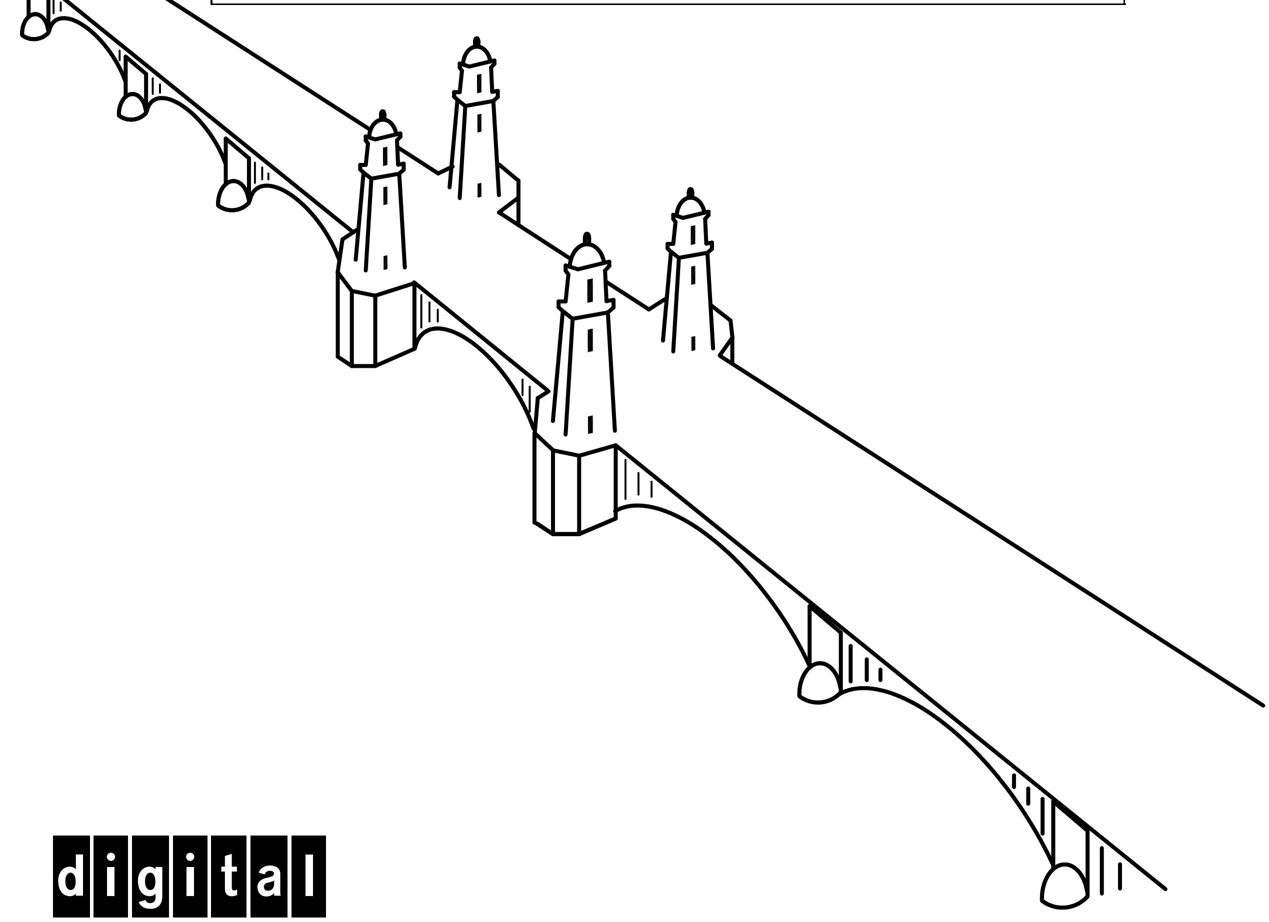
Motion Estimation with Quadtree Splines

Richard Szeliski and Heung-Yeung Shum

Digital Equipment Corporation
Cambridge Research Lab

CRL 95/1 March, 1995



digital

CAMBRIDGE RESEARCH LABORATORY
Technical Report Series

Cambridge Research Laboratory
One Kendall Square
Cambridge, Massachusetts 02139

Motion Estimation with
Quadtree Splines

Richard Szeliski and Heung-Yeung Shum[1]

Digital Equipment Corporation
Cambridge Research Lab

CRL 95/1 March, 1995

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.

Keywords: motion analysis, multiframe image analysis, hierarchical image registration, optical flow, splines, quadtree splines, adaptive subdivision.

[1]The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890

Contents

List of Figures

List of Tables

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 [Szeliski and Coughlan, 1994], 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 [Szeliski and Coughlan, 1994].

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. By examining the local certainty in the flow computation, we can also use our algorithm as a parallel feature tracker for very long motion sequences where image deformations may be significant [Szeliski *et al.*, 1995].

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 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 formulations 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.

2 Previous work

Motion estimation has long been one of the most actively studied areas of computer vision and image processing [Aggarwal and Nandhakumar, 1988; Brown, 1992]. 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 [Horn and Schunck, 1981]. 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* [Horn and Schunck, 1981; Lucas and Kanade, 1981; Nagel, 1987], correlation-based techniques such as the *sum of squared differences* (SSD) [Anandan, 1989], spatio-temporal filtering [Adelson and Bergen, 1985; Heeger, 1987; Fleet and Jepson, 1990; Weber and Malik, 1993], and regularization [Horn and Schunck, 1981; Hildreth, 1986; Poggio *et al.*, 1985]. Nagel [1987], Anandan [1989], and Otte and Nagel [1994] provide comparisons and derive relations between different techniques, while Barron *et al.* [1994] provide some numerical comparisons.

Global motion estimators [Lucas, 1984; Bergen *et al.*, 1992] 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.* [1992], 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* [Wang and Adelson, 1993; Jepson and Black, 1993; Etoh and Shirai, 1993; Bober and Kittler, 1993]. 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 [Müller *et al.*, 1994]. 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 [Barnard and Fischler, 1982; Quam, 1984; Dhond and Aggarwal, 1989] 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*) [Bolles *et al.*, 1987; Matthies *et al.*, 1989; Okutomi and Kanade, 1993]. The idea of using adaptive window sizes in stereo [Okutomi and Kanade, 1992; Okutomi and Kanade, 1994] 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 [Quam, 1984; Witkin *et al.*, 1987] and in motion estimation [Enkelmann, 1988; Anandan, 1989; Singh, 1990; Bergen *et al.*, 1992]. 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 computation efficiency and the ability to find better solutions by escaping from local minima.

