zero acceleration, we cannot perform a single batch optimization. Instead, we can compute a separate flow field between each pair of images, using the previous flow as an initial guess. Alternatively, we can compute the motion between a base image and each successive image, using a linear predictor $\mathbf{u}_t = \mathbf{u}_{t-1} + (\mathbf{u}_{t-1} - \mathbf{u}_{t-2})$. This latter approach is useful if we are trying to track feature points without the problem of *drift* (accumulated error) which can occur if we just use inter-frame flows.

The linearly predicted multiframe motion estimator forms the basis of our parallel extended image sequence feature tracker [57]. To separate locations in the image where features are being tracked reliably from uninformative or confusing regions, we use a combination of the local Hessian estimate (9) and the local intensity error within each spline patch. This is similar to Shi and Tomasi's tracker [48], except that we use bilinear patches stitched together by the spline motion model, which yields better stability than isolated affine patches.

To deal with the local minima which can trap our gradient descent technique, we have also added an exhaustive search component to our algorithm. At the beginning of each set of iterations, e.g., after inter-level transfers in the coarse to fine algorithm, or after splitting in the quadtree spline estimator, we search around the current (u, v) estimate by trying a discrete set of nearby (u, v) values (as in SSD algorithms [3]). However, because we must maintain spline continuity, we cannot make the selection of best motion estimate for each patch independently. Instead, we average the motion estimates of neighboring patches to determine the motion of each spline control vertex.

7. In the most common case, e.g., for spatiotemporal filtering, a uniform temporal sampling ($s_t = t$) is assumed, but this is not strictly necessary.

In future work, we plan to extend our algorithm to handle occlusions in order to improve the accuracy of the flow estimates. The first part, which is simpler to implement, is to simply detect *foldovers*, i.e., when one region occludes another due to faster motion, and to disable error contributions from the occluded background. The second part would be to add an explicit occlusion model, which is not as straightforward because our splines are currently C^0 continuous. However, previous work on piecewise-continuous hierarchical basis functions could be used [53]. Additional possibilities include using our method as a robust way to bootstrap layered motion models, and applying our technique to stereo matching problems.

10 Discussion and Conclusions

The quadtree-spline motion algorithm we have developed provides a novel way of computing an accurate motion estimate while performing an initial segmentation of the motion field. Our approach optimizes the same stability versus detail tradeoff as adaptively-sized correlation windows, without incurring the large computational cost of overlapping windows and trial-and-error window size adjustment. Compared to the recursively split affine patch tracker of [35], our technique provides a higher level of continuity in the motion field, which should lead to more accurate motion estimates.

In summary, quadtree splines provide a novel and useful solution to the problem of describing a motion field at a number of resolution levels while simultaneously optimizing the tradeoff between estimate stability and detail. The ideas behind quadtree splines and adaptive hierarchical basis functions are very general, and can be applied to almost any domain where splines and finite element methods are currently used, including computer graphics and numerical relaxation problems. Another promising application is motion field estimation and compression for video coding, where our technique would combine wavelet-based coding of the motion field with run-length coding for the zero wavelets. Octree-splines have already been applied successfully to the elastic registration of 3D medical images, and we plan to extend our approach to other applications.

Acknowledgments

This work was performed while the authors were employed at Digital Equipment Corporation, Cambridge Research Lab. We would like to thank John Wang and Michael Otte for providing us with the motion sequences used in their research [62], [41], and the three anonymous reviewers for their detailed and insightful suggestions.

References

[1] E.H. Adelson and J.R. Bergen, "Spatiotemporal Energy Models for the Perception of Motion," *J. Optical Soc. Am.*, vol. A2, no. 2, pp. 284–299, Feb. 1985.

[2] J.K. Aggarwal and N. Nandhakumar, "On the Computation of Motion from Sequences of Images—a Review," *Proc. IEEE*, vol. 76, no. 8, pp. 917–935, Aug. 1988.

[3] P. Anandan, "A Computational Framework and an Algorithm for the Measurement of Visual Motion," *Int'l J. Computer Vision*, vol. 2, no. 3, pp. 283–310, Jan. 1989.

[4] O. Axelsson and V.A. Barker, *Finite Element Solution of Boundary Value Problems: Theory and Computation*. Orlando, Fla.: Academic Press, 1984.

[5] S.T. Barnard and M.A. Fischler, "Computational Stereo," *Computing Surveys*, vol. 14, no. 4, pp. 553–572, Dec. 1982.

