Joint Tracking of Features and Edges

Stanley T. Birchfield Shrinivas J. Pundlik
Electrical and Computer Engineering Department
Clemson University, Clemson, SC 29634
{stb, spundli}@clemson.edu

Abstract

Sparse features have traditionally been tracked from frame to frame independently of one another. We propose a framework in which features are tracked jointly. Combining ideas from Lucas-Kanade and Horn-Schunck, the estimated motion of a feature is influenced by the estimated motion of neighboring features. The approach also handles the problem of tracking edges in a unified way by estimating motion perpendicular to the edge, using the motion of neighboring features to resolve the aperture problem. Results are shown on several image sequences to demonstrate the improved results obtained by the approach.

1. Introduction

Tracking features between consecutive image frames is a fundamental problem in computer vision. By establishing correspondence between sparse points, feature tracking captures the essence of the motion of a scene in a compact description, making it useful for a wide variety of applications, such as structure from motion [19, 21, 5, 9], motion segmentation [5, 13, 16], object tracking [11], image mosaicking [23], and face tracking [6].

The classic approach to feature tracking is the differential technique of Lucas and Kanade [14], which estimates the motion of a small patch of image intensities (features). The basic idea is simple: A linear system is repeatedly solved to find the best alignment for the image patch in the other image. Over the years, researchers have proposed a number of improvements to the basic algorithm. Shi and Tomasi [17] determine when a feature has been lost by computing the best affine warp between the first frame and the current frame. They noted that, although the translation model is acceptable between consecutive image frames, a richer model such as affine is necessary when considering frames separated widely in time. Their work was later extended by Tommassini *et al*. [20] to automatically reject spurious features. More recently, Šegvić *et al*. [22] model the feature support adaptively to improve long-range performance of a feature in the presence of dominant forward motion. When non-causal processing is appropriate, Sivic *et al*. [18] describe a technique for repairing the trajectories of features using the start and end locations in a set of frames. To handle lighting and exposure changes, various methods have been proposed [12, 10]. Other researchers have focused their efforts upon improving the performance of Lucas-Kanade when applied to a single large image patch to be tracked [7, 15, 1].

One problem that remains largely unaddressed in the literature, however, is that of tracking features together. Most previous approaches to feature tracking consider each feature independently of the other features, thus neglecting important information that is available in determining the motion of a feature. This bottom-up approach is at odds with the Gestalt emphasis upon the importance of *motion coherence*, namely that nearby pixels will often have similar motions. Even though independent tracking oftentimes succeeds, errors persist because the neighboring information is ignored.

In this paper we propose a framework in which features are tracked jointly. Inspired by the work of Bruhn *et al*. [4], we use global optic flow methods (Horn-Schunck) to improve the results obtained by local optic flow (Lucas-Kanade). A smoothness term is added to the formulation to penalize the deviation of the displacement of a feature from its expected value, which is computed by fitting a motion model to the displacements of the neighbors. The approach bears a resemblance to that of Kim *et al*. [12] in which features are also tracked jointly, but in our work the features influence one another *geometrically* rather than *photometrically*. One advantage of the proposed framework is that it facilitates the uniform treatment of features and edges for tracking. Edges have often been avoided in tracking due to their underconstrained nature (the aperture problem). Here, however, the gradient information that enables an edge to be tracked in the direction perpendicular to the edge is combined naturally with the motion of neighboring features. The motion of the edge is determined by minimizing a 2D energy functional that takes into account both the image data and the smoothness of the nearby estimated motions.

2. Lucas-Kanade and Horn-Schunck

As explained by Bruhn *et al*. [4], differential methods for both dense optical flow as well as sparse feature tracking are based on the assumption that the intensity values of the projection of scene points do not change over time:

$$I(x+u, y+v, t+1) = I(x, y, t), \tag{1}$$

where $I(x, y, t)$ is the intensity of pixel $\mathbf{x} = (x, y)^T$ in frame t, and $\mathbf{u} = (u, v)^T$ is the displacement of the pixel between consecutive frames t and $t+1$. For small displacements, a linearized Taylor series expansion yields the well-known *optic flow constraint equation*:

$$f(u, v; I) = I_x u + I_y v + I_t = 0, \tag{2}$$

where the subscripts denote partial derivatives. The well-known *aperture problem* arises because this single equation is insufficient to recover the two unknowns u and v.