The algorithm presented in this paper is also related to patch-based feature trackers [Lucas and Kanade, 1981; Rehg and Witkin, 1991; Tomasi and Kanade, 1992]. 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 [Szeliski *et al.*, 1995].

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”) 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

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

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

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

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

This equation is called the *sum of squared differences* (SSD) formula [Anandan, 1989]. Expanding I_1 in a first order Taylor series expansion in (u_i, v_i) yields the the *image brightness constraint* [Horn and Schunck, 1981]

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

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

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

where b and c are the (per-frame) brightness and contrast correction terms. 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 [Black and Anandan, 1993; Bober and Kittler, 1993; Black and Rangarajan, 1994]. 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 [Simoncelli *et al.*, 1991]. To further increase noise immunity, the intensity images used in (3) can be replaced by filtered images [Burt and Adelson, 1983].

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) [Anandan, 1989]. More recent variations use adaptive windows [Okutomi and Kanade, 1992] and multiple frames [Okutomi and Kanade, 1993]. Regularization-based algorithms add smoothness constraints on the u and v fields to obtain good solutions [Horn and Schunck, 1981; Hildreth, 1986; Poggio *et al.*, 1985]. 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.

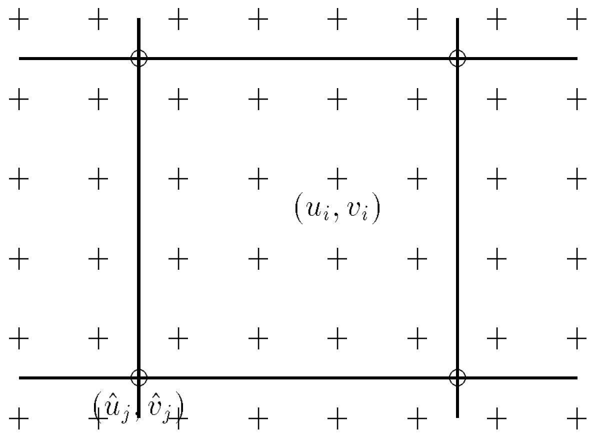


Figure 1: Displacement spline: the spline control vertices $\{(\hat{u}_j, \hat{v}_j)\}$ are shown as circles ($\circ$) and the pixel displacements $\{(u_i, v_i)\}$ are shown as pluses ($+$).

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 [Lucas, 1984; Bergen *et al.*, 1992; Szeliski and Coughlan, 1994]. 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 [Matthies *et al.*, 1989] and regularization [Szeliski, 1989], 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* (Figure 1). The value for the displacement at a pixel i can be written as

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

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 $(\hat{u}_j, \hat{v}_j)$.[2]

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 [Szeliski and Coughlan, 1994]. 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 [Press *et al.*, 1992]. First, we compute the gradient of E in (3) with respect to each of the parameters $\hat{u}_j$ and $\hat{v}_j$,

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

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,

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

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 [Szeliski and Coughlan, 1994].

[2] In the remainder of the paper, we will use indices i for pixels and j for spline control vertices.

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

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

The entries of $\mathbf{A}$ 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 [Press *et al.*, 1992]. 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 $\hat{\mathbf{A}} = \text{block\_diag}(\mathbf{A})$ 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*.[3] 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 [Szeliski and Coughlan, 1994] 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 3-point filter [Burt and Adelson, 1983]. 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).

[3] Preconditioning means adjusting the descent direction to accelerate the convergence, e.g., by pre-multiplying it by an approximate inverse of $\mathbf{A}$ [Axelsson and Barker, 1984; Press *et al.*, 1992].

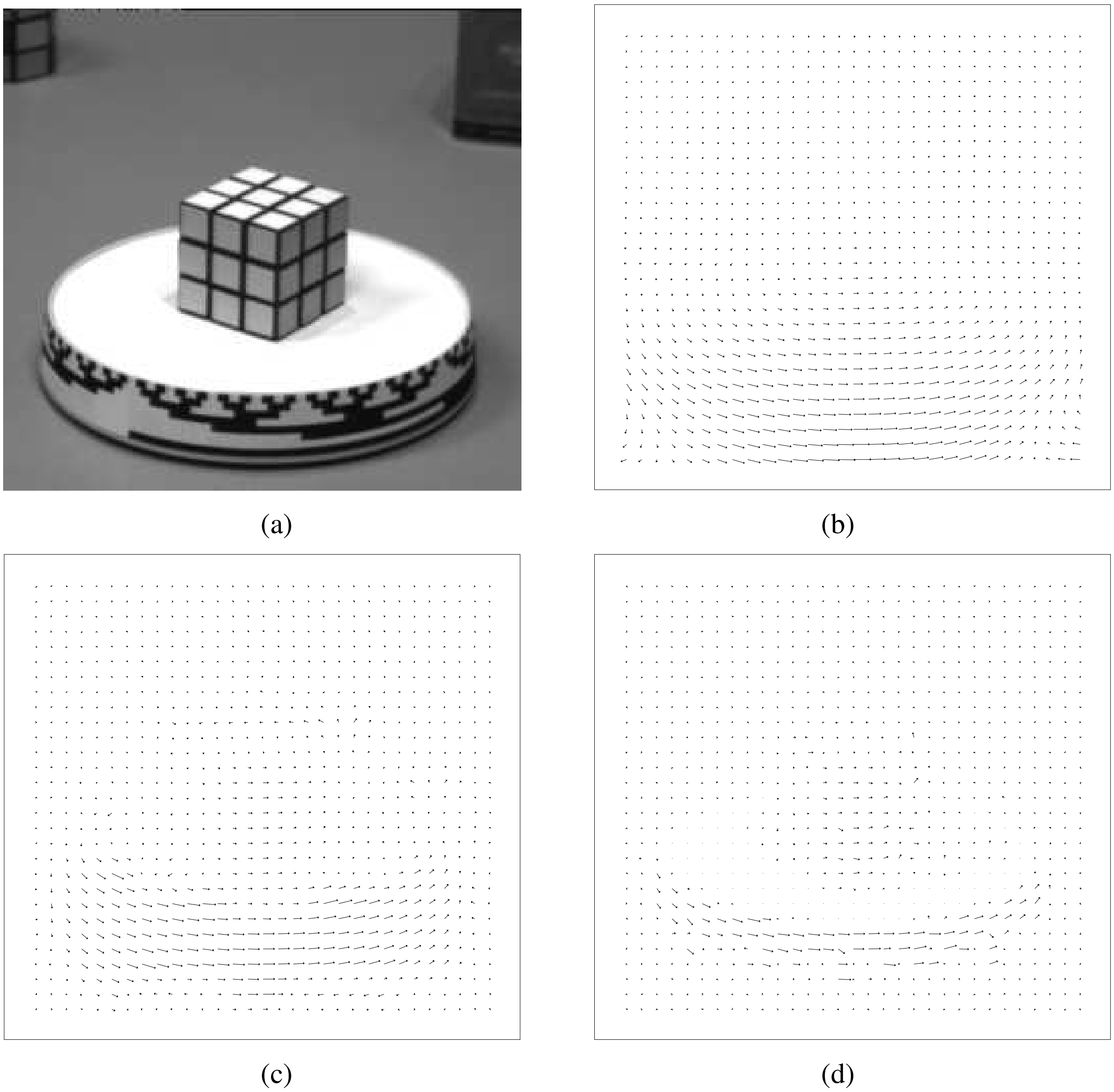


Figure 2: Example of general flow computation: (a) input image, (b)–(d) flow estimates for $m = 64$, 16, and 4.

