Motion Estimation with Quadtree Splines

Richard Szeliski
Digital Equipment Corporation
Cambridge Research Lab
One Kendall Square, Bldg. 700
Cambridge, MA 02139

Heung-Yeung Shum
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

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.

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 [18], 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.

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 [19].

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. Sections 6 and 7 present our novel quadtree splines and discuss how their shape can be estimated using both split and merge techniques. 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. A longer version of this paper is available as [21].

2 Previous work

Motion estimation has long been one of the most actively studied areas of computer vision and image processing [2]. The general motion estimation problem is often called *optical flow* recovery [8]. Approaches to this problem include

0-8186-7042-8/95 $4.00

gradient-based approaches based on the *brightness constraint* [8, 10, 12], correlation-based techniques such as the *sum of squared differences* (SSD) [3], spatio-temporal filtering [1], and regularization [8]. Nagel [12], Anandan [3] derive relations between different techniques, while Barron *et al.* [5] provide some numerical comparisons.

Global motion estimators [6] 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.* [6], 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 can be extended to more complex motions using a collection of parametric motion models. For example, each pixel can be associated with one of several global motion hypotheses, resulting in a *layered motion model* [22]. 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 [11]. Our approach is similar to this latter work, except that it preserves inter-patch motion continuity, and uses both split and merge techniques. Our approach is also similar in spirit to the adaptively sized windows used in stereo matching [13].

The algorithm presented in this paper is also related to patch-based feature trackers [10, 15]. 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 [19].

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 $u_t(x, y)$ and $v_t(x, y)$, i.e.,

$$I_t(x + u_t, y + v_t) = I(x, y). \quad (1)$$

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

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

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

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

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* [8]

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

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

The 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 [13]. Regularization-based algorithms add smoothness constraints on the u and v fields to obtain good solutions [8]. 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 [6, 18].

4 Spline-based flow estimation

In our work, we have chosen to model 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* [21]. The value for the displacement at a pixel i can be written as

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

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

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 [14]. First, we compute the gradient of E in (3) with respect to each of the parameters $\hat{u}_j$ and $\hat{v}_j$, $g_j^u = \frac{\partial E}{\partial \hat{u}_j}$ and $g_j^v = \frac{\partial E}{\partial \hat{v}_j}$ [18, 21]. We

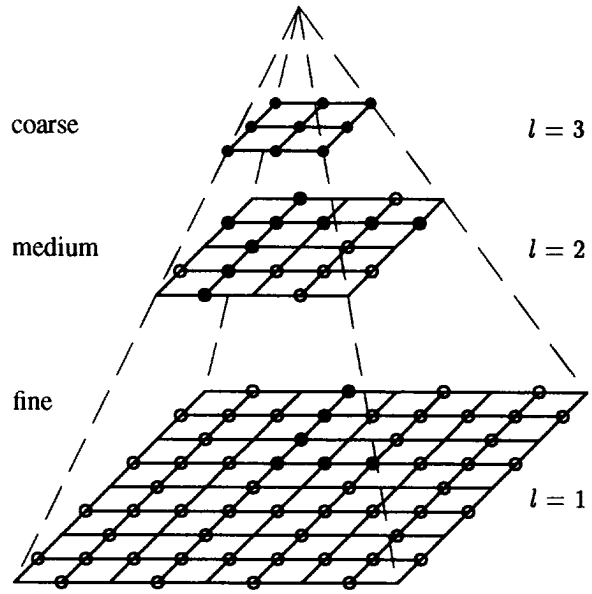

Figure 1: Multiresolution pyramid. The multiple resolution levels are a schematic representation of the hierarchical basis spline. The circles indicate the nodes in the hierarchical basis. Filled circles (•) are free variables in the quadtree spline (Section 6), while open circles (○) must be zero (see Figure 3).

also compute the approximate Hessian matrix $\mathbf{A}$, which contains entries of the form $a_{jk}^{uv} = 2\sum_i \frac{\partial e_i}{\partial \hat{u}_j}\frac{\partial e_i}{\partial \hat{v}_k}$. Then, we compute an increment $\Delta\mathbf{u}$ to the current displacement estimate $\mathbf{u}$ which satisfies

$$(\mathbf{A} + \lambda\mathbf{I})\Delta\mathbf{u} = -\mathbf{g}, \tag{5}$$

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 [14]. To solve this large, sparse system of linear equations, we use preconditioned gradient descent

$$\Delta\mathbf{u} = -\alpha\,\mathbf{B}^{-1}\mathbf{g} = -\alpha\tilde{\mathbf{g}} \tag{6}$$

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 of $\mathbf{A}$, and $\tilde{\mathbf{g}} = \mathbf{B}^{-1}\mathbf{g}$ is called the *preconditioned residual vector*. See [18] for more details on our algorithm implementation.

