Good Features to Track

Jianbo Shi
Carlo Tomasi*

TR 93-1399
November 1993

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

* This research was supported by the National Science Foundation under contract IRI-9201751.

Good Features to Track

Jianbo Shi and Carlo Tomasi[1]

December 1993

[1]This research was supported by the National Science Foundation under contract IRI-9201751.

Abstract

No feature-based vision system can work until good features can be identified and tracked from frame to frame. Although tracking itself is by and large a solved problem, selecting features that can be tracked well and correspond to physical points in the world is still an open problem. We propose a feature selection criterion that is optimal by construction because is based on how the tracker works, as well as a feature monitoring method that can detect occlusions, disocclusions, and features that do not correspond to points in the world. These methods are based on a new tracking algorithm that extends previous Newton-Raphson style search methods to work under affine image transformations. We test performance with several simulations and experiments on real images.

Chapter 1

Introduction

Is feature tracking a solved problem? The extensive studies of image correlation [CL74], [CR76], [RGH80], [Woo83], [FP86], [TH86] and sum-of-squared-difference (SSD) methods [BYX82], [Ana89] show that all the basics are in place. When image displacements are small from one frame to the next, a window can be tracked by optimizing some matching criterion [LK81], [Ana89] over all possible small translations and, in more sophisticated systems, over all moderate linear image deformations [F$\ddot{8}$7], [FM91], [MO93]. Furthermore, feature windows can be *selected* by maximizing an interest criterion, or some measure of texturedness or cornerness in the first image. Favorite criteria are a high standard deviation in the spatial intensity profile [Mor80], the presence of zero crossings of the Laplacian of the image intensity [MPU79], and corners [KR80], [DN81]. Finally, even the size of the window to be tracked can be selected adaptively based on local variations of image intensity and inter-frame disparity [OK92].

Yet a nagging problem remains open. In fact, even a region of high interest or rich texture content can be poor. For instance, it can straddle a depth discontinuity or the boundary of a reflection highlight. In either case, the window is not attached to a fixed point in the world, making that feature useless or more likely harmful to most structure-from-motion algorithms. This phenomenon occurs very often. Extreme but typical examples are trees and cars. In a tree, branches at different depths and orientations create intersections in the image that would trigger any feature detector and yet correspond to no physical point in the world. With a car, most features on the body and windows are reflections that change their position on the surface as the car drives past the camera. Even in carefully engineered imaging situations, the problem of poor features is so pervasive that good features must often be picked by hand. Furthermore, even good features can be occluded by nearer surfaces, and trackers often blissfully drift away from their original point in the world when this occurs. No vision system based on feature tracking can be claimed to really work until these issues have been settled.

In this report we show how to monitor the quality of image features during tracking. Specifically, we investigate a measure of feature *dissimilarity* that quantifies how much the appearance of a feature changes between the first and the current frame. The idea is straightforward: dissimilarity is the feature's rms residue between the first and the current frame, and when dissimilarity grows too large the feature should be abandoned. However, in this report we make two main contributions to this problem. First, we provide experimental evidence that pure translation is not an adequate model for image motion when measuring dissimilarity, but affine image changes, that is, linear warping and translation, are adequate. Second, we propose a numerically sound and efficient way of determining affine changes by a Newton-Raphson stile minimization procedure, much in the style of what Lucas and Kanade [LK81] do for the pure translation model.

In addition to these two main contributions, we improve tracking in two more ways. First, we propose a more principled way to select features than the more traditional "interest" or "cornerness" measures. Specifically, we show that feature windows with good texture properties can be defined by explicitly optimizing the tracker's accuracy. In other words, the right features are exactly those that make the tracker work best. Second, we submit that two models of image motion are better than one. In fact, pure translation gives more stable and reliable results than affine changes when the inter-frame camera translation is small. On the other hand, affine changes are necessary to compare distant frames as is done when determining dissimilarity.

In the next chapter, we introduce affine image changes and pure translation as our two models for image motion. In chapter 3 we describe our method for the computation of affine image changes. Then, in chapters 4 and 5, we discuss our measure of texturedness and feature dissimilarity, which are based on the definition of the tracking method given in chapter 3. We discuss simulations and experiments on real sequences in chapters 6 and 7, and conclude in chapter 8.

Chapter 2

Two Models of Image Motion

In this chapter, we introduce two models of image motion: the more general affine motion is a combination of translation and linear deformation, and will be described first. The second model, pure translation, is the restriction of the general model to zero deformation.