Figure 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 3 level pyramid, 9 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.

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 [Quam, 1984; Anandan, 1989] can help, but often don't converge as quickly to the optimal solution as multigrid techniques [Terzopoulos, 1986]. Conjugate gradient descent can also be used, especially for non-linear problems such as shape-from-shading [Simchony *et al.*, 1989]. Perhaps the most effective technique is a combination of conjugate gradient descent with hierarchical basis functions [Yserentant, 1986], which has been applied both to interpolation problems in stereo matching [Szeliski, 1990] and to shape-from-shading [Szeliski, 1991].

Hierarchical basis functions are based on using a pyramidal representation for the data [Burt and Adelson, 1983], where the number of nodes in the pyramid is equal to the original number of nodes at the finest level (Figure 3). To convert from the hierarchical basis representation to the usual fine-level representation (which is called the *nodal basis* representation [Yserentant, 1986]), 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.[4] This process can be written algorithmically as

procedure S
  for $l = L - 1$ down to 1

[4]Hierarchical basis splines are therefore a degenerate (non-orthogonal) form of wavelets [Mallat, 1989] with extremely compact support and inverses.

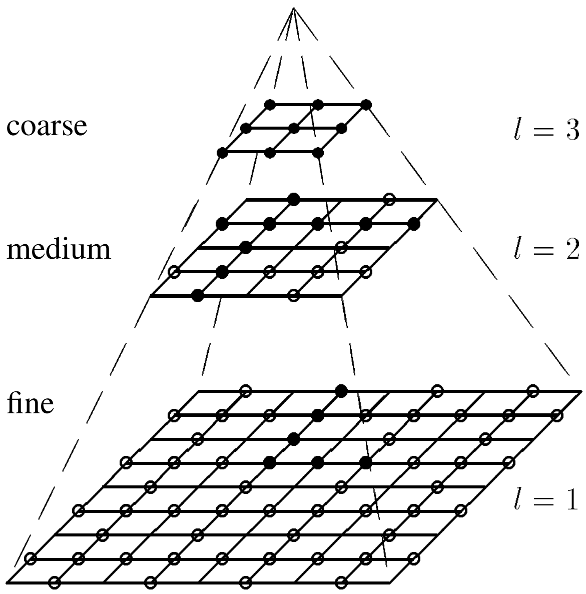


Figure 3: Multiresolution pyramid

The multiple resolution levels are a schematic representation of the hierarchical basis spline. The circles indicate the nodes in the hierarchical basis. Filled circles (•) are free variables in the quadtree spline (Section 6), while open circles (○) must be zero (see Figure 6).

---

for $j \in \mathcal{M}_l$

$\mathbf{u}_j^l = \hat{\mathbf{u}}_j^l + \sum_{k \in \mathcal{N}_j} \tilde{w}_{jk} \hat{\mathbf{u}}_k^{l+1}$

end S .

In this procedure, each node is assigned to one of the level collections $\mathcal{M}_l$ (the circles in Figure 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 [Szeliski, 1990].

We can write the above process algebraically as

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

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 $\tilde{\mathrm{u}}$ 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., $\tilde{\mathrm{g}} = \mathbf{SS}^T\mathrm{g}$ [Axelsson and Barker, 1984; Szeliski, 1990]. 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 $k \in \mathcal{M}_{l+1}$
      $\hat{\mathbf{u}}_k^{l+1} = \mathbf{u}_k^{l+1} + \sum_{j:k\in\mathcal{N}_j} \tilde{w}_{jk}\hat{\mathbf{u}}_j^l$
end S^T .

When combining hierarchical basis preconditioning with the block diagonal preconditioning in (10), we have several choices. We can apply the block diagonal preconditioning first, $\tilde{\mathrm{g}} = \mathbf{SS}^T\mathbf{B}^{-1}\mathrm{g}$, or second, $\tilde{\mathrm{g}} = \mathbf{B}^{-1}\mathbf{SS}^T\mathrm{g}$, or we can interleave the two preconditioners $\tilde{\mathrm{g}} = \mathbf{SB}^{-1}\mathbf{S}^T\mathrm{g}$, or $\tilde{\mathrm{g}} = \hat{\mathbf{B}}^{-T}\mathbf{SS}^T\hat{\mathbf{B}}\mathrm{g}$, 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.

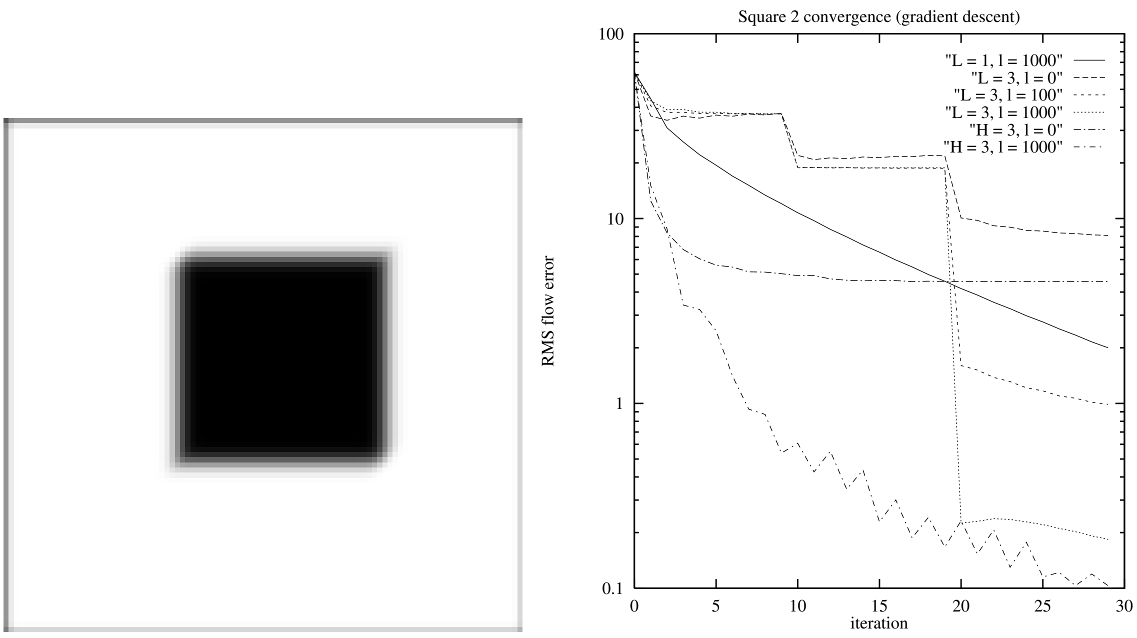
To summarize our algorithm (Figure 4), we keep both the hierarchical and nodal representations, and map between the two as required. For accumulating the distances and gradients required in (9), we compute the image flows and the derivatives with respect to the parameters in the nodal basis. We then use the hierarchical basis to smooth the residual vector 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.