[6] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, "Performance of Optical Flow Techniques," *Int'l J. Computer Vision*, vol. 12, no. 1, pp. 43–77, Jan. 1994.

[7] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani, "Hierarchical Model-Based Motion Estimation," *Second European Conf. Computer Vision (ECCV'92)*, pp. 237–252, Santa Margherita Liguere, Italy, May 1992, Springer-Verlag.

[8] M.J. Black and P. Anandan, "A Framework for the Robust Estimation of Optic Flow," *Fourth Int'l Conf. Computer Vision (ICCV'93)*, pp. 231–236, Berlin, Germany, May 1993, IEEE CS Press.

[9] M.J. Black and A. Rangarajan, "The Outlier Process: Unifying Line Processes and Robust Statistics," *IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR'94)*, pp. 15–22, Seattle, Washington, June 1994, IEEE CS Press.

[10] M. Bober and J. Kittler, "Estimation of Complex Multimodal Motion: An Approach Based on Robust Statistics and Hough Transform," *British Machine Vision Conf.*, pp. 239–248, 1993, BMVA Press.

[11] R.C. Bolles, H.H. Baker, and D.H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *Int'l J. Computer Vision*, vol. 1, pp. 7–55, 1987.

[12] L.G. Brown, "A Survey of Image Registration Techniques," *Computing Surveys*, vol. 24, no. 4, pp. 325–376, Dec. 1992.

[13] P.J. Burt and E.H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Comm.*, vol. 31, no. 4, pp. 532–540, Apr. 1983.

[14] T.M. Chin, M.R. Luettgen, W.C. Karl, and A.S. Willsky, "An Estimation-Theoretic Perspective on Image Processing and the Calculation of Optic Flow," *Advances in Image Sequence Processing*. Kluwer, 1993.

[15] U.R. Dhond and J.K. Aggarwal, "Structure from Stereo—a Review," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1,489–1,510, Nov./Dec. 1989.

[16] W. Enkelmann, "Investigations of Multigrid Algorithms for Estimation of Optical Flow Fields in Image Sequences," *Computer Vision, Graphics, and Image Processing*, pp. 150–177, 1988.

[17] M. Etoh and Y. Shirai, "Segmentation and 2D Motion Estimation by Region Fragments," *Fourth Int'l Conf. Computer Vision (ICCV'93)*, pp. 192–199, Berlin, Germany, May 1993, IEEE CS Press.

[18] D. Fleet and A. Jepson, "Computation of Component Image Velocity from Local Phase Information," *Int'l J. Computer Vision*, vol. 5, pp. 77–104, 1990.

[19] C.-S. Fuh and P. Maragos, "Motion Displacement Estimation Using an Affine Model for Image Matching," *Optical Eng.*, vol. 30 no. 7, pp. 881–887, July 1991.

[20] D. Geiger and F. Girosi, "Mean Field Theory for Surface Reconstruction," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 401–412, May 1991.

[21] M.A. Gennert, "Brightness-Based Stereo Matching," *Second Int'l Conf. Computer Vision (ICCV'88)*, pp. 139–143, Tampa, Florida, Dec. 1988, IEEE CS Press.

[22] D.J. Heeger, "Optical Flow from Spatiotemporal Filters," *First Int'l Conf. Computer Vision (ICCV'87)*, pp. 181–190, London, June 1987, IEEE CS Press.

[23] E.C. Hildreth, "Computing the Velocity Field Along Contours," N.I. Badler and J.K. Tsotsos, eds., *Motion: Representation and Perception*. New York: North-Holland, 1986, pp. 121–127.

[24] B.K.P. Horn and B.G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[25] M. Irani, B. Rousso, and S. Peleg, "Detecting and Tracking Multiple Moving Objects Using Temporal Integration," *Second European Conf. Computer Vision (ECCV'92)*, pp. 282–287, Santa Margherita Liguere, Italy, May 1992, Springer-Verlag.

[26] A. Jepson and M.J. Black, "Mixture Models for Optical Flow Computation," *IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR'93)*, pp. 760–761, New York, June 1993.

[27] G. Kimeldorf and G. Wahba, "A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines," *Annals of Math. Statistics*, vol. 41, no. 2, pp. 495–502, 1970.