As the camera moves, the patterns of image intensities change in a complex way. In general, any function of three variables $I(x, y, t)$, where the space variables x, y and the time variable t are discrete and suitably bounded, can represent an image sequence. However, images taken at near time instants are usually strongly related to each other, because they refer to the same scene taken from only slightly different viewpoints.

We usually express this correlation by saying that there are patterns that move in an image stream. Formally, this means that the function $I(x, y, t)$ is not arbitrary, but satisfies the following property:

$$I(x, y, t+\tau) = I(x - \xi(x, y, t, \tau), y - \eta(x, y, t, \tau)) \ . \tag{2.1}$$

Thus, a later image taken at time $t+\tau$ can be obtained by moving every point in the current image, taken at time t, by a suitable amount. The amount of motion $\delta = (\xi, \eta)$ is called the *displacement* of the point at $\mathbf{x} = (x, y)$ between time instants t and $t + \tau$.

Even in a static environment under constant lighting, the property described by equation (2.1) is often violated. For instance, at occluding boundaries, points do not just move within the image, but appear and disappear. Furthermore, the photometric appearance of a surface changes when reflectivity is a function of the viewpoint. However, the invariant (2.1) is by and large satisfied at surface markings that are away from occluding contours. At these locations, the image intensity changes fast with x and y, and the location of this change remains well defined even in the presence of moderate variations of overall brightness around it.

A more pervasive problem derives from the fact that the displacement vector δ is a function of the image position $\mathbf{x}$, and variations in δ are often noticeable even within the small windows used for tracking. It then makes little sense to speak of "the" displacement of a feature window, since there are different displacements within the same window. Unfortunately, one cannot just shrink windows to single pixels to avoid this difficulty. In fact, the value of a pixel can both change due to noise and be confused with adjacent pixels, making it hard or impossible to determine where the pixel went in the subsequent frame.

A better alternative is to enrich the description of motion within a window, that is, to define a set of possible displacement functions $\delta(\mathbf{x})$, for given t and τ, that includes more than just constant functions of $\mathbf{x}$. An *affine motion field* is a good compromise between simplicity and flexibility:

$$\delta = D\mathbf{x} + \mathbf{d}$$

where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}$$

is a deformation matrix, and $\mathbf{d}$ is the translation of the feature window's center. The image coordinates $\mathbf{x}$ are measured with respect to the window's center. Then, a point $\mathbf{x}$ in the first image I moves to point $A\mathbf{x} + \mathbf{d}$ in the second image J, where

$$A = \mathbf{1} + D$$

and $\mathbf{1}$ is the 2×2 identity matrix. Thus, the affine motion model can be summarized by the following equation relating image intensities:

$$J(A\mathbf{x} + \mathbf{d}) = I(\mathbf{x}) \ . \tag{2.2}$$

Given two images I and J and a window in image I, tracking means determining the six parameters that appear in the deformation matrix D and displacement vector $\mathbf{d}$. The quality of this estimate depends on the size of the feature window, the texturedness of the image within it, and the amount of camera motion between frames. When the window is small, the matrix D is harder to estimate, because the variations of motion within it are smaller and therefore less reliable. However, smaller windows are in general preferable for tracking because they are more likely to contain features at similar depths, and therefore correspond to small patches in the world, rather than to pairs of patches in different locations as would be the case along a depth discontinuity. For this reason, a *pure translation* model is preferable during tracking:

$$J(\mathbf{x} + \mathbf{d}) = I(\mathbf{x})$$

where the deformation matrix D is assumed to be zero.

The experiments in chapters 6 and 7 show that the best combination of these two motion models is pure translation for tracking, because of its higher reliability and accuracy over the small inter-frame motion of the camera, and affine motion for comparing features between the first and the current frame in order to monitor their quality. In order to address these issues quantitatively, however, we first need to introduce our tracking method. In fact, we define texturedness based on how tracking works. Rather than the more *ad hoc* definitions of interest operator or "cornerness", we define a feature to have a good texture content if the feature can be tracked well.

Chapter 3

Computing Image Motion

The affine motion model is expressed by equation (2.2):

$$J(A\mathbf{x}+\mathbf{d}) = I(\mathbf{x})$$

and for pure translation the matrix A is assumed to be equal to the identity matrix. Because of image noise and because the affine motion model is not perfect, the equation above is in general not satisfied exactly. The problem of determining the motion parameters can then be defined as that of finding the A and $\mathbf{d}$ that minimize the *dissimilarity*

$$\epsilon = \int\int_W [J(A\mathbf{x}+\mathbf{d}) - I(\mathbf{x})]^2 \, w(\mathbf{x})\, d\mathbf{x} \tag{3.1}$$

where W is the given feature window and $w(\mathbf{x})$ is a weighting function. In the simplest case, $w(\mathbf{x}) = 1$. Alternatively, w could be a Gaussian-like function to emphasize the central area of the window. Under pure translation, the matrix A is constrained to be equal to the identity matrix. In the following, we first look at the unconstrained problem, which we solve by means of a Newton-Raphson style iterative search for the optimal values of $A = \mathbf{1} + D$ and $\mathbf{d}$. The case of pure translation is then obtained as a specialization.