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.* [1994].

$$
\begin{array}{llcl}
0. & \beta_0 & = & 0,\ \ \mathbf{d}_{-1} = 0 \\
1. & \mathbf{g}_n & = & -\nabla E(\mathbf{u}) \\
2.\dagger & \tilde{\mathbf{g}}_n & = & \mathbf{S}\mathbf{Z}\mathbf{S}^T\mathbf{B}^{-1}\mathbf{g}_n \\
3. & \beta_n & = & \tilde{\mathbf{g}}_n \cdot \mathbf{g}_n / \tilde{\mathbf{g}}_{n-1} \cdot \mathbf{g}_{n-1} \\
4. & \mathbf{d}_n & = & \tilde{\mathbf{g}}_n - \beta_n \mathbf{d}_{n-1} \\
5. & \alpha_n & = & \mathbf{d}_n \cdot \mathbf{g}_n / \mathbf{d}_n^T \mathbf{A} \mathbf{d}_n \\
6. & \mathbf{u}_{n+1} & = & \mathbf{u}_n + \alpha_n \mathbf{d}_n \\
7. & \multicolumn{3}{l}{\text{increment } n, \text{ loop to } 1.}
\end{array}
$$

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

Figure 4: Hierarchical basis preconditioned conjugate gradient algorithm



(a) (b)

(a) input image, (b) convergence plot (error vs. iteration number)

Figure 5: **Square 2** sample image and convergence plot

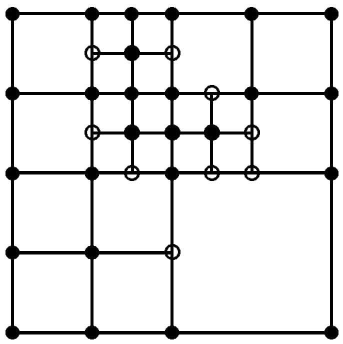
Figure 5a shows one image in the sequence, while Figure 5b shows the convergence rates for regular gradient descent ($L = 1$), coarse-to-fine estimation ($L = 3$), and preconditioning with hierarchical basis functions ($H = 3$), with different amounts of regularization ($\lambda_1 = 0, 100, 1000$). As 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

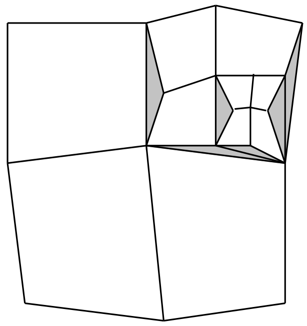
While hierarchical basis splines can help accelerate an estimation algorithm or even to add extra smoothness to the solution, they do not in themselves solve the problem of having adaptively-sized patches. For this, we will use the idea of *quadtree splines*, i.e., splines defined on a quadtree domain. A quadtree is a 2-D representation built by recursively subdividing rectangles into four pieces (Figure 6) [Samet, 1989]. 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 will arise (Figure 6b) unless a *crack-filling* strategy is used [Samet, 1989]. The simplest strategy is to simply replace the values at the nodes along a crack edge (the white circles in Figure 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 3-D pose estimation problems [Lavallée *et al.*, 1991].

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 3-D displacement fields in elastic medical image registration [Szeliski and Lavallée, 1994], is to use a hierarchical basis and to selectively zero out nodes in this basis. Observe that if in Figures 3 and 6a we set the values of the open circles ($\circ$) to zero in the hierarchical basis and then re-compute the nodal basis using $\mathbf{S}$, the resulting spline has the desired continuity, i.e., nodes along longer edges are the averages of their parents.

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,



(a)



(b)

Figure 6: Quadtree associated with spline function, and potential cracks in quadtree spline (a) the nodes with filled circles (•) are free variables in the associated hierarchical basis, whereas the open circles (○) (and also the nodes not drawn) must be zero (in the nodal basis, these nodes are interpolated from their ancestors); (b) potential cracks in a simpler quadtree spline are shown as shaded areas.

simply requiring a selective zeroing step between the $\mathbf{S}^T$ and $\mathbf{S}$ operations (algebraically, we write $\tilde{\mathbf{g}} = \mathbf{SZS}^T\mathbf{g}$, where $\mathbf{Z}$ is a diagonal matrix with 1's and 0's on the diagonal—see Figure 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 multi-resolution 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 Figure 6a 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 Figure 6a are zeroed. In this approach, large squares can have arbitrarily-detailed ripples inside their domain resulting

[5]Whenever the $\mathbf{Z}$ matrix changes, we also have to re-compute 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 $\hat{\mathbf{u}}_j \leftarrow \hat{\mathbf{u}}_j - \tilde{w}_{jk}\hat{\mathbf{u}}_k$.

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* [Irani *et al.*, 1992], recently used by Müller *et al.* [1994] 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,

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

where the intensity error e_i and the gradient $\nabla I_1 = (G_i^x, G_i^y)$ are given in (6–7). This measure is different from that used in [Irani *et al.*, 1992; Müller *et al.*, 1994], 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 $\|\mathbf{u}_i^{\mathrm{N}}\|$ over all the pixels in the patch and compare it to a threshold $\theta_{\mathbf{u}}$.[6] 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 4-8 pixels wide) is reached.

Figures 7a–c 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

[6] Actually, we use a $p-$norm, $(\sum_i \|\mathbf{u}_i^{\mathrm{N}}\|^p)^{1/p}$, which can model a max operation as $p \to \infty$.

