EFFICIENT SELECTION OF IMAGE PATCHES WITH HIGH MOTION CONFIDENCE

Peter Sand and Leonard McMillan

Laboratory for Computer Science
Massachusetts Institute of Technology

ABSTRACT

Motion confidence measures aim to identify how well an image patch determines image motion. These kinds of confidence measures are commonly used to select points for optical flow estimation and feature tracking. The brute force approach of computing confidence for every image patch is inefficient, especially when the patches are large. The faster approach of computing confidence for a regular grid of patches is sub-optimal; good patches may be missed because they straddle grid boundaries. We present a new algorithm that efficiently selects globally optimal patches. Our primary innovation is the use of confidence bounds to identify image regions that should be explored by a queue-based search algorithm.

1. INTRODUCTION

Motion confidence measures provide information about the quality of apparent motion estimates at any given point in an image. These reliability measures allow algorithms to identify and avoid patches suffering from the aperture problem. Considerable research [1, 2] has addressed the issue of measuring motion confidence, but very little work has focused on efficiency—the goal of this paper.

Measures of motion confidence are important to optical flow [3, 4]. For the sake of efficiency, several optical flow algorithms limit their estimation to points where the flow can be computed reliably. This implies that the point selection algorithm must also be efficient.

Confidence measures are also important to feature tracking [1]. In many video sequences, features are lost due to camera motion and mistracking, requiring that the set of features be augmented repeatedly throughout the tracking process. When feature selection is made part of an online tracking algorithm (not just a pre-processing step), efficiency is essential.

2. BACKGROUND

Many patch-based motion estimation algorithms attempt to find a translation vector which minimizes the pixel error for a small region [5, 2]. Given the intensity gradient g_x, g_y over the image and a patch P_{x_0,y_0} centered at x_0, y_0, the optimal least-squares translation estimate involves the following matrix:

$$M(x_0, y_0) = \begin{pmatrix} \sum_{x,y \in P_{x_0,y_0}} g_x^2 & \sum_{x,y \in P_{x_0,y_0}} g_x g_y \\ \sum_{x,y \in P_{x_0,y_0}} g_x g_y & \sum_{x,y \in P_{x_0,y_0}} g_y^2 \end{pmatrix}$$

When this matrix is near-singular, the motion equation is ill-conditioned, suggesting that the image patch provides at most normal flow, not a unique 2D translation vector. A non-singular but ill-conditioned matrix does give a 2D motion solution, but this solution depends critically on small changes in the inputs.

The conditioning of the matrix provides a number of confidence metrics, including the determinant, the sum of eigenvalues, the product of eigenvalues, the least eigenvalue, the greatest eigenvalue, and the conditioning number (ratio of greatest to least eigenvalues). All of these can be be found in closed form from the matrix M. For example, using the symmetry of M, we have that the minimum eigenvalue is given by:

$$\lambda_{min} = \frac{M_{11} + M_{22} - \sqrt{(M_{11} - M_{22})^2 + 4M_{12}^2}}{2}$$

While the maximum eigenvalue is useful for normal flow estimation, the minimum eigenvalue is useful for feature tracking and 2D flow estimation. When the minimum eigenvalue is large, the motion is well constrained in two directions (along both eigenvectors). Experiments have suggested that the minimum eigenvalue is indeed useful in practice [3].

There are a number of other reliability measures [4]. Some are based on robust approaches to the least-squares problem [6, 7]. Non-least-squares measures include Gaussian curvature, properties of a single spatial gradient, and covariance eigenvalues [8].

3. BASIC OPTIMIZATIONS

When summing gradient products across overlapping image patches, several simple optimizations can be made. These optimizations work for naive brute-force methods and the more complicated method presented in this paper.

A substantial gain in efficiency can be obtained by storing the gradient products g_x^2, $g_x g_y$, and g_y^2 for each pixel. These stored values can be used instead of performing a multiplication for every pixel of every patch. By moving the multiplications out of the inner loop, we reduce the number of multiplications by a factor of A, where A is the area (in pixels) of the patch.

A further optimization involves computing a hierarchy of gradient products which are stored in a pyramid structure. Three pyramids can be constructed: one for each of g_x^2, $g_x g_y$, and g_y^2. When summing these values over a patch, if the patch overlaps a high-level element in the pyramid, part of the sum can be obtained directly from the pyramid.

