Efficiency and Accuracy Tradeoffs in using Projections for Motion Estimation*

Dirk Robinson
Department of Electrical Engineering
University of California
Santa Cruz, CA 95064
dirkr@soe.ucsc.edu

Peyman Milanfar
Department of Electrical Engineering
University of California
Santa Cruz, CA 95064
milanfar@ee.ucsc.edu

Abstract

This paper presents an investigation of the use of the Radon (projection) transform in speeding up existing image registration techniques. The ultimate goal is to make these algorithms more computationally simple, while simultaneously realizing acceptably accurate performance.

The use of the projections in estimating translational motion decomposes a 2-D problem into a pair of 1-D problems, leading to significant computational savings. Here we present the tradeoffs of computational efficiency and accuracy for two current methods. Our experiments show that for most applications, the modifications we suggest in using the projections instead of the image directly cost little in performance, yet realize dramatic improvements in computational efficiency.

1 Introduction

Image registration and motion estimation are well researched problems having many applications in various fields such as computer vision and video processing and compression. In many practical applications, the computational cost of performing accurate motion estimation is generally prohibitively high. For instance, in the field of video coding, fast and accurate motion estimation is fundamental for any real-time motion compensating video encoder. In fact, most real-time video coders require special hardware to achieve the necessary motion estimation efficiency to support real-time encoding. As also realized in [9], utilization of the Radon transform within these motion estimation techniques can provide significant computational savings. However, since the projection-based techniques do not make full use of the data directly, they will naturally suffer some loss in performance. This paper presents a quantitative characterization of the performance-speed tradeoff from both experimental and theoretical perspectives.

We have chosen two current motion estimation techniques to evaluate the tradeoffs between performance and accuracy by using projections. We examine the differential technique of Lucas/Kanade [4] and the relative phase-based method of Stone et al. [8]. We chose these two techniques as they produce accurate results while representing two distinctly different methodologies for estimating a motion field. Furthermore, from [3] we see that the Lucas/Kanade method is one of the faster methods and, with our computational speedup, will come even closer to real-time given the current frame rates. We will describe the basic ideas behind each of these techniques below and then describe the modifications required in each case to take advantage of the Radon transform. We will compare the general computational costs associated with each method as well as the accuracy of each method on a set of image sequences.

2 The Radon transform

The Radon transform or projection of an image at an angle θ is defined as,

$$\begin{aligned} g(p,\theta) &= \mathcal{R}_\theta\left[f(x,y)\right] \\ &= \iint f(x,y)\delta\left(p - x\cos(\theta) - y\sin(\theta)\right)dx\,dy \end{aligned} \tag{1}$$

It is well-known that uniform translation in the image domain corresponds to translation in the projection domain,

$$\begin{aligned} &\mathcal{R}_\theta\left[f(x-v_x, y-v_y)\right] = \\ &\qquad g(p - v_x\cos(\theta) - v_y\sin(\theta), \theta). \end{aligned} \tag{2}$$

*This work was supported in part by the National Science Foundation under Grant CCR-9984246

From this result, it becomes evident that by estimating the components of *projected* motion (i.e. $u_\theta = v_x \cos(\theta) + v_y \sin(\theta)$), for at least two independent directions, we can solve for the uniform motion vector $v = [v_x, v_y]^T$. More generally, the two-dimensional motion vectors can then be computed by solving the following least-squares problem:

$$\begin{bmatrix} u_{\theta_1} \\ \vdots \\ u_{\theta_n} \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) \\ \vdots & \vdots \\ \cos(\theta_n) & \sin(\theta_n) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} + \text{error} \quad (3)$$

In this way, motion estimation in 2-D is reduced to a set of 1-D estimation problems, reducing the dimensionality of the optimization problem for any motion estimation technique, and hence dramatically reducing the computational complexity. In what follows, to keep the computational complexity low, we only use two angles of projection at zero and 90 degrees. We will show that the use of only these two angles produces excellent overall results. In general, there may be some benefit in choosing the projection angles differently from these, but we leave the analysis of that issue for future work.

Finally, as indicated in [5], the model of *projected motion* can also be generalized to include higher order models of motion such as general affine motion. However, for the purposes of this paper, we will treat only the case of locally translational motion.