The Lucas-Kanade [14] approach to overcoming the aperture problem assumes that the unknown displacement $\mathbf{u}$ of a pixel is constant within some neighborhood. As a result, the displacement can be computed by minimizing

$$E_{LK}(u, v) = K_\rho * \left((f(u, v; I))^2 \right), \tag{3}$$

where $K_\rho * (\cdot)$ denotes convolution with an integration window of size ρ. Differentiating with respect to u and v, and setting the partial derivatives to zero, yields the linear system

$$\begin{bmatrix} K_\rho * (I_x^2) & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & K_\rho * (I_y^2) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} K_\rho * (I_x I_t) \\ K_\rho * (I_y I_t) \end{bmatrix} \tag{4}$$

which is solved iteratively to minimize E_{LK}.

Alternatively, the Horn-Schunck [8] approach regularizes the underconstrained optic flow constraint equation by imposing a global smoothness term. While Lucas-Kanade finds the displacement of a small window around a single pixel, Horn-Schunck computes the global displacement functions $u(x, y)$ and $v(x, y)$ by minimizing

$$E_{HS}(u, v) = \int_\Omega (f(u, v; I))^2 + \lambda \left(|\nabla u|^2 + |\nabla v|^2 \right) dx\, dy, \tag{5}$$

where λ is the regularization parameter and Ω is the domain of the image. The minimum of this functional is found by solving the corresponding Euler-Lagrange equations, leading to

$$\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \lambda \nabla^2 u - I_x I_t \\ \lambda \nabla^2 v - I_y I_t \end{bmatrix}, \tag{6}$$

where $\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ and $\nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}$ are the Laplacian of u and v, respectively. Solving this equation for u and v and using the approximation that $\nabla^2 u \approx h(\bar{u} - u)$, where $\bar{u}$ is the average of the values of u among the neighbors of the pixel, and h is a constant scale factor, we get

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \bar{u} \\ \bar{v} \end{bmatrix} - \frac{I_x \bar{u} + I_y \bar{v} + I_t}{h\lambda + I_x^2 + I_y^2} \begin{bmatrix} I_x \\ I_y \end{bmatrix}. \tag{7}$$

Thus, the sparse linear system can be solved using the Jacobi method with iterations for pixel $(i, j)^T$ of the form:

$$\begin{aligned} u_{ij}^{(k+1)} &= \bar{u}_{ij}^{(k)} - \gamma I_x \qquad (8) \\ v_{ij}^{(k+1)} &= \bar{v}_{ij}^{(k)} - \gamma I_y, \end{aligned}$$

where

$$\gamma = \frac{I_x \bar{u}_{ij}^{(k)} + I_y \bar{v}_{ij}^{(k)} + I_t}{h\lambda + I_x^2 + I_y^2}. \tag{9}$$

Faster convergence is obtained by performing computations in place (Gauss-Seidel) and using successive overrelaxation (SOR).

3. Joint Feature Tracking

It is important to note that, although the derivation of Eq. (6) assumes a continuous formulation, the final result in Eqs. (8)–(9) corresponds to a discrete energy functional, due to the discrete approximation of the Laplacian. This observation motivates us to combine the Lucas-Kanade and Horn-Schunck approaches in Eqs. (3) and (5) into the following functional to be minimized:

$$E_{JLK} = \sum_{i=1}^{N} (E_D(i) + \lambda_i E_S(i)), \tag{10}$$

where N is the number of feature points, and the data and smoothness terms are given by

$$\begin{aligned} E_D(i) &= K_\rho * \left((f(u_i, v_i; I))^2 \right) \qquad (11) \\ E_S(i) &= \left((u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2 \right). \qquad (12) \end{aligned}$$

In these equations, the energy of feature i is determined by how well its displacement $(u_i, v_i)^T$ matches the local image data, as well as how far the displacement deviates from the expected displacement $(\hat{u}_i, \hat{v}_i)^T$. Note that the expected displacement can be computed in any desired manner and is not necessarily required to be the average of the neighboring displacements.