4. CONFIDENCE MONOTONICITY

To the extent that the confidence metrics describe how much information is provided by image patches, we expect that the confidence increases as the patch size increases. If each pixel provides some sort of constraint, adding pixels should create more constraints, creating a better-conditioned solution.

This assumption applies only so long as the additional constraints do not contradict the original constraints. In general, larger patches are more likely to cross occlusion boundaries or undergo non-translational deformation, giving a less-accurate motion estimation. However, most confidence measures incorporate temporal information only for smoothing (i.e. in the computation of the spatial gradient). Thus the notion of contradictory motion is not directly captured by the confidence measures. As a result, a number of the reliability metrics exhibit this kind of monotonicity.

For some confidence measures, monotonicity is easy to verify. In the case of the sum of eigenvalues, we have $\lambda_{min} + \lambda_{max} = M_{11} + M_{22}$. Since M_{11} and M_{22} are sums of squares of gradient values, this clearly does not decrease as more gradient values are added to the sum.

5. CONFIDENCE SEARCHING

We present an algorithm which localizes patches of high motion confidence. The algorithm is given an image and a desired patch size, then asked to find the N patches of this size with the highest confidence.

The algorithm exploits the tendency for real-world images to contain regions of low confidence, such as areas with little texture. We save time by identifying regions of low confidence without actually computing the confidence for every point in the region.

5.1. Bounding functions

The primary tool used by this algorithm is a bounding function that produces an upper bound on the confidence for every patch contained within a given region. This bounding function can be the confidence measure itself when confidence is monotonic. We may also use an upper bound on the confidence measure, so long as the bound converges to the true confidence measure as the region size approaches the desired patch size.

5.2. Confidence queue search

The search algorithm uses a priority queue of rectangular image regions. The priority value is the upper bound on the confidence measure for that region.

The first item placed in the priority queue is the entire image rectangle. At each iteration we dequeue the rectangle with the largest upper bound (largest priority value), split the rectangle along the widest dimension, and compute new confidence bounds for each half. We then insert the new rectangles back into the queue.

The first time we remove a rectangle of the desired patch size, we have found the global optimum of the confidence function (the patch of maximal confidence). If this is our goal, we can terminate here; if we want the top N patches, we continue with the algorithm until we have removed N patches of the desired patch size (Figure 1).

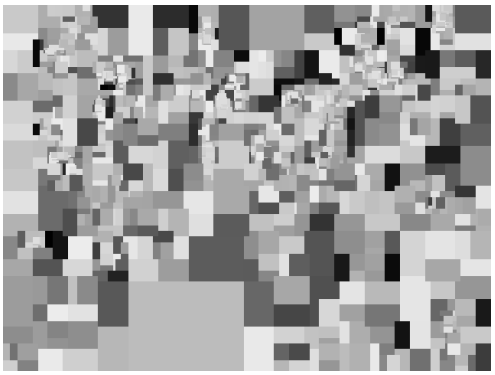

Fig. 1. Bounds computed by the queue algorithm in the process of finding the 100 most confident patches

5.3. Region splitting

Simply splitting each rectangle in half would eliminate those patches that cross the boundary between the halves. We want to consider every patch contained within the image, so we split regions such that they overlap across a distance equal to the desired patch width (Figure 2).

6. DISTRIBUTION OF MULTIPLE POINTS

For most motion estimation applications, we want the estimates to not only be confident, but to be well-scattered, i.e. distributed throughout the image. Estimation of egomotion is better conditioned for widely-separated local motion estimates. For geometric reconstruction, we want well-scattered samples to avoid gaps in the scene model. Thus we want to find some subset of features that are both confident and well-separated.