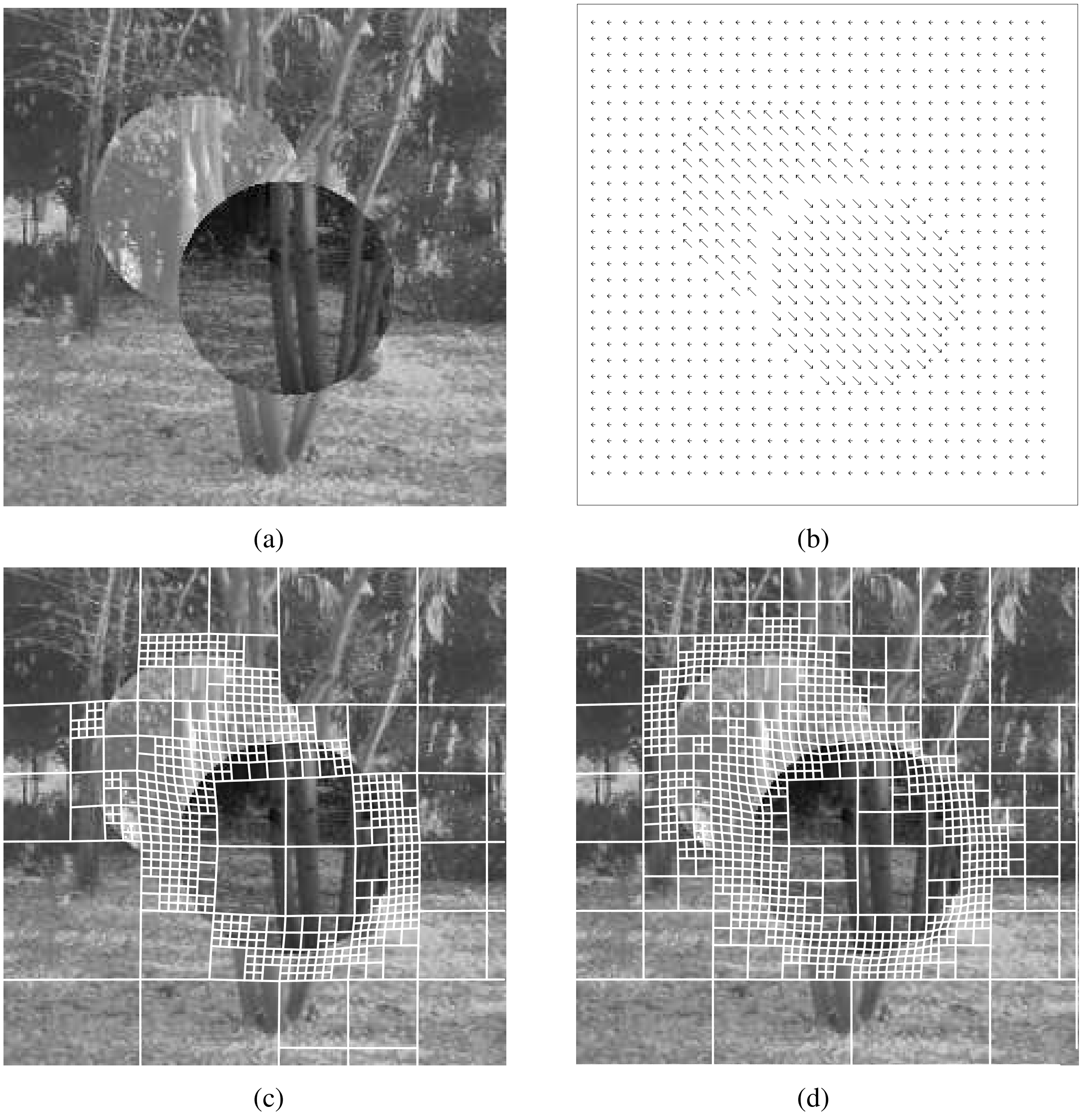


Figure 7: Quadtree spline motion estimation (**Two Discs (SRI Trees)** sequence): (a) input image, (b) true flow, (c) split technique, (d) merge technique.

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 expected. 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 Figure 6a to visualize this). The relative difference between the estimated and interpolated values,

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

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 converting small hierarchical basis values close to be exactly zero (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.

Figure 7d 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.

7 A Bayesian interpretation

The connection between energy-based or regularized low-level vision problems and Bayesian estimation formulations is well known [Kimeldorf and Wahba, 1970; Marroquin *et al.*, 1987; Szeliski, 1989]. 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.,

$$\mathcal{E}(\mathbf{u}) = \mathcal{E}_{\mathrm{d}}(\mathbf{u}, \mathbf{d}) + \mathcal{E}_{\mathrm{p}}(\mathbf{u}) \quad \Leftrightarrow \quad p(\mathbf{u}) \propto e^{-\mathcal{E}(\mathbf{u})} = e^{-\mathcal{E}_{\mathrm{d}}(\mathbf{u},\mathbf{d})}\, e^{-\mathcal{E}_{\mathrm{p}}(\mathbf{u})} \propto p(\mathbf{u}, \mathbf{d})\, p(\mathbf{u}) \tag{13}$$

It then becomes straightforward to make use of robust statistical models by simply modifying the appropriate energy terms [Black and Anandan, 1993; Black and Rangarajan, 1994].

The basic spline-based flow model introduced in [Szeliski and Coughlan, 1994] 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. However, just to be on the safe side, it is easy to add a small amount of regularization with quadratic penalty terms on the $\hat{\mathbf{u}}_j$'s and their finite differences.

Hierarchical basis splines, as well as other multilevel representations such as overcomplete pyramids can be viewed as multiresolution priors [Szeliski and Terzopoulos, 1989]. 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 [Szeliski, 1990]. 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) [Szeliski and Terzopoulos, 1989; Pentland, 1994]. An extreme example of this is the *scale-recursive* multiscale Markov Random Fields introduced in [Chin *et al.*, 1993], 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" [Luettgen *et al.*, 1994].

Within this framework, adaptive hierarchical basis splines can be viewed as having a more complex multiresolution prior where each hierarchical node has a non-zero 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 [Marroquin *et al.*, 1987] and mean-field annealing [Geiger and Girosi, 1991].

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 model further in this paper, and leave their investigation to future work.

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%

Table 1: Summary of **Two Discs (SRI Trees)** results

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 Figure 7, for which we know the true motion (Figure 7b). 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 is error between the various techniques is due to most of the error being concentrated in the regions where occlusions occur (Figure 8). Adding an occlusion detection process to our algorithm should help reduce the errors in these regions.