Differentiating E_{JLK} with respect to the displacements $(u_i, v_i)^T$, $i = 1, \ldots, N$, and setting the derivatives to zero, yields a large $2N \times 2N$ sparse matrix equation, whose $(2i-1)$th and $(2i)$th rows are given by

$$Z_i \mathbf{u}_i = \mathbf{e}_i, \tag{13}$$

where

$$Z_i = \begin{bmatrix} \lambda_i + K_\rho * (I_x I_x) & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & \lambda_i + K_\rho * (I_y I_y) \end{bmatrix}$$

$$\mathbf{e}_i = \begin{bmatrix} \lambda_i \hat{u}_i - K_\rho * (I_x I_t) \\ \lambda_i \hat{v}_i - K_\rho * (I_y I_t) \end{bmatrix}.$$

This sparse system of equations can be solved using Jacobi iterations of the form

$$\tilde{u}_i^{(k+1)} = \hat{u}_i^{(k)} - \frac{J_{xx}\hat{u}_i^{(k)} + J_{xy}\hat{v}_i^{(k)} + J_{xt}}{\lambda_i + J_{xx} + J_{yy}} \quad (14)$$

$$\tilde{v}_i^{(k+1)} = \hat{v}_i^{(k)} - \frac{J_{xy}\hat{u}_i^{(k)} + J_{yy}\hat{v}_i^{(k)} + J_{yt}}{\lambda_i + J_{xx} + J_{yy}}, \quad (15)$$

where $J_{xx} = K_\rho * (I_x^2)$, $J_{xy} = K_\rho * (I_x I_y)$, $J_{xt} = K_\rho * (I_x I_t)$, $J_{yy} = K_\rho * (I_y^2)$, and $J_{yt} = K_\rho * (I_y I_t)$.

As before, convergence is greatly increased by performing Gauss-Seidel iterations so that $\hat{u}_i^{(k)}$ and $\hat{v}_i^{(k)}$ are actually computed using a mixture of values from the kth and $(k+1)$th iterations (depending upon the order in which the values are updated), and by performing a weighted average of the most recent estimate and the new estimate (successive overrelaxation). With this modification, the update equations are given by $\mathbf{u}_i^{(k+1)} = (1-\omega)\mathbf{u}_i^{(k)} + \omega\tilde{\mathbf{u}}_i^{(k+1)}$, where $\tilde{\mathbf{u}}_i^{(k+1)}$ is the estimate expressed in Eqs. (14–15), and $\omega \in (0, 2)$ is the relaxation parameter. For fast convergence, ω is usually set to a value between 1.9 and 1.99. Note that for $\omega = 1$ the approach reduces to Gauss-Seidel.

4. Pyramidal Implementation

Both the standard Lucas-Kanade method and the proposed joint Lucas-Kanade method involve iteratively solving a sparse $2N \times 2N$ linear system to find the minimum of a quadratic cost functional. In the former, the matrix is block-diagonal, leading to a simple and efficient implementation via a set of 2×2 linear systems, while in the latter, the off-diagonal terms require the approach presented in the previous section. The difference between the approaches becomes apparent when considering a pyramidal implementation, which is usually necessary to overcome the deficiencies in the linearization approximation in the formulation of the problem in Eq. (2).

The two algorithms are shown. Standard Lucas-Kanade iterates through each pyramid level for each feature, while joint Lucas-Kanade iterates through each feature for each pyramid level. Note that if $\lambda_i = 0$, $i = 1, \ldots, N$, then the two algorithms are exactly the same (except for minor differences in the termination criterion), and that the computation required is the same. Both algorithms are $O(Nnm)$, where N is the number of features, n is the number of pyramid levels, and m is the average number of iterations. However, because it considers all the features at a time, the joint algorithm involves different memory requirements: Instead of precomputing all the pyramidal images, it must precompute the Z_i matrices for all the features.