To handle larger displacements, we run our algorithm in a coarse-to-fine (hierarchical) fashion using a Gaussian image pyramid [7]. We start the algorithm on one of the smaller pyramid levels, and use the resulting flow estimates to initialize the next finer level.

5 Hierarchical basis splines

To accelerate the convergence of the gradient descent algorithm, we use hierarchical basis splines. Hierarchical basis functions are based on using a pyramidal representation for the data [7], but where the number of nodes in the pyramid is equal to the original number of nodes at the finest level (Figure 1).

0. $\beta_0 = 0,\ \mathbf{d}_{-1} = 0$
1. $\mathbf{g}_n = -\nabla E(\mathbf{u})$
2.† $\tilde{\mathbf{g}}_n = \mathbf{S}\mathbf{Z}\mathbf{S}^T\mathbf{B}^{-1}\mathbf{g}_n$
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. increment n, loop to 1.

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

Figure 2: Hierarchical basis preconditioned conjugate gradient algorithm

To convert from the hierarchical basis representation $\tilde{\mathbf{u}}$ to the usual fine-level representation $\mathbf{u}$ (which is called the *nodal basis* representation [23]), 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. This process can be written algebraically as

$$\mathbf{u} = \mathbf{S}\tilde{\mathbf{u}} = \mathbf{S}_1\mathbf{S}_2\ldots\mathbf{S}_{L-1}\tilde{\mathbf{u}}, \tag{7}$$

with

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

where the $\tilde{w}_{jk}$ are weighting functions that depend on the particular choice of interpolation function.

Using a hierarchical basis representation for the flow field is equivalent to using $\mathbf{S}\mathbf{S}^T$ as a preconditioner, i.e., $\tilde{\mathbf{g}} = \mathbf{S}\mathbf{S}^T\mathbf{g}$ [4, 17]. The transformation $\mathbf{S}\mathbf{S}^T$ 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. When combining hierarchical basis preconditioning with the block diagonal preconditioning in (6), we use $\tilde{\mathbf{g}} = \mathbf{S}\mathbf{S}^T\mathbf{B}^{-1}\mathbf{g}$, which may not be optimal. In future work, we plan to develop optimal combinations of block diagonal and hierarchical basis preconditioning.

To summarize our algorithm (Figure 2), we keep both the hierarchical and nodal representations, and map between the two as required. For accumulating the distances and gradients required in (5), 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 $\mathbf{g}$ 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. Examples of the accelerations in convergence available with hierarchical basis splines are shown in [21].

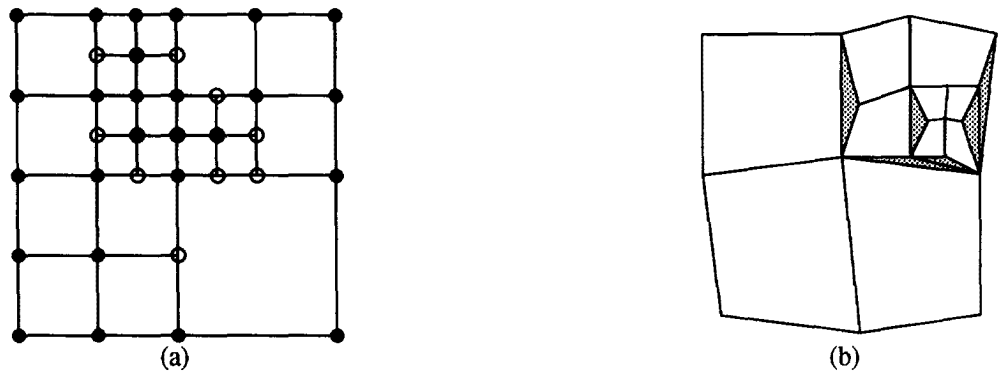

Figure 3: Quadtree associated with spline function, and potential cracks in quadtree spline: (a) the nodes with filled circles (•) are free variables in the associated hierarchical basis, whereas the open circles (○) (and also the nodes not drawn) must be zero (in the nodal basis, these nodes are interpolated from their ancestors); (b) potential cracks in a simpler quadtree spline are shown as shaded areas.

6 Quadtree 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 3) [16]. A quadtree spline is a continuous function built over a quadtree domain which interpolates 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 3b) unless a *crack-filling* strategy is used [16]. The simplest strategy is to simply replace the values at the nodes along a crack edge (the white circles in Figure 3) with the average values of its two parent nodes along the edge.