3 Motion Estimation Techniques

Each of the two techniques we will study subdivides the images into blocks and performs the motion estimation on each block. Throughout this paper we use the notation Ω to indicate a block or region. The block sizes are chosen to both avoid the aperture problem, and to improve the likelihood that the actual block motion fits the assumed translational motion model. If the block size is small enough, the motion within the block can be modeled as translational. We shall say more about this issue later.

3.1 Gradient Based Technique ([4])

A differential method computes image velocity directly from the image pixel intensities by making an assumption about the conservation of intensity between a pair of images [2], which leads to the well recognized gradient constraint equation relating image motion to image gradients in space and time:

$$f_x v_x + f_y v_y + f_t = 0, \quad (4)$$

where the subscripts on f denote partial derivatives with respect to the relevant variables. In the current context, the velocities v_x and v_y in (4) are modeled as locally translational so that each image block is a shifted version of the corresponding block in the adjacent frame. Lucas and Kanade, [4] use a locally constant model coupled with a weighted least squares approach, to solve for the motion vector field. viz.

$$\sum_{x,y \in \Omega} W^2(x,y) \left[\nabla f \cdot [v_x \; v_y]^T + f_t \right]^2 \quad (5)$$

where Ω is the local neighborhood or the block for which motion is being estimated, and ∇ is the spatial gradient operator. The positive weighting function W is chosen such that the center of the block is weighted more than the periphery, which allows for better localization, and larger block overlaps, while simultaneously reducing the likelihood of encountering the aperture problem[1].

3.2 Projection-Based Gradient Technique

The implementation of the projection-based variant of the Lucas/Kanade method is straightforward. As before, the image is subdivided into overlapping blocks corresponding to Ω in equation (5), but here we apply the Radon transform to the pixel intensities first in each block.

As in the original gradient-based method, we assume a model of constant velocity for each block in the image. Applying the Radon transform to (4) we generate two equations of the form

$$g_p(p, \theta_i, t) u_{\theta_i} + g_t(p, \theta_i, t) = 0, \quad (6)$$

for $i = 1, 2$, from which we find the weighted least squares estimates of u_{θ_i} by minimizing

$$\sum_{p \in \Omega_{\theta_i}} W^2_{\theta_i}(p) \left[\frac{\partial}{\partial p} g(p, \theta_i, t) \; u_{\theta_i} + \frac{\partial}{\partial t} g(p, \theta_i, t) \right]^2 \quad (7)$$

For this pair of minimization problems the neighborhood Ω_θ is the image of the original Ω from (5) under the Radon transform at an angle θ. Finally, using equation (3) we calculate the estimates of constant velocity $\hat{v}_x$ and $\hat{v}_y$ for each block from the pair of estimates $\hat{u}_{\theta_{1,2}}$.

3.3 Relative-Phase Technique [8]

Stone et al. have presented a variation of the traditional Fourier based image registration technique, examining the relative phase between a pair of images. Here again, the image is subdivided into overlapping blocks. Given an $N \times N$ block $f_1 = f(x, y, 0)$ and

[1] The aperture problem describes a situation wherein there is insufficient intensity gradient information to determine unique motion vectors [7].

a translated version of the block $f_2 = f(x, y, \Delta t) = f(x - v_x\Delta t, y - v_y\Delta t, 0)$, in the 2D Discrete Fourier Transform domain we have,

$$F_2 = F_1 e^{-j2\pi(\omega v_x \Delta t + \nu v_y \Delta t)/N} \quad (8)$$

where F_i denotes the Fourier transform of the image f_i, and j represents $\sqrt{-1}$. For convenience, we can let $\Delta t = 1$. We have

$$\frac{F_1\, F_2^*}{|F_2|^2} = \frac{F_1(\omega,\nu)}{F_2(\omega,\nu)} = e^{j2\pi(\omega v_x + \nu v_y)/N} = e^{j\phi(\omega,\nu)} \quad (9)$$

The displacement vector $[v_x, v_y]^T$ can then be estimated by fitting a linear model to the phase function ϕ. i.e. by minimizing:

$$\sum_{\omega,\nu} C(\omega,\nu)\,[2\pi(\omega v_x + \nu v_y)/N - \phi(\omega,\nu)]^2 \quad (10)$$

where the sum is computed over the appropriate region of the Fourier plane. In general, this phase-based approach works quite well only for images with high SNR, and its performance degrades rather rapidly in the presence of noise. Also, this model assumes sub-pixel motion. In the case of super-pixel motion, phase-wrapping again degrades its performance. To address these issues, Stone et al. generate a binary weighting function (or mask) C, which reduces the influence of spatial aliasing and noise and significantly improves performance. In effect, C regularizes the problem and produces more robust estimates by trimming or eliminating spectral values that are more susceptible to noise and aliasing effects.