Algorithm: `Standard Lucas-Kanade`

```
For each feature $i$,
  1. Initialize $\mathbf{u}_i \leftarrow (0,0)^T$
  2. Set $\lambda_i \leftarrow 0$
  3. For pyramid level $n-1$ to 0 step $-1$,
     (a) Compute $Z_i$
     (b) Repeat until convergence:
         i. Compute the difference $I_t$ between the first image and the shifted second image: $I_t(x,y) = I_1(x,y) - I_2(x+u_i, y+v_i)$
         ii. Compute $\mathbf{e}_i$
         iii. Solve $Z_i\mathbf{u}'_i = \mathbf{e}_i$ for incremental motion $\mathbf{u}'_i$
         iv. Add incremental motion to overall estimate: $\mathbf{u}_i \leftarrow \mathbf{u}_i + \mathbf{u}'_i$
     (c) Expand to the next level: $\mathbf{u}_i \leftarrow \alpha\mathbf{u}_i$, where $\alpha$ is the pyramid scale factor
```

Algorithm: `Joint Lucas-Kanade`

```
For each feature $i$,
  1. Initialize $\mathbf{u}_i \leftarrow (0,0)^T$
  2. Initialize $\lambda_i$
For pyramid level $n-1$ to 0 step $-1$,
  1. For each feature $i$, compute $Z_i$
  2. Repeat until convergence:
     (a) For each feature $i$,
         i. Determine $\hat{\mathbf{u}}_i$
         ii. Compute the difference $I_t$ between the first image and the shifted second image: $I_t(x,y) = I_1(x,y) - I_2(x+u_i, y+v_i)$
         iii. Compute $\mathbf{e}_i$
         iv. Solve $Z_i\mathbf{u}'_i = \mathbf{e}_i$ for incremental motion $\mathbf{u}'_i$
         v. Add incremental motion to overall estimate: $\mathbf{u}_i \leftarrow \mathbf{u}_i + \mathbf{u}'_i$
  3. Expand to the next level: $\mathbf{u}_i \leftarrow \alpha\mathbf{u}_i$, where $\alpha$ is the pyramid scale factor
```

Several implementation issues remain. First, how should the regularization parameters λ_i be chosen? Since a large number of features can often be tracked accurately without any assistance from their neighbors, one could imagine weighting some features more than others, e.g., using one of the standard measures for detecting features in the first place [17]. For example, since large eigenvalues of the gradient covariance matrix indicate sufficient image intensity information for tracking, such features could receive smaller smoothing weights (regularization parameter values) than those with insufficient information. However, this scheme is frustrated by the fact that the eigenvalues do not take into account important issues such as occlusions, motion discontinuities, and lighting changes, making it difficult to determine beforehand which features will actually be tracked reliably. As a result, we simply set all of the regularization parameters to a constant value in this work: $\lambda_i = 50$.

Another issue is how to determine the expected values $(\hat{u}_i, \hat{v}_i)^T$ of the displacements. Because the features are sparse, a significant difference in motion between neighboring features is not uncommon, even when the features are on the same rigid surface in the world. As a result, we cannot simply average the values of the neighbors as is commonly done [8, 4]. Instead, we predict the motion displacement of a pixel by fitting an affine motion model to the displacements of the surrounding features, which are inversely weighted according to their distance to the pixel. We use a Gaussian weighting function on the distance, with $\sigma = 10$ pixels.

Finally, because the algorithm enforces smoothness, it is able to overcome the aperture problem by determining the motion of underconstrained pixels that lie along intensity edges. We modify the feature detection algorithm accordingly. To detect features, we use the two eigenvalues $e_{\min}$ and $e_{\max}$, $e_{\min} \leq e_{\max}$ of the original Lucas-Kanade gradient covariance matrix (i.e., Z_i with $\lambda_i = 0$). Rather than selecting the minimum eigenvalue $e_{\min}$, as is often done [17], we select features using $\max(e_{\min}, \eta e_{\max})$, where $\eta < 1$ is a scaling factor. The rationale behind this choice is that along an intensity edge $e_{\max}$ will be large while $e_{\min}$ will be arbitrarily small. Instead of treating an edge like an untextured region, the proposed measure rewards the feature for the information that it does have. For pixels having two comparable eigenvalues, the proposed measure reduces to the more common minimum eigenvalue. In this work we set $\eta = 0.1$.