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 [20], is to use a hierarchical basis and to selectively zero out nodes in this basis. Observe that if in Figures 1 and 3a we set the values of the open circles (○) 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, simply requiring a selective zeroing step between the $\mathbf{S}^T$ and $\mathbf{S}$ operations (algebraically, we write $\tilde{\mathbf{g}} = \mathbf{S}\mathbf{Z}\mathbf{S}^T\mathbf{g}$, where $\mathbf{Z}$ is a diagonal matrix with 1's and 0's on the diagonal—see Figure 2). 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. Third, as we discuss in [21], the adaptive hierarchical basis idea is even more general than the quadtree spline, and corresponds to a kind of multi-resolution prior model.

7 Subdivision strategy

A 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. 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* [9], recently used by Müller *et al.* [11] 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,

$$\mathbf{u}_i^{\mathrm{N}} = \frac{(e_i G_i^x, e_i G_i^y)}{\|(G_i^x, G_i^y)\| + \epsilon}, \tag{8}$$

where e_i is the intensity error at pixel i and (G_i^x, G_i^y) is the gradient. This measure is different from that used in [9, 11], who sum the numerator and denominator in (8) 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 $\|\mathbf{u}_i^{\mathrm{N}}\|^p$ over all the pixels in the patch and compare it to a threshold $\theta_{\mathbf{u}}$. 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 4a–c show an example of a quadtree spline motion estimate produced with this splitting technique for a sim-