3.4 Projection-Based Relative-Phase Technique

As with the gradient-based method, we utilize the Radon Transform in two orthogonal directions $\theta_{1,2}$ on the intensity values for each block as an initial step, generating a pair of 1-D image sequences $g(p, \theta_{1,2}, t)$.

Analogous to the 2-D case, equation (9) becomes a pair of equations after applying the Radon transform, where each equation has the form,

$$\frac{G_1(k)}{G_2(k)} = e^{j2\pi(k u_\theta)/N} = e^{j\phi_\theta(k)} \quad (11)$$

where $G_i(k)$ denotes the Fourier transform of $g(p, \theta_i)$. As in the 2-D method, the pair of parameters u_{θ_1} and u_{θ_2} are then estimated as the minimizers of

$$\sum_k C_\theta(k)\,[(2\pi k u_\theta)/N - \phi_\theta(k)]^2 \quad (12)$$

where again the sum is computed over the appropriate region of the 1-D Fourier space. Here we apply the same assumptions as made by [8] to generate a binary weighting mask C_θ to reduce the effect of spurious spectral values.

Both estimates of $\hat{u}_\theta$ can then be used in (3) to generate estimates of $\hat{v}_x$ and $\hat{v}_y$.

4 Experiments

Following the papers [1], [3] we applied the methods described above to many standard experimental image sequences, for which the true motion fields were given. For the sake of brevity, we report representative results from three such sequences. These sequences, as in [1], are:

1. **Diverging Tree** - Motion in the line of sight of the camera with approximately radially symmetric motion vector field. The motion is sub-pixel at the center of the image and becomes super-pixel towards the edges of the image.
2. **Translating Tree** - Simulation of translational camera motion in the x-direction. The motion in the x-direction is super-pixel and sub-pixel in the y-direction.
3. **Yosemite Flythrough** - Image sequence with the most complex motion field. There are regions of sub and super pixel motion.

4.1 Error Measures

Following [1] we measured the mean angular error between the correct (true) motion vectors $\vec{V_c} = [v_x, v_y, 1]^T$ and the estimated motion vector $\vec{V_e} = [\hat{v}_x, \hat{v}_y, 1]^T$. (The 1 in the third or time dimension reveals the assumption of normalized (unit) time steps between frames.) The mean angular error between $\vec{V_c}$ and $\vec{V_e}$ is measured as

$$\overline{\psi}_{ang} = \frac{1}{B}\sum_b \arccos(\vec{V_c}^T(b) \cdot \vec{V_e}(b)) \quad (13)$$

where B is the total number of blocks (motion vectors) estimated for the image sequence, $\vec{V_e}(b)$ represents the motion vector estimate for the b-th block, and $\vec{V_c}(b)$ represents the actual motion at the center of the same block. We measure $\overline{\psi}_{ang}$ in units of degrees.

To gather more information about the four motion estimation methods we also computed the mean magnitude error defined as:

$$\overline{\psi}_{mag} = \frac{1}{B}\sum_b \sqrt{(v_x(b) - \hat{v}_x(b))^2 + (v_y(b) - \hat{v}_y(b))^2} \quad (14)$$

Again, the standard deviation of the estimated mean magnitude is also included in the results of our tests.

Finally, we also measured the mean-squared error for each direction

$$\begin{aligned} \overline{\psi}_{MSE_x} &= \frac{1}{B}\sum_b (v_x(b) - \hat{v}_x(b))^2 \\ \overline{\psi}_{MSE_y} &= \frac{1}{B}\sum_b (v_y(b) - \hat{v}_y(b))^2 \end{aligned} \qquad (15)$$

and the corresponding biases and sample variances for the motion estimates. Computation of bias and sample variances allow us to study the tradeoffs in these values. We also tabulated the standard deviation about the estimated mean angular and magnitude error as in [1].