5. Experimental Results

The proposed algorithm has been compared against a state-of-the-art implementation of pyramidal Lucas-Kanade [3]. We ran both algorithms with identical parameters on four image sequences (Rubber Whale, Hydrangea, Venus, and Dimetrodon) recently proposed for comparing optical flow algorithms [2]. These sequences include static scenes as well as independently moving rigid and non-rigid objects. For all images we detected 1000 features and tracked using a 7×7 window, with ten maximum iterations and three pyramid levels (i.e., the original image plus two images obtained by downsampling the original by a factor of two and four, respectively). No color information was used by either algorithm.

The results of the experiment are shown in Table 1. For both algorithms, we computed the average angular error (AE) and the average endpoint error (EP) [2]. In all cases, the joint tracking algorithm considerably outperforms the traditional approach, oftentimes reducing the error by nearly half. More recent techniques, such as [22, 12], would perform similarly to the traditional approach on these images since they also do not enforce spatial continuity. It is also worth noting that the errors of the joint tracking algorithm are significantly less than those of the leading dense optical flow algorithms, which achieve 9.26 (AE) and 0.35 (EP) on Dimetrodon and 7.64 (AE) and 0.51 (EP) on Venus.[1]

Figure 1 displays the results of the two algorithms on three of the image sequences. In general, the joint tracking algorithm exhibits smoother flows and is thus better equipped to handle features without sufficient local information. In particular, repetitive textures that cause individual features to be distracted by similar nearby patterns using the traditional algorithm do not pose a problem for the proposed algorithm. A close-up showing this behavior is in Figure 2.

The difference between the two algorithms is even more pronounced when the scene does not contain much texture, as is often the case in indoor man-made environments. Figure 3 shows one such scene, along with the results computed by the two algorithms. In this sequence the camera is moving down and to the right with a slight counterclockwise rotation. The camera gain control causes a severe intensity change in the window of the door, causing those features to be lost. The joint algorithm is able to compute accurate flow vectors for features that do not contain sufficient local information to be accurately tracked, while the traditional algorithm fails in these locations.[2]

6. Conclusion

In this paper we have combined the ideas of Lucas-Kanade and Horn-Schunck in the opposite manner as that of Bruhn *et al*. [4]. Instead of aggregating local information to improve global flow, we aggregate global information to improve the tracking of sparse feature points. Because of their

[1] `http://vision.middlebury.edu/flow`

[2] For more results and videos of this sequence, see `www.ces.clemson.edu/~stb/research/jointtracking`.

Algorithm	Rubber Whale AE	Rubber Whale EP	Hydrangea AE	Hydrangea EP	Venus AE	Venus EP	Dimetrodon AE	Dimetrodon EP
Standard LK [3]	8.09	0.44	7.65	0.57	8.56	0.63	2.40	0.13
Joint LK (this paper)	4.32	0.13	6.13	0.45	4.66	0.25	1.34	0.08

Table 1. Quantitative comparison of the two algorithms on images with ground truth, showing the average angular error (AE) in degrees and the average endpoint error (EP) in pixels.