[28] S. Lavallée, R. Szeliski, and L. Brunie, "Matching 3-D Smooth Surfaces with Their 2-D Projections Using 3-D Distance Maps," *SPIE*, vol. 1,570, *Geometric Methods in Computer Vision*, pp. 322–336, San Diego, July 1991, Soc. Photo-Optical Instrumentation Eng.

[29] B.D. Lucas, *Generalized Image Matching by the Method of Differences*. PhD thesis, Carnegie Mellon Univ., July 1984.

[30] B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application in Stereo Vision," *Seventh Int'l Joint Conf. Artificial Intelligence (IJCAI-81)*, pp. 674–679, Vancouver, 1981.

[31] M.R. Luettgen, W.C. Karl, and A.S. Willsky, "Efficient Multiscale Regularization with Applications to the Computation of Optical Flow," *IEEE Trans. Image Processing*, vol. 3, no. 1, pp. 41–64, Jan. 1994.

[32] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, July 1989.

[33] J. Marroquin, S. Mitter, and T. Poggio, "Probabilistic Solution of Ill-Posed Problems in Computational Vision," *J. Am. Statistical Assoc.*, vol. 82, no. 397, pp. 76–89, Mar. 1987.

[34] L.H. Matthies, R. Szeliski, and T. Kanade, "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences," *Int'l J. Computer Vision*, vol. 3, pp. 209–236, 1989.

[35] J.R. Müller, P. Anandan, and J.R. Bergen, "Adaptive-Complexity Registration of Images," *IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR'94)*, pp. 953–957, Seattle, Washington, June 1994, IEEE CS Press.

[36] H.-H. Nagel, "On the Estimation of Optical Flow: Relations Between Different Approaches and Some New Results," *Artificial Intelligence*, vol. 33, pp. 299–324, 1987.

[37] S. Negahdaripour and C.H. Yu, "A Generalized Brightness Change Model for Computing Optical Flow," *Fourth Int'l Conf. Computer Vision (ICCV'93)*, pp. 2–11, Berlin, Germany, May 1993, IEEE CS Press.

[38] M. Okutomi and T. Kanade, "A Locally Adaptive Window for Signal Matching," *Int'l J. Computer Vision*, vol. 7, no. 2, pp. 143–162, Apr. 1992.

[39] M. Okutomi and T. Kanade, "A Multiple Baseline Stereo," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 353–363, Apr. 1993.

[40] M. Okutomi and T. Kanade, "A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 920–932, Sept. 1994.

[41] M. Otte and H.-H. Nagel, "Optical Flow Estimation: Advances and Comparisons," *Third European Conf. Computer Vision (ECCV'94)*, vol. 1, pp. 51–60, Stockholm, Sweden, May 1994, Springer-Verlag.

[42] A.P. Pentland, "Interpolation Using Wavelet Bases," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 4, pp. 410–414, Apr. 1994.

[43] T. Poggio, V. Torre, and C. Koch, "Computational Vision and Regularization Theory," *Nature*, vol. 317, no. 6,035, pp. 314–319, Sept. 26, 1985.

[44] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, England: Cambridge Univ. Press, 2nd ed., 1992.

[45] L.H. Quam, "Hierarchical Warp Stereo," *Image Understanding Workshop*, pp. 149–155, New Orleans, Louisiana, Dec. 1984, Science Applications International Corporation.

[46] J. Rehg and A. Witkin, "Visual Tracking with Deformation Models," *IEEE Int'l Conf. Robotics and Automation*, pp. 844–850, Sacramento, California, Apr. 1991, IEEE CS Press.

[47] H. Samet, *The Design and Analysis of Spatial Data Structures*. Reading, Mass.: Addison-Wesley, 1989.

[48] J. Shi and C. Tomasi, "Good Features to Track," *IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593–600, Seattle, Washington, June 1994, IEEE CS Press.

[49] T. Simchony, R. Chellappa, and Z. Lichtenstein, "Pyramid Implementation of Optimal-Step Conjugate-Search Algorithms for Some Low-Level Vision Problems," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1,408–1,425, Nov./Dec. 1989.

[50] E.P. Simoncelli, E.H. Adelson, and D.J. Heeger, "Probability Distributions of Optic Flow," *IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR'91)*, pp. 310–315, Maui, Hawaii, June 1991, IEEE CS Press.

[51] A. Singh, "An Estimation-Theoretic Framework for Image-Flow Computation," *Third Int'l Conf. Computer Vision (ICCV'90)*, pp. 168–177, Osaka, Japan, Dec. 1990, IEEE CS Press.