The general brute-force method of evaluating some confidence/separation measure for every subset of patches is infeasible (i.e. exponential: on the order of the number of

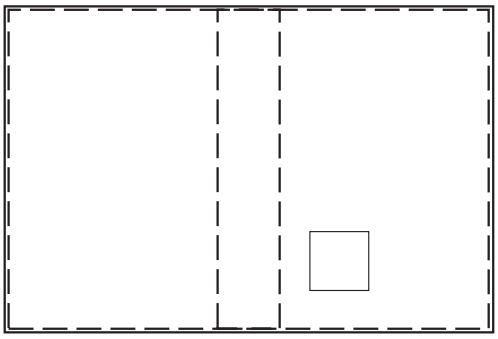

Fig. 2. A rectangle (large solid outline) is split into two overlapping rectangles (dashed) where the amount of overlap is determined by the target patch size (small square)

pixels raised to the size of the subset). The approaches described in this section are substantially more efficient, but they are ad-hoc and sub-optimal in various ways (due to greedy searching and heuristic measures of separation).

6.1. Minimum distance separation

A traditional solution to this problem is to impose a minimum distance between the estimates. This method gives a straightforward greedy selection algorithm: pick the best point, then pick the best point that is sufficiently far from the previous points, and so on.

To implement this kind of separation using the confidence queue, we simply maintain a mask of disallowed pixels and prune the search whenever we fall inside the mask.

6.2. Cell maxima

An approach that guarantees a certain spatial scattering is to tesselate the image and find the best point within each cell. This ensures a certain minimum sample density (in the case of rectangular grid cells, at least one point per four cells of area). Alternately, we may prefer to allocate multiple points to cells that are found to have an abundance of high-confidence patches.

For rectangular tessellations this can easily be handled by the confidence queue algorithm: simply run the algorithm for each cell, initializing the search with the cell boundaries (or expanded boundaries to allow patches that extend partially into neighboring cells).

7. RESULTS

We perform experiments using a set of ten 640 by 480 images that represent a wide range of scenes (with various texture properties and lighting conditions).

7.1. Confidence filtering methods

Section 6 described a trade-off between selecting patches with maximal confidence and selecting patches that are well-scattered. A number of measures can be used to quantify the scattering of a set of points. Presuming that we are concerned with pairwise separation, not overall variance, we take mean pairwise distance as our measure of scattering.

Our baseline algorithm is the standard optical flow approach of using a grid of points. Each grid cell is 20 pixels by 20 pixels, resulting in a total of 768 flow estimates. A confidence value is computed using the minimum eigenvalue of the least-squares matrix for an 8 by 8 neighborhood centered at the center of the cell.

The baseline approach is compared with several others: finding the best 768 points according to the confidence measure (with no penalty for closeness), finding the best point in each of the 768 cells (Figure 3), greedily selecting 768 points while maintaining a minimum distance of 8 pixels, and greedily selecting 768 points while maintaining a minimum distance of 16 pixels (Figure 4).

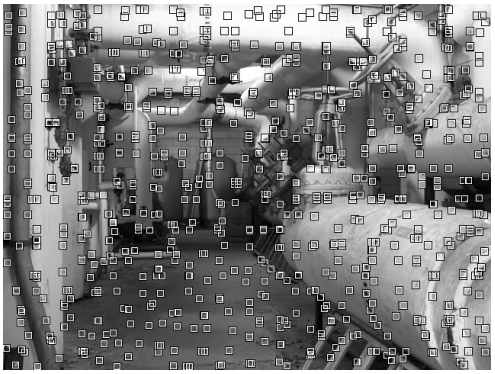

Fig. 3. The Best-Per-Cell patch distribution method

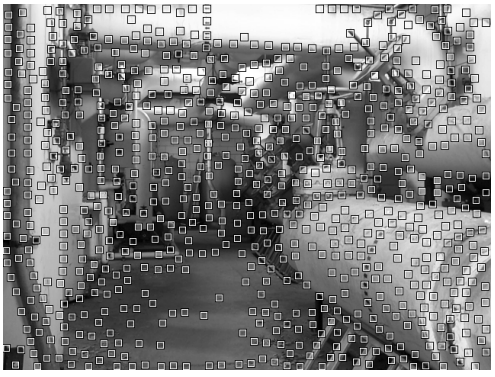

Fig. 4. The Min-Dist-16 patch distribution method

Method	Mean Normalized Confidence	Mean Normalized Separation
Grid-Centers	1.000	1.000
Best-Overall	13.448	0.577
Best-Per-Cell	2.943	0.999
Min-Dist-8	4.483	0.866
Min-Dist-16	2.305	0.989

The best 768 points do indeed have a high confidence, but many of the points are close together, as indicated by the scattering metric. The best-per-cell and min-dist methods are more successful at handling the trade-off: they maintain good scattering while finding points with substantially more confidence than the simple grid approach.