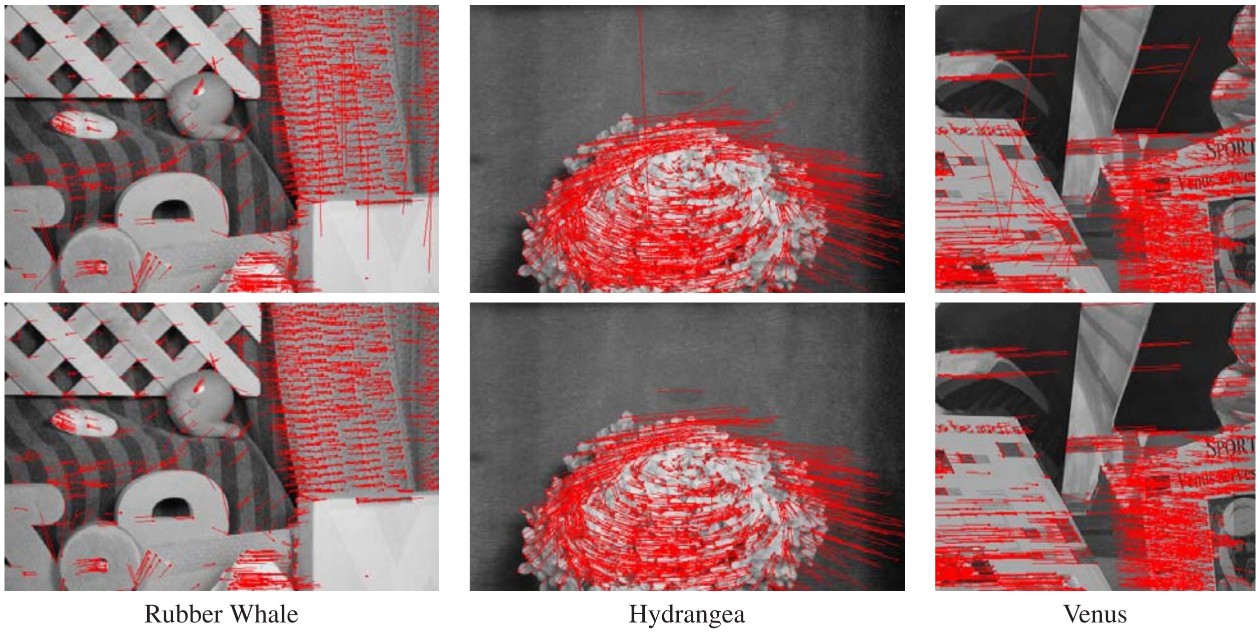

Figure 1. Results of standard Lucas-Kanade (top) and joint Lucas-Kanade (bottom) on image pairs from three sequences. Red dots indicate tracked feature points, and red lines show the displacements (scaled for display). The latter algorithm exhibits fewer erroneous displacements in regions of repetitive texture.

sparsity, the motion displacements of neighboring features cannot simply be averaged as is commonly done. Rather, an affine motion model is fit to the neighboring features, and the resulting expected flow vector is used in performing the Newton-Raphson iterations for computing the displacement of a particular feature. By incorporating off-diagonal elements into the otherwise block-diagonal tracking matrix, significantly improved results are obtained, particularly in areas of repetitive texture, one-dimensional texture (edges), or no texture.

Many improvements are possible with this work. Most notably, the smoothing of motion displacements across motion discontinuities will create artifacts in the resulting flow fields. To solve this problem, robust penalty functions or segmentation algorithms can be employed. In addition, explicit modeling of occlusions would reduce the effects of drastic unpredictable changes within the feature window itself due to the appearance and disappearance of surfaces. Finally, the displacements of sparse feature points can be used as a starting point for interpolating a dense flow field. We are exploring these ideas in our current research.

References

[1] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[2] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *Proceedings of the International Conference on Computer Vision*, 2007.

[3] J.-Y. Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. OpenCV documentation, Intel Corporation, Microprocessor Research Labs, 1999.

[4] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[5] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the International Conference on Computer Vision*, pages 1071–1076, 1995.

[6] D. DeCarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2):99–127, July 2000.