To minimize the residual (3.1), we differentiate it with respect to the unknown parameters in the deformation matrix D and the displacement vector $\mathbf{d}$ and set the result to zero. This yields the following two matrix equations:

$$\frac{1}{2}\frac{\partial \epsilon}{\partial D} = \int\int_W [J(A\mathbf{x}+\mathbf{d}) - I(\mathbf{x})]\, \mathbf{g}\mathbf{x}^T \, w\, d\mathbf{x} = 0 \tag{3.2}$$

$$\frac{1}{2}\frac{\partial \epsilon}{\partial \mathbf{d}} = \int\int_W [J(A\mathbf{x}+\mathbf{d}) - I(\mathbf{x})]\, \mathbf{g}\, w\, d\mathbf{x} = 0 \tag{3.3}$$

where

$$\mathbf{g} = \left(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y}\right)^T$$

is the spatial gradient of the image intensity and the superscipt T denotes transposition.

If the image motion

$$\mathbf{u} = D\mathbf{x} + \mathbf{d} \tag{3.4}$$

could be assumed to be small, the term $J(A\mathbf{x}+\mathbf{d})$ could be approximated by its Taylor series expansion truncated to the linear term:

$$J(A\mathbf{x}+\mathbf{d}) = J(\mathbf{x}) + \mathbf{g}^T(\mathbf{u}) \tag{3.5}$$

which, combined with equations (3.2) and (3.3) would yield the following systems of equations:

$$\int\int_W \mathbf{g}\mathbf{x}^T(\mathbf{g}^T\mathbf{u})\, w\, d\mathbf{x} = \int\int_W [I(\mathbf{x}) - J(\mathbf{x})]\, \mathbf{g}\mathbf{x}^T\, w\, d\mathbf{x} \tag{3.6}$$

$$\int\int_W \mathbf{g}(\mathbf{g}^T\mathbf{u})\, w\, d\mathbf{x} = \int\int_W [I(\mathbf{x}) - J(\mathbf{x})]\, \mathbf{g}\, w\, d\mathbf{x}\,. \tag{3.7}$$

Even when affine motion is a good model, these equations are only approximately satisfied, because of the linearization of equation (3.5). However, equations (3.6) and (3.7) can be solved iteratively as follows. Let

$$D_0 = \mathbf{1} \qquad \mathbf{d}_0 = \mathbf{0} \qquad J_0 = J(\mathbf{x})$$

be the initial estimates of deformation D and displacement $\mathbf{d}$ and the initial intensity function $J(\mathbf{x})$. At the i-th iteration, equations (3.6) and (3.7) can be solved with $J(\mathbf{x})$ replaced by J_{i-1} to yield the new values

$$D_i \qquad \mathbf{d}_i \qquad J_i = J_{i-1}(A_i\mathbf{x} + \mathbf{d}_i)$$

where $A_i = \mathbf{1} + D_i$ is the transformation between images J_{i-1} and J_i. To make this computation more explicit, we can write equations (3.6) and (3.7) in a more compact form by factoring out the unknowns D and $\mathbf{d}$. This yields (see Appendix A) the following linear 6×6 system:

$$T\mathbf{z} = \mathbf{a} \tag{3.8}$$

where

$$T = \int\int_W \begin{bmatrix} x^2 g_x^2 & x^2 g_x g_y & xy g_x^2 & xy g_x g_y & x g_x^2 & x g_x g_y \\ x^2 g_x g_y & x^2 g_y^2 & xy g_x g_y & xy g_y^2 & x g_x g_y & x g_y^2 \\ xy g_x^2 & xy g_x g_y & y^2 g_x^2 & y^2 g_x g_y & y g_x^2 & y g_x g_y \\ xy g_x g_y & xy g_y^2 & y^2 g_x g_y & y^2 g_y^2 & y g_x g_y & y g_y^2 \\ x g_x^2 & x g_x g_y & y g_x^2 & y g_x g_y & g_x^2 & g_x g_y \\ x g_x g_y & x g_y^2 & y g_x g_y & y g_y^2 & g_x g_y & g_y^2 \end{bmatrix} w\, d\mathbf{x}$$

is a matrix that can be computed from one image,

$$\mathbf{z} = \begin{bmatrix} d_{xx} \\ d_{yx} \\ d_{xy} \\ d_{yy} \\ d_x \\ d_y \end{bmatrix}$$

is a vector that collects the unknown entries of the deformation D and displacement $\mathbf{d}$, and

$$\mathbf{a} = \int\int_W [I(\mathbf{x}) - J(\mathbf{x})] \begin{bmatrix} x g_x \\ x g_y \\ y g_x \\ y g_y \\ g_x \\ g_y \end{bmatrix} w\, d\mathbf{x}$$

is an error vector that depends on the difference between the two images.