We also tested our algorithm on some of the standard motion sequences used in other recent motion estimation papers [Barron *et al.*, 1994; Wang and Adelson, 1993; Otte and Nagel, 1994]. The results on the **Hamburg Taxi** sequence are shown in Figure 9, 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.

The result on the **Flower Garden** sequence are shown in Figure 10. Here, the trunk of the tree is clearly segmented from the rest of the scene. The top of the flower garden, on the other hand, is not clearly segmented from the house and sky, since it appears that the C^0 continuous motion field represented by the splines is an adequate description.[7]

The final sequence which we studied is the table of marble blocks acquired by Michael Otte

[7] Unlike the global motion estimates used in [Wang and Adelson, 1993], we do not require that the motion be a combination of a few global affine motions.

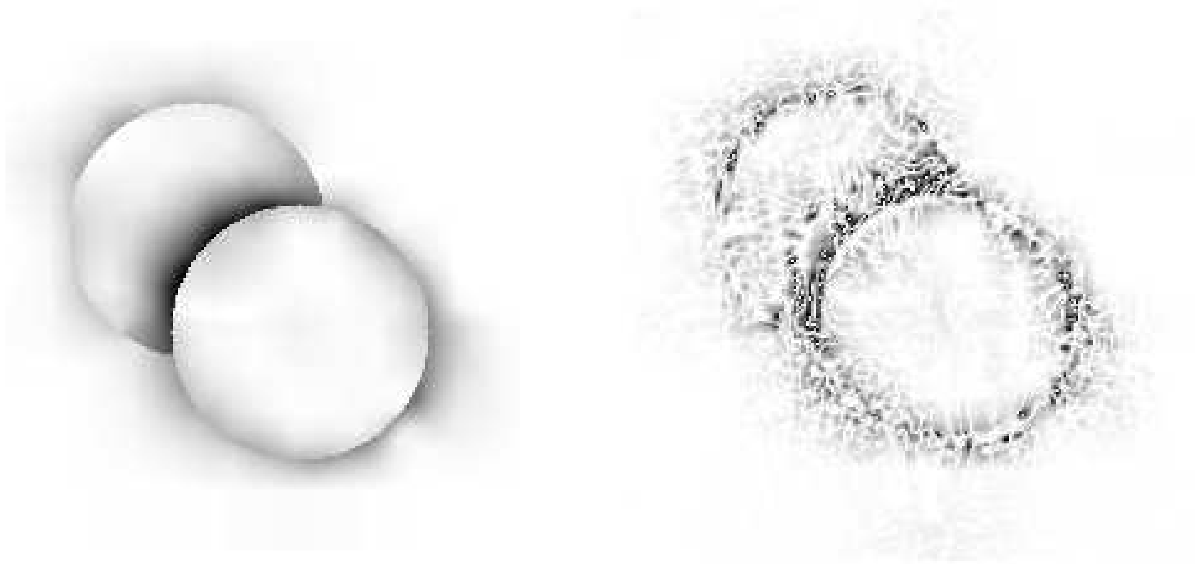


Figure 8: Flow error $\|\mathbf{u}-\mathbf{u}^*\|$ and residual normal flow $\|\mathbf{u}_i^{\mathrm{N}}\|$ for **Two Discs (SRI Trees)** sequence. Note how most of the errors are concentrated near the motion discontinuities and especially the disoccluded region in the center.

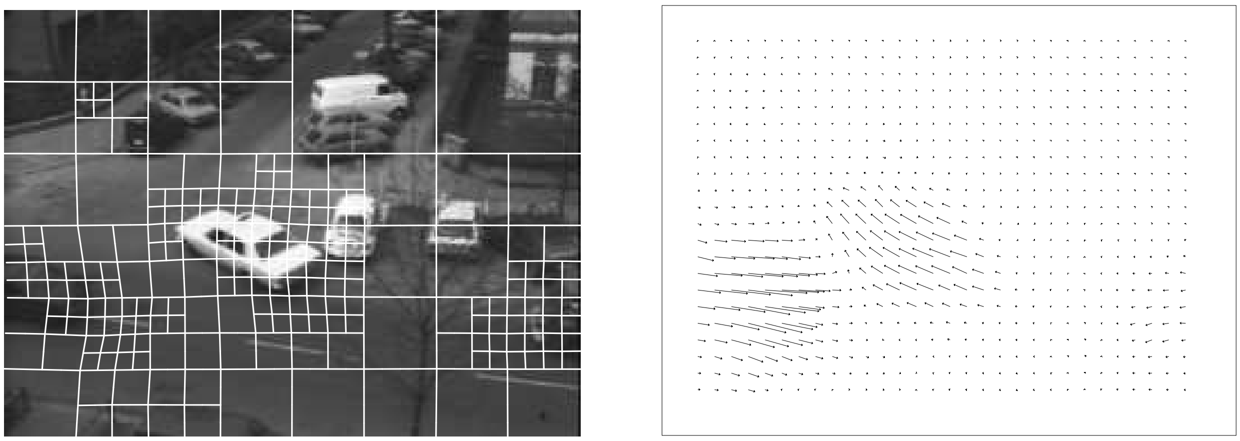


Figure 9: **Hamburg Taxi** sequence: estimated quadtree and estimated flow, merging $s = 8$ patches in a 3-level pyramid.

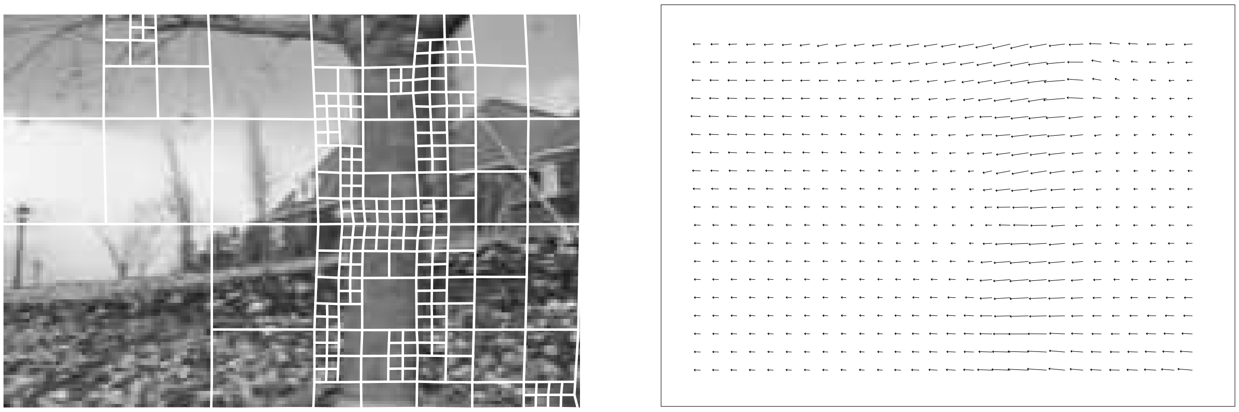


Figure 10: **Flower Garden** sequence: estimated quadtree and estimated flow, merging $s = 4$ patches in a 4-level pyramid.