7.2. Efficiency of the confidence queue

Here we compare a naive algorithm (exhaustive search) for selecting patches of high confidence with the our queue-based algorithm (implemented with a heap-based priority queue). Both methods compute gradient products outside the inner loop, but do not use gradient product pyramids.

We first run each algorithm to determine the best 100 patches in the image (without separation constraints). We then run each algorithm to determine the best single point within each of 100 grid cells (where each grid cell is 64 by 48 pixels). In each case, we perform the experiment on 8 by 8 and 16 by 16 pixel patches. Mean running times are reported in seconds, on a 500MHz desktop PC.

Method	Patch Size	Naive	Queue
Best-Overall	8 x 8	4.107	1.738
Best-Per-Cell	8 x 8	3.498	1.565
Best-Overall	16 x 16	12.028	1.786
Best-Per-Cell	16 x 16	11.227	2.281

In each case, the confidence queue algorithm outperforms the naive algorithm. As expected, the queue algorithm has better relative performance on the larger patch size, since larger patches cause the naive algorithm to execute longer inner loops.

7.3. Behavior of the confidence queue

The running time of the naive algorithm is independent of the image content, in contrast to the queue method. The performance of the queue algorithm is related to the rate of convergence of the bounding function (i.e. how good the bounds are) and the shape of the confidence function for a particular image. If the confidence function has similar values across the image, the algorithm is unable to prune its search.

Using our set of sample images, we are able to perform a limited exploration this dependence on image content. For each image, we compute the median confidence across all 8 by 8 patches in the image, using this as a crude measure of the distribution of "texture" in the image. We plot this against the running time of the queue algorithm and find substantial correlation (Figure 5).

8. CONCLUSION

We present an algorithm for efficiently finding patches with high motion confidence, using a variety of different confidence measures. The algorithm has performance which exceeds a naive algorithm on our test cases.

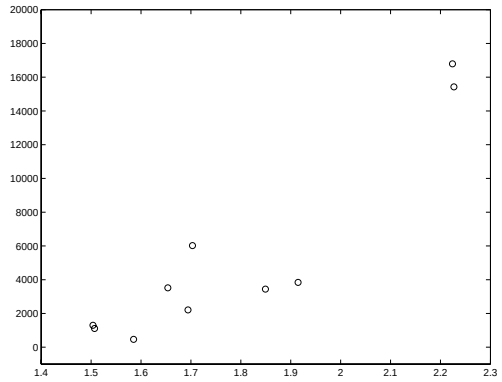

Fig. 5. A plot of median confidence (vertical axis) vs. running time of the queue-based algorithm

As described, the queue approach finds exact, globally-optimal solutions, but it also provides a framework for developing a variety of approximate algorithms. Given a true bounding function (such as a monotonic confidence metric), the queue finds the same solutions as the brute-force exhaustive search method. However, if the bounding function is highly optimistic, we may be able to find patches of reasonably high confidence faster than any exact algorithm. Thus we hope in the future to develop approximate algorithms for confidence optimization that approach real-time performance.

9. REFERENCES

[1] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto, "Improving feature tracking with robust statistics," 1999.

[2] Jianbo Shi and Carlo Tomasi, "Good features to track," in *CVPR*, 1994, pp. 593–600.

[3] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt, "Performance of optical flow techniques," *CVPR*, 1992, pp. 236–242.

[4] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys*, vol. 27, no. 3, 1995, pp. 433–467.

[5] B.D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI81*, 1981, pp. 674–679.

[6] E.P. Simoncelli, E.H. Adelson, and D.J. Heeger, "Probability distributions of optical flow," in *CVPR*, 1991, pp. 310–315.

[7] M.J. Black and P. Anandan, "A framework for the robust estimation of optical flow," in *ICCV*, 1993, pp. 231–236.

[8] D. J. Heeger, "Optical flow using spatiotemporal filters," *International Journal of Computer Vision*, vol. 1, pp. 279–302, 1988.