The symmetric 6×6 matrix T can be partitioned as follows:

$$T = \int\int_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} w\, d\mathbf{x} \tag{3.9}$$

where U is 4×4, Z is 2×2, and V is 4×2.

During tracking, the affine deformation D of the feature window is likely to be small, since motion between adjacent frames must be small in the first place for tracking to work at all. It is then safer to set D to the zero matrix. In fact, attempting to determine deformation parameters in this situation is not only useless but can lead to poor displacement solutions: in fact, the deformation D and the displacement $\mathbf{d}$ interact through the 4×2 matrix V of equation (3.9), and any error in D would cause errors in $\mathbf{d}$. Consequently, when the goal is to determine $\mathbf{d}$, the smaller system

$$Z\mathbf{d} = \mathbf{e} \tag{3.10}$$

should be solved, where $\mathbf{e}$ collects the last two entries of the vector $\mathbf{a}$ of equation (3.8).

When monitoring features for dissimilarities in their appearance between the first and the current frame, on the other hand, the full affine motion system (3.8) should be solved. In fact, motion is now too large to be described well by the pure translation model. Furthermore, in determining dissimilarity, the whole transformation between the two windows is of interest, and a precise displacement is less critical, so it is acceptable for D and $\mathbf{d}$ to interact to some extent through the matrix V.

In the next two chapters we discuss these issues in more detail: first we determine when system (3.10) yields a good displacement measurement (chapter 4) and then we see when equation (3.8) can be used reliably to monitor a feature's quality (chapter 5).

Chapter 4

Texturedness

Regardless of the method used for tracking, not all parts of an image contain motion information. This has been known ever since the somewhat infelicitous label of *aperture problem* was attached to the issue: for instance, only the vertical component of motion can be determined for a horizontal intensity edge. To overcome this difficulty, researchers have proposed to track corners, or windows with a high spatial frequency content, or regions where some mix of second-order derivatives is sufficiently high. However, there are two problems with these "interest operators". First, they are often based on a preconceived and arbitrary idea of what a good window looks like. In other words, they are based on the assumption that good features can be defined independently of the method used for tracking them. The resulting features may be intuitive, but are not guaranteed to be the best for the tracking algorithm to produce good results. Second, "interest operators" have been usually defined for the simpler pure translation model of chapter 2, and the underlying concept are hard to extend to affine motion.

In this report, we propose a more principled definition of feature quality. Rather than introducing this notion *a priori*, we base our definition on the method used for tracking: a good window is one that can be tracked well. With this approach, a window is chosen for tracking only if it is good enough for the purpose, so that the selection criterion is optimal by construction.

To introduce our definition of a good feature, we turn to equation (3.10), the basic equation to be solved during tracking. We can track a window from frame to frame if this system represents good measurements, and if it can be solved reliably. Consequently, the symmetric 2×2 matrix Z of the system must be both above the image noise level and well-conditioned. The noise requirement implies that both eigenvalues of Z must be large, while the conditioning requirement means that they cannot differ by several orders of magnitude. Two small eigenvalues mean a roughly constant intensity profile within a window. A large and a small eigenvalue correspond to a unidirectional texture pattern. Two large eigenvalues can represent corners, salt-and-pepper textures, or any other pattern that can be tracked reliably.

In practice, when the smaller eigenvalue is sufficiently large to meet the noise criterion, the matrix Z is usually also well conditioned. This is due to the fact that the intensity variations in a window are bounded by the maximum allowable pixel value, so that the greater eigenvalue cannot be arbitrarily large.

As a consequence, if the two eigenvalues of Z are λ_1 and λ_2, we accept a window if

$$\min(\lambda_1, \lambda_2) > \lambda \ , \tag{4.1}$$

where λ is a predefined threshold.

To determine λ, we first measure the eigenvalues for images of a region of approximately uniform brightness, taken with the camera to be used during tracking. This yields a lower bound for λ. We then select a set of various types of features, such as corners and highly textured regions, to obtain an upper bound for λ. In practice, we have found that the two bounds are comfortably separate, and the value of λ, chosen halfway in-between, is not critical.