[Otte and Nagel, 1994]. 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 are currently extending 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, $\mathbf{u}_t = s_t \mathbf{u}_1$.[8] Flow estimation can then be formulated by summing the intensity differences between the base frame and all other frames [Szeliski and Coughlan, 1994], which is similar to the *sum of sum of squared-distance* (SSSD) algorithm of [Okutomi and Kanade, 1993]. We have found that in practice this works well, although it is often necessary to *bootstrap* the motion

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

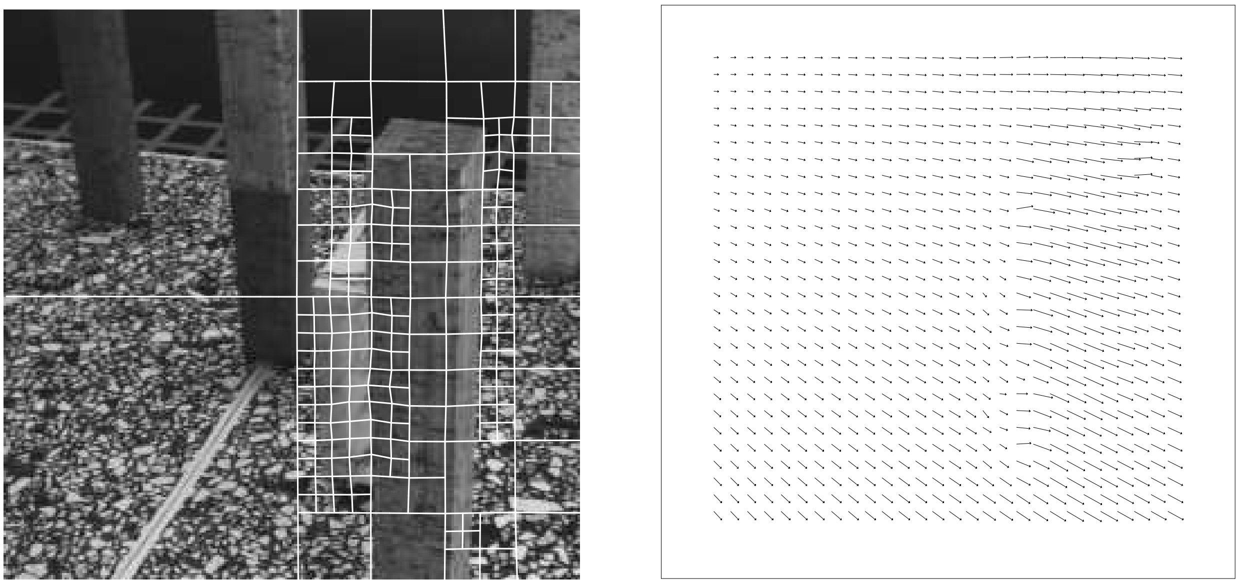


Figure 11: **Michael Otte's** sequence: estimated quadtree and estimated flow, merging $s = 8$ patches in a 4-level pyramid.

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).

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 [Szeliski *et al.*, 1995]. 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 [Shi and Tomasi, 1994], 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 are also adding 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 [Anandan, 1989]). 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.

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. In other work, we would also like to study the suitability of our method as a robust way to bootstrap layered motion models. We also plan to test our technique on standard stereo 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 [Müller *et al.*, 1994], our technique provides a higher level of continuity in the motion field, which leads to more accurate motion estimates.

The general framework of quadtree splines and hierarchical basis functions is equally applicable to other computer vision problems such as surface interpolation, as well as computer graphics and numerical relaxation problems. It has already been applied successfully to the elastic registration of 3D medical images [Szeliski and Lavallée, 1994], and we plan to extend our approach to other applications.

References

[Adelson and Bergen, 1985] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America*, A 2(2):284–299, February 1985.

[Aggarwal and Nandhakumar, 1988] J. K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images—a review. *Proceedings of the IEEE*, 76(8):917–935, August 1988.

[Anandan, 1989] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, January 1989.

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

[Barnard and Fischler, 1982] S. T. Barnard and M. A. Fischler. Computational stereo. *Computing Surveys*, 14(4):553–572, December 1982.

[Barron *et al.*, 1994] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, January 1994.

[Bergen *et al.*, 1992] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Second European Conference on Computer Vision (ECCV'92)*, pages 237–252, Springer-Verlag, Santa Margherita Liguere, Italy, May 1992.

[Black and Anandan, 1993] M. J. Black and P. Anandan. A framework for the robust estimation of optic flow. In *Fourth International Conference on Computer Vision (ICCV'93)*, pages 231–236, IEEE Computer Society Press, Berlin, Germany, May 1993.

[Black and Rangarajan, 1994] M. J. Black and A. Rangarajan. The outlier process: Unifying line processes and robust statistics. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 15–22, IEEE Computer Society, Seattle, Washington, June 1994.

[Bober and Kittler, 1993] M. Bober and J. Kittler. Estimation of complex multimodal motion: An approach based on robust statistics and hough transform. In *British Machine Vision Conference*, pages 239–248, BMVA Press, 1993.

[Bolles *et al.*, 1987] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55, 1987.

[Brown, 1992] L. G. Brown. A survey of image registration techniques. *Computing Surveys*, 24(4):325–376, December 1992.

[Burt and Adelson, 1983] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540, April 1983.

[Chin *et al.*, 1993] 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. In *Advances in Image Sequence Processing*, Kluwer, March 1993.

[Dhond and Aggarwal, 1989] U. R. Dhond and J. K. Aggarwal. Structure from stereo—a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510, November/December 1989.

[Enkelmann, 1988] W. Enkelmann. Investigations of multigrid algorithms for estimation of optical flow fields in image sequences. *Computer Vision, Graphics, and Image Processing*, :150–177, 1988.

[Etoh and Shirai, 1993] M. Etoh and Y. Shirai. Segmentation and 2D motion estimation by region fragments. In *Fourth International Conference on Computer Vision (ICCV'93)*, pages 192–199, IEEE Computer Society Press, Berlin, Germany, May 1993.

[Fleet and Jepson, 1990] D. Fleet and A. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–104, 1990.