[7] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE*

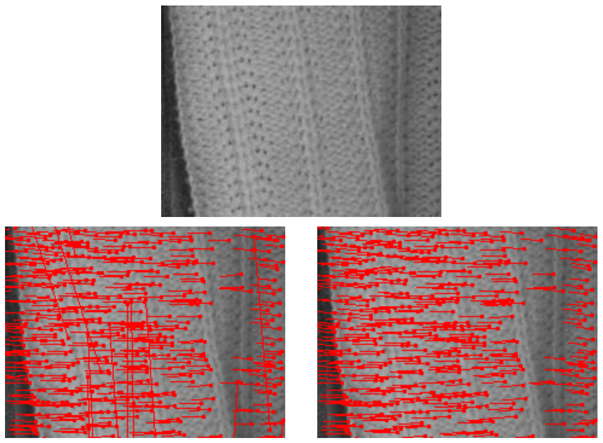

Figure 2. TOP: A close-up of the repetitive texture from the top-right corner of the Rubber Whale image. BOTTOM: The results of the two algorithms (motion vectors are scaled for display). The actual motion is between 0.78 and 1.3 pixels to the left with almost no vertical motion (less than 0.1 pixels). With the standard algorithm (left), many pixels move ten or more pixels in the vertical direction, while the joint algorithm accurately estimates the motion of all the pixels.

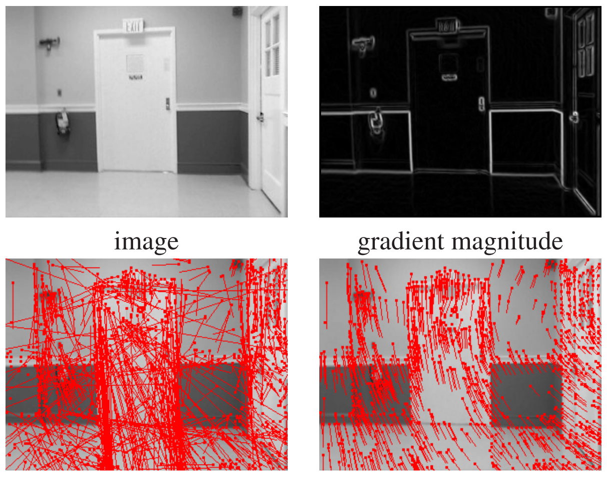

Figure 3. Comparison of the two algorithms on a relatively untextured scene. The standard algorithm computes erroneous results for many features, while the joint algorithm computes accurate flow vectors even for features on intensity edges or in untextured regions.

Transactions on Pattern Analysis and Machine Intelligence, 20(10):1025–1039, Oct. 1998.

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

[9] H. Jin, P. Favaro, and S. Soatto. Real-time 3-D motion and structure of point features: Front-end system for vision-based control and interaction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[10] H. Jin, P. Favaro, and S. Soatto. Real-time feature tracking and outlier rejection with changes in illumination. In *Proceedings of the International Conference on Computer Vision*, 2001.

[11] N. K. Kanhere, S. J. Pundlik, and S. T. Birchfield. Vehicle segmentation and tracking from a low-angle off-axis camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1152–1157, June 2005.

[12] S. J. Kim, J.-M. Frahm, and M. Pollefeys. Joint feature tracking and radiometric calibration from auto-exposure video. In *Proceedings of the International Conference on Computer Vision*, 2007.

[13] R. Lublinerman, M. Sznaier, and O. Camps. Dynamics based robust motion segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1176–1184, 2006.

[14] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[15] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, June 2004.

[16] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, modeling, and matching video clips containing multiple moving objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 477–491, 2004.

[17] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[18] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *Proceedings of the European Conference on Computer Vision*, volume 2, pages 85–98, 2004.

[19] 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, 1992.

[20] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto. Making good features track better. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[21] L. Torresani, D. Yang, G. Alexander, and C. Bregler. Tracking and modelling non-rigid objects with rank constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[22] S. Šegvić, A. Remazeilles, and F. Chaumette. Enhancing the point feature tracker by adaptive modelling of the feature support. In *Proceedings of the European Conference on Computer Vision*, pages 112–124, 2006.

[23] S. Zhilong and R. Qiuqi. Image registration based on KLT feature tracker in image mosaicing application. In *Proceedings of the 5th International Conference on Signal Processing (ICSP)*, volume 2, pages 1143–1146, 2000.