[52] R. Szeliski, *Bayesian Modeling of Uncertainty in Low-Level Vision*. Boston: Kluwer, 1989.

[53] R. Szeliski, "Fast Surface Interpolation Using Hierarchical Basis Functions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, pp. 513–528, June 1990.

[54] R. Szeliski, "Fast Shape from Shading," *CVGIP: Image Understanding*, vol. 53, no. 2, pp. 129–153, Mar. 1991.

[55] R. Szeliski and J. Coughlan, "Hierarchical Spline-Based Image Registration," *IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR'94)*, pp. 194–201, Seattle, Washington, June 1994, IEEE CS Press.

[56] R. Szeliski and J. Coughlan, "Hierarchical Spline-Based Image Registration," *Int'l J. Computer Vision*, to appear.

[57] R. Szeliski, S.B. Kang, and H.-Y. Shum, "A Parallel Feature Tracker for Extended Image Sequences," *IEEE Int'l Symp. Computer Vision*, pp. 241–246, Coral Gables, Florida, Nov. 1995.

[58] R. Szeliski and S.Lavallée, "Matching 3-D Anatomical Surfaces with Non-Rigid Deformations Using Octree-Splines," *Int'l J. Computer Vision*, vol. 18, no. 2, pp. 171–186, May 1996.

[59] R. Szeliski and D. Terzopoulos, "Parallel Multigrid Algorithms and Computer Vision Applications," *Fourth Copper Mountain Conf. Multigrid Methods*, pp. 383–398, Copper Mountain, Colorado, Apr. 1989, Soc. for Industrial and Applied Math.

[60] D. Terzopoulos, "Image Analysis Using Multigrid Relaxation Methods," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 129–139, Mar. 1986.

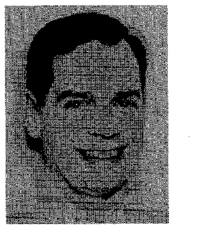
[61] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams Under Orthography: A Factorization Method," *Int'l J. Computer Vision*, vol. 9, no. 2, pp. 137–154, Nov. 1992.

[62] J.Y.A. Wang and E.H. Adelson, "Layered Representation for Motion Analysis," *IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR'93)*, pp. 361–366, New York, June 1993.

[63] J. Weber and J. Malik, "Robust Computation of Optical Flow in a Multi-Scale Differential Framework," *Fourth Int'l Conf. Computer Vision (ICCV'93)*, pp. 12–20, Berlin, Germany, May 1993, IEEE CS Press.

[64] A. Witkin, D. Terzopoulos, and M. Kass, "Signal Matching Through Scale Space," *Int'l J. Computer Vision*, vol. 1, pp. 133–144, 1987.

[65] H. Yserentant, "On the Multi-Level Splitting of Finite Element Spaces," *Numerische Mathematik*, vol. 49, pp. 379–412, 1986.



Richard Szeliski received the BEng degree in honors electrical engineering from McGill University (Montreal) in 1979; the MA Sc degree in electrical engineering from the University of British Columbia (Vancouver) in 1981; and the PhD degree in computer science from Carnegie Mellon University (Pittsburgh, Pennsylvania) in 1988. He has worked at Bell-Northern Research (Montreal), Schlumberger Palo Alto Research (Palo Alto, California), the Artificial Intelligence Center of SRI International (Menlo Park, California), and the Cambridge Research Lab of Digital Equipment Corporation (Cambridge, Massachusetts). He is currently a senior researcher at Microsoft Corporation (Redmond, Washington), where he is pursuing research in 3D computer vision and geometric modeling. Dr. Szeliski has published more than 60 research papers in computer vision, computer graphics, medical imaging, neural nets, and parallel numerical algorithms, as well as the book *Bayesian Modeling of Uncertainty in Low-Level Vision*. He is a member of the ACM, IEEE, and Sigma Xi. He served as cochair of the SPIE Conferences on Geometric Methods in Computer Vision and is currently an associate editor of *Transactions on Pattern Analysis and Machine Intelligence*.

Heung-Yeung Shum received the MPhil in electrical and electronic engineering from the University of Hong Kong in 1991. He graduated from the robotics PhD program at Carnegie Mellon University in 1996. His research interests are computer vision, virtual reality modeling, multimedia, and human–computer interface. He is currently a researcher at Microsoft Corporation.