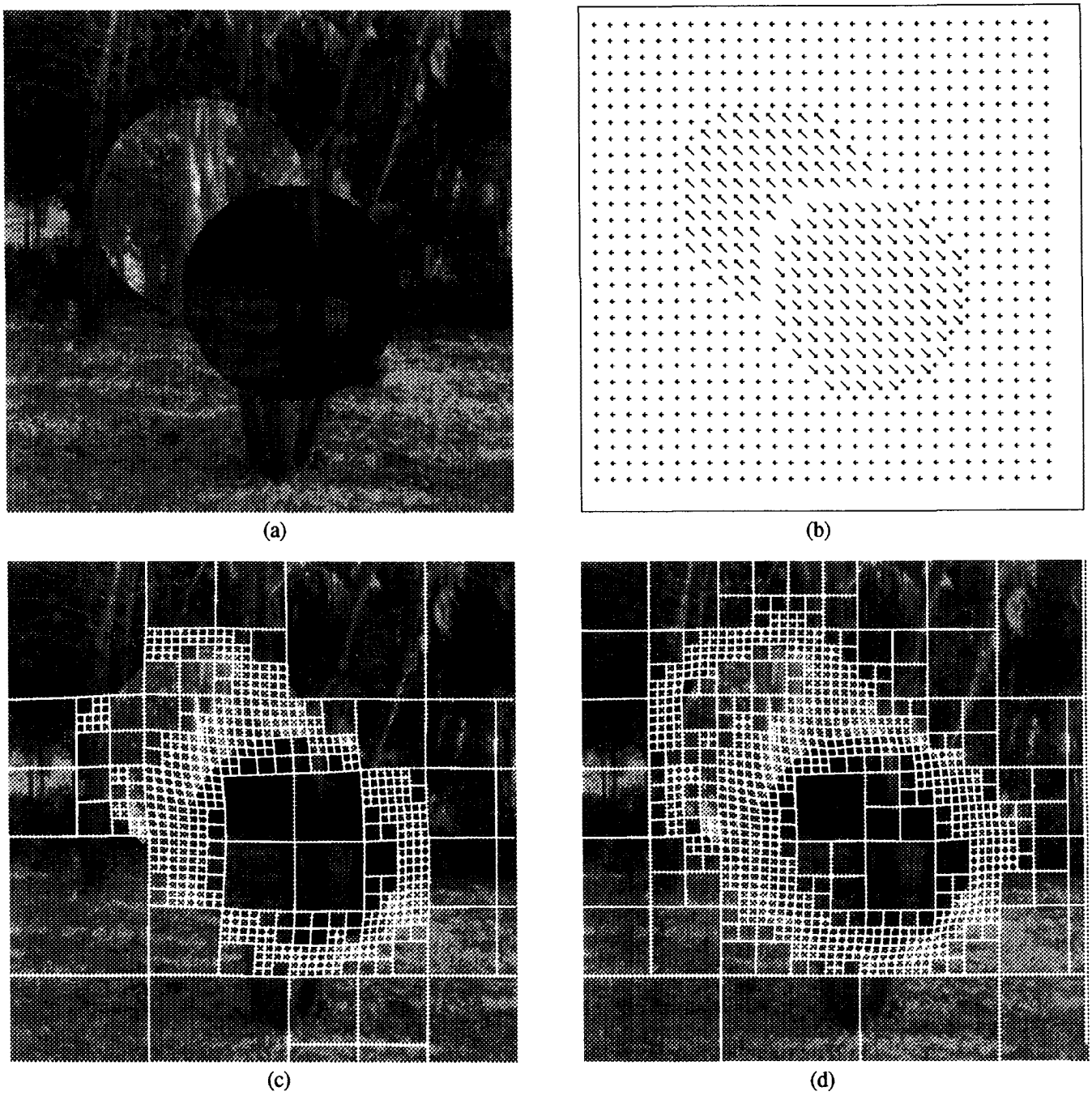

Figure 4: Quadtree spline motion estimation (**Two Discs (SRI Trees)** sequence): (a) input image, (b) true flow, (c) split technique, (d) merge technique.

ple 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 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 3a 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 discuss in [21]). Figure 4d 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.

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 4, for which we know the true motion (Figure 4b). 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 given in [21]. 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. 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 [5, 22]. The results on the **Hamburg Taxi** sequence are shown in Figure 5, 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. Additional experimental results with real image sequences can be found in [21].

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, $\mathbf{u}_t = s_t \mathbf{u}_1$. Flow estimation can then be formulated by summing the intensity differences between the base frame and all other frames [18]. 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.

The multiframe motion estimator forms the basis of our parallel extended image sequence feature tracker [19]. 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 $\mathbf{B}$ and the local intensity error within each spline patch.

To deal with the local minima which can trap our gradient descent technique, we are also adding a discrete search component to our algorithm. At the beginning of each set of iterations, we search around the current (u, v) estimate by trying a discrete set of nearby (u, v) values (as in SSD algorithms [3]). We then average the best 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 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 [11], 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 al-

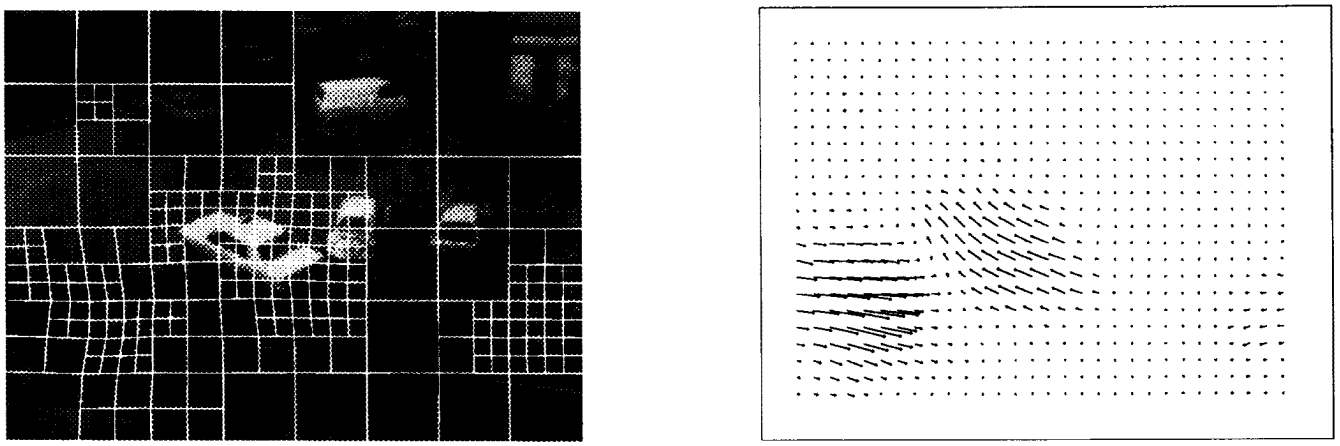

Figure 5: **Hamburg Taxi** sequence: estimated quadtree and estimated flow, merging $s = 8$ patches in a 3-level pyramid.

ready been applied successfully to the elastic registration of 3D medical images [20], and we are in the process of extending our approach to other applications.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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, Santa Margherita Liguere, Italy, May 1992. Springer-Verlag.

[7] 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.

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

[9] 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, Santa Margherita Liguere, Italy, May 1992. Springer-Verlag.

[10] 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.

[11] 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, Seattle, Washington, June 1994. IEEE Computer Society.

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

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

[14] 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.

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

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

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

[18] 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, Seattle, Washington, June 1994. IEEE Computer Society.

[19] 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.

[20] 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, Seattle, Washington, June 1994. IEEE Computer Society.

[21] R. Szeliski and H.-Y. Shum. Motion estimation with quadtree splines. Technical Report 95/1, Digital Equipment Corporation, Cambridge Research Lab, March 1995.

[22] 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.

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