5 Results and Conclusions

After conducting exhaustive experiments using different values of the parameters that can be chosen for each algorithm, for final comparison, we selected a representative set of parameters and used the same values (when appropriate) across all the methods. In effect, we strived to choose a set of parameters which would not unfairly disadvantage any one technique. Furthermore, we wanted these parameters to be realistic in a practical setting. In summary, we chose a block size of 30 pixels and a motion vector spacing of 10 pixels to allow for block overlap. A block size of 30 pixels represents a modest tradeoff between speed and accuracy. Furthermore, this block size and spacing could fit the needs of a video codec. Another parameter we chose for the gradient-based methods was a weighting function W with a standard deviation of 20% of the block size or 6 pixels. This is intuitively reasonable since the desired motion vector spacing was 10 pixels and therefore this weighting function would help localize the motion estimates in all cases. Finally, we chose to use the modified Prewitt operator to compute the gradients. To eliminate edge effects we discarded the outside ring of gradient estimates, essentially reducing the block size to 28 pixels. The parameters we chose for the relative-phase methods from our experiments were basically in line with the suggestions made in [8].

Tables 1 to 3 show the results of our tests using the above parameters[2].

As a visual example, Figure 1 shows the estimated motion vector fields for the Diverging Tree image sequence overlaid atop one image of the sequence. The motion vector fields were obtained from the gradient-based methods. Note that the motion vector fields are visually quite similar.

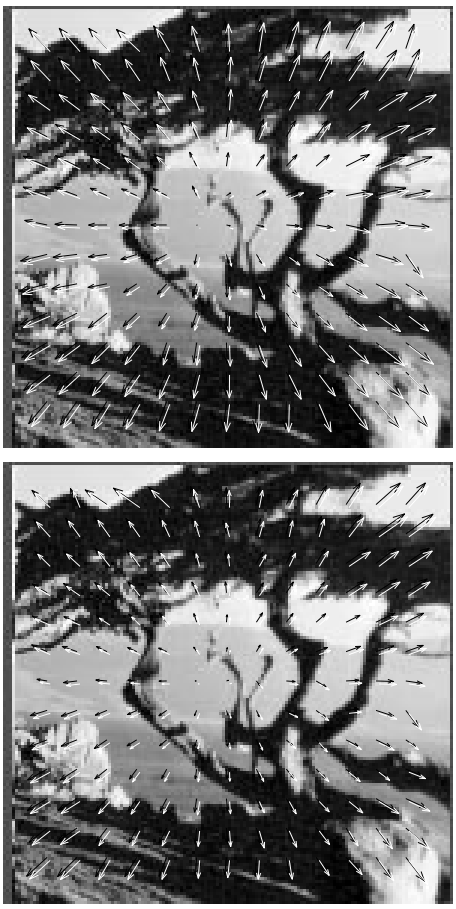

Figure 1: Motion Vector field from 2-D (upper) and 1-D (lower) Gradient-Based method

We observe that while the accuracy of the 1-D and 2-D methods appear to be statistically equivalent, the computational complexity is dramatically reduced in the projection-based approaches. The 1-D gradient-based method required a factor of 10 fewer floating-point operations than its 2-D counterpart, and the total CPU time was reduced by about a factor of 12. More dramatic still, the projection-based phase-based technique reduced the number of floating-point operations by a factor of about 25, while still yielding a factor of about 9 improvement in total CPU time[3]. On balance, these results indicate that by using projections in motion estimation, dramatic savings in computation can be realized with essentially no degrada-

[2]Note that we did not perform any pre-smoothing of the image sequences to reduce the effects of aliasing and noise. This pre-processing step was typically used in producing the results found in [1]. Also, we do not perform any post-processing of the motion vectors to eliminate or smooth the motion estimates. Many other methods perform this step in an attempt to eliminate spurious estimates [1].

[3]The difference between the total floating-point operations and total CPU time results from memory management.