[Geiger and Girosi, 1991] D. Geiger and F. Girosi. Mean field theory for surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(5):401–412, May 1991.

[Heeger, 1987] D. J. Heeger. Optical flow from spatiotemporal filters. In *First International Conference on Computer Vision (ICCV'87)*, pages 181–190, IEEE Computer Society Press, London, England, June 1987.

[Hildreth, 1986] E. C. Hildreth. Computing the velocity field along contours. In N. I. Badler and J. K. Tsotsos, editors, *Motion: Representation and Perception*, pages 121–127, North-Holland, New York, New York, 1986.

[Horn and Schunck, 1981] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[Irani *et al.*, 1992] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *Second European Conference on Computer Vision (ECCV'92)*, pages 282–287, Springer-Verlag, Santa Margherita Liguere, Italy, May 1992.

[Jepson and Black, 1993] A. Jepson and M. J. Black. Mixture models for optical flow computation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 760–761, New York, New York, June 1993.

[Kimeldorf and Wahba, 1970] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[Lavallée *et al.*, 1991] S. Lavallée, R. Szeliski, and L. Brunie. Matching 3-d smooth surfaces with their 2-d projections using 3-d distance maps. In *SPIE Vol. 1570 Geometric Methods in Computer Vision*, pages 322–336, Society of Photo-Optical Instrumentation Engineers, San Diego, July 1991.

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

[Lucas and Kanade, 1981] B. D. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In *Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pages 674–679, Vancouver, 1981.

[Luettgen *et al.*, 1994] M. R. Luettgen, W. C. Karl, and A. S. Willsky. Efficient multiscale regularization with applications to the computation of optical flow. *IEEE Transactions on Image Processing*, 3(1):41–64, January 1994.

[Mallat, 1989] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(7):674–693, July 1989.

[Marroquin *et al.*, 1987] J. Marroquin, S. Mitter, and T. Poggio. Probabilistic solution of ill-posed problems in computational vision. *Journal of the American Statistical Association*, 82(397):76–89, March 1987.

[Matthies *et al.*, 1989] L. H. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3:209–236, 1989.

[Müller *et al.*, 1994] J. R. Müller, P. Anandan, and J. R. Bergen. Adaptive-complexity registration of images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 953–957, IEEE Computer Society, Seattle, Washington, June 1994.

[Nagel, 1987] H.-H. Nagel. On the estimation of optical flow: Relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324, 1987.

[Okutomi and Kanade, 1992] M. Okutomi and T. Kanade. A locally adaptive window for signal matching. *International Journal of Computer Vision*, 7(2):143–162, April 1992.

[Okutomi and Kanade, 1993] M. Okutomi and T. Kanade. A multiple baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, April 1993.

[Okutomi and Kanade, 1994] M. Okutomi and T. Kanade. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, September 1994.

[Otte and Nagel, 1994] M. Otte and H.-H. Nagel. Optical flow estimation: advances and comparisons. In *Third European Conference on Computer Vision (ECCV'94)*, pages 51–60, Springer-Verlag, Stockholm, Sweden, May 1994.

[Pentland, 1994] A. P. Pentland. Interpolation using wavelet bases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):410–414, April 1994.

[Poggio *et al.*, 1985] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 26 September 1985.

[Press *et al.*, 1992] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, second edition, 1992.

[Quam, 1984] L. H. Quam. Hierarchical warp stereo. In *Image Understanding Workshop*, pages 149–155, Science Applications International Corporation, New Orleans, Louisiana, December 1984.

[Rehg and Witkin, 1991] J. Rehg and A. Witkin. Visual tracking with deformation models. In *IEEE International Conference on Robotics and Automation*, pages 844–850, IEEE Computer Society Press, Sacramento, California, April 1991.

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

[Shi and Tomasi, 1994] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600, IEEE Computer Society, Seattle, Washington, June 1994.

[Simchony *et al.*, 1989] T. Simchony, R. Chellappa, and Z. Lichtenstein. Pyramid implementation of optimal-step conjugate-search algorithms for some low-level vision problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-19(6):1408–1425, November/December 1989.

[Simoncelli *et al.*, 1991] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optic flow. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pages 310–315, IEEE Computer Society Press, Maui, Hawaii, June 1991.

[Singh, 1990] A. Singh. An estimation-theoretic framework for image-flow computation. In *Third International Conference on Computer Vision (ICCV'90)*, pages 168–177, IEEE Computer Society Press, Osaka, Japan, December 1990.

[Szeliski, 1989] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision*. Kluwer Academic Publishers, Boston, Massachusetts, 1989.

[Szeliski, 1990] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528, June 1990.

[Szeliski, 1991] R. Szeliski. Fast shape from shading. *CVGIP: Image Understanding*, 53(2):129–153, March 1991.

[Szeliski and Coughlan, 1994] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 194–201, IEEE Computer Society, Seattle, Washington, June 1994.

[Szeliski and Lavallée, 1994] R. Szeliski and S. Lavallée. Matching 3-D anatomical surfaces with non-rigid deformations using octree-splines. In *IEEE Workshop on Biomedical Image Analysis*, pages 144–153, IEEE Computer Society, Seattle, Washington, June 1994.

[Szeliski and Terzopoulos, 1989] R. Szeliski and D. Terzopoulos. Parallel multigrid algorithms and computer vision applications. In *Fourth Copper Mountain Conference on Multigrid Methods*, pages 383–398, Society for Industrial and Applied Mathematics, Copper Mountain, Colorado, April 1989.

[Szeliski *et al.*, 1995] R. Szeliski, S. B. Kang, and H.-Y. Shum. *A Parallel Feature Tracker for Extended Image Sequences*. Technical Report 95/2, Digital Equipment Corporation, Cambridge Research Lab, April 1995.

[Terzopoulos, 1986] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(2):129–139, March 1986.

[Tomasi and Kanade, 1992] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[Wang and Adelson, 1993] J. Y. A. Wang and E. H. Adelson. Layered representation for motion analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 361–366, New York, New York, June 1993.

[Weber and Malik, 1993] J. Weber and J. Malik. Robust computation of optical flow in a multi-scale differential framework. In *Fourth International Conference on Computer Vision (ICCV'93)*, pages 12–20, IEEE Computer Society Press, Berlin, Germany, May 1993.

[Witkin *et al.*, 1987] A. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. *International Journal of Computer Vision*, 1:133–144, 1987.

[Yserentant, 1986] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.

Acknowledgements

We would like to thank John Wang and Michael Otte for providing us with the motion sequences used in their research [Wang and Adelson, 1993; Otte and Nagel, 1994].