We illustrate this idea through two extreme cases. The left picture in figure 4.1 is an image window with a broad horizontal white bar on a black background. The picture on the right shows the corresponding *confidence ellipse* defined as the ellipse whose half axes have lengths λ_1, λ_2 and directions given by the corresponding eigenvectors.

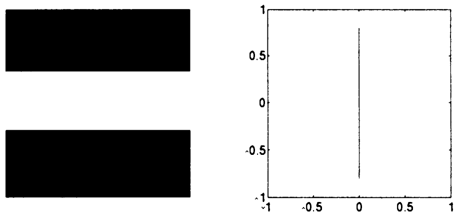

Figure 4.1: An image window with a white horizontal bar (left) and the corresponding confidence ellipse (right).

Because of the aperture problem mentioned above, a horizontal motion of the bar cannot be detected, and the horizontal axis of the confidence ellipse is correspondingly zero.

The situation is very different in figure 4.2, showing four circular blobs in the image window. Because motion of this pattern can be detected equally well in all directions, the corresponding confidence ellipse, shown on the right, is a circle.

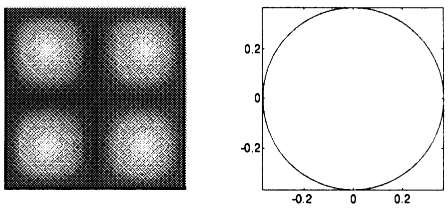

Figure 4.2: An image window with four circular blobs (left) and the corresponding confidence ellipse (right).

Similar considerations hold also when solving the full affine motion system (3.8) for the deformation D and displacement $\mathbf{d}$. However, an essential difference must be pointed out: deformations are used to determine whether the window in the first frame matches that in the current frame well enough during feature monitoring. Thus, the goal is *not* to determine deformation *per se*. Consequently, it does not matter if one component of deformation (such as horizontal scaling in figure 4.1) cannot be determined reliably. In fact, this means that that component does not affect the window substantially, and any value along this component will do in the comparison. In practice, the system (3.8) can be solved by computing the pseudo-inverse of T. Then, whenever some component is undetermined, the minimum norm solution is computed, that is, the solution with a zero deformation along the undetermined component(s).

It is still instructive, however, to consider through some examples the relative size of the eigenvectors and eigenvalues of the matrix U in equation (3.9), corresponding to the space of linear deformations of the image window. For instance, a horizontal stretch or shear of the window of figure 4.1 would pass unnoticed, while a vertical stretch or shear would not. This is illustrated in figure 4.3: the diagram on the lower left shows the four eigenvalues of the deformation matrix D.

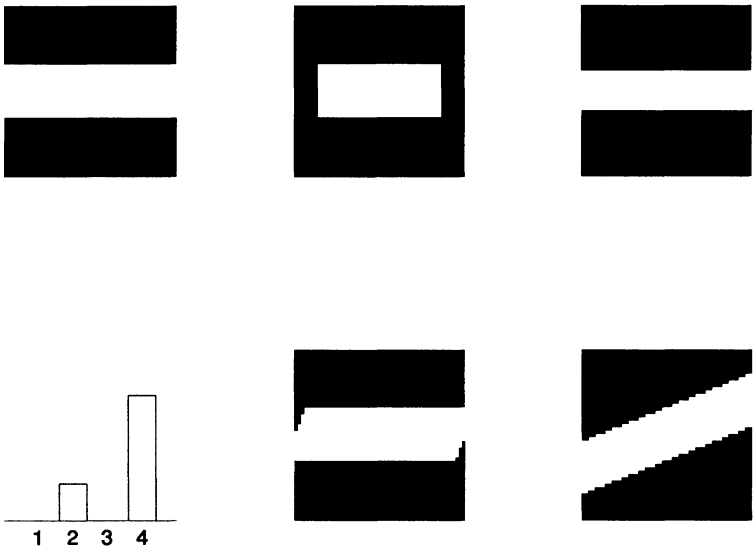

Figure 4.3: Effects of a horizontal scaling (top center), horizontal shear (bottom center), vertical scaling (top right) and vertical shear (bottom right) for the image at the top left. The diagram shows the eigenvalues of the deformation matrix.

The corresponding eigenvectors are orthogonal directions in the four-dimensional space of deformations (the four entries of D are the parameters of this space). What these directions correspond to depends on the particular image content. The four right pictures in figure 4.3 show the effects of a deformation along each of the four eigenvectors for the bar window of figure 4.1, repeated for convenience at the top left of figure 4.3. The two deformations in the middle of figure 4