Method	Lucas1	Lucas2	Stone 1	Stone2
Mean Angular Error (degrees)	5.888	6.112	7.474	6.882
Standard Deviation (about the mean)	0.3325	0.3361	0.1928	0.1896
Mean Magnitude Error (pix/frame)	0.153	0.169	0.204	0.189
Standard Deviation (about the mean)	0.0094	0.0110	0.0073	0.0067
Mean Square Error X-dir	0.025	0.033	0.029	0.025
Mean Square Error Y-dir	0.013	0.017	0.022	0.018
Bias X-dir (pix/frame)	-0.0237	-0.0400	-0.0177	-0.0066
Bias Y-dir (pix/frame)	0.0036	-0.0063	-0.0008	-0.0003
Sample Variance X-dir	0.2897	0.2233	0.1921	0.1991
Sample Variance Y-dir	0.2946	0.2490	0.2098	0.2193
Cpu Time (s)	1.930	24.030	4.690	23.450
Flop Count	6326861	52885970	22703975	495772359

Table 1: Results for Divtree sequence using set parameters

Method	Lucas1	Lucas2	Stone 1	Stone2
Mean Angular Error (degrees)	11.385	14.108	8.391	9.423
Standard Deviation (about the mean)	0.7064	0.6470	0.6108	0.4673
Mean Magnitude Error (pix/frame)	0.574	0.778	0.501	0.593
Standard Deviation (about the mean)	0.0269	0.0231	0.0286	0.0256
Mean Square Error X-dir	0.366	0.680	0.382	0.461
Mean Square Error Y-dir	0.085	0.015	0.008	0.001
Bias X-dir (pix/frame)	-0.5029	-0.7701	-0.4904	-0.5919
Bias Y-dir (pix/frame)	0.0323	0.0263	0.0075	0.0012
Sample Variance X-dir	0.1046	0.0738	0.0842	0.0507
Sample Variance Y-dir	0.0840	0.0147	0.0078	0.0014
Cpu Time (s)	1.920	23.880	4.710	23.710
Flop Count	6326861	52885970	22703975	495773097

Table 2: Results for Trantree sequence using set parameters

Method	Lucas1	Lucas2	Stone 1	Stone2
Mean Angular Error (degrees)	18.820	21.195	19.866	31.106
Standard Deviation (about the mean)	0.7245	0.7861	0.9009	1.0724
Mean Magnitude Error (pix/frame)	1.120	1.023	0.915	1.261
Standard Deviation (about the mean)	0.0503	0.0359	0.0377	0.0451
Mean Square Error X-dir	0.874	0.993	1.290	1.847
Mean Square Error Y-dir	2.071	0.913	0.496	1.099
Bias X-dir (pix/frame)	0.3316	0.1934	-0.0200	-0.0136
Bias Y-dir (pix/frame)	-0.1000	0.4245	0.3706	0.5686
Sample Variance X-dir	2.4748	1.3411	0.6628	0.1786
Sample Variance Y-dir	1.5224	0.1360	0.2613	0.1725
Cpu Time (s)	7.530	96.160	18.770	91.360
Flop Count	24970487	207414886	89606789	1956688790

Table 3: Results for Yosem sequence using set parameters

tion in accuracy. With fast hardware implementations of the Radon transform such as that found in [6] and [9], we can expect that the use of projections will definitely provide significant speedups for applications such as video coding that require fast and accurate motion estimation.

References

[1] J. Barron, D. Fleet, S. Beauchemin, and T. Burkitt. Performance of optical flow techniques. *CVPR*, 92:236–242, 1992.

[2] B. K. Horn. *Robot Vision*. MIT Press, Cambridge, 1986.

[3] H. Liu, T.-H. Hong, M. Herman, T. Camus, and R. Chellappa. Accuracy vs. efficiency trade-offs in optical flow algorithms. *Computer Vision and Image Understanding*, 72:271–286, December 1998.

[4] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA81*, pages 121–130, 1981.

[5] P. Milanfar. A model of the effect of image motion in the Radon transform domain. *IEEE Transactions on Image Processing*, 8(9):1276–1281, September 1999.

[6] J. Sanze, E.B.Hinkle, and A.K.Jain. *Radon and Projection Transform-based Computer Vision*. Springer-Verlag, Berlin, Germany, 1988.

[7] A. Singh. *Optic Flow Computation*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[8] H. S. Stone, M. Orchard, and E.-C. Chang. Subpixel registration of images. *Proceedings of the 1999 Asilomar Conference on Signals, Systems, and Computers*, October 1999.

[9] C. Tu, T. D. Tran, J. L. Prince, and P. Topiwala. Projection-based block matching motion estimation. *Proc. SPIE Applications of Digital Image Processing XXIII*, pages 374–384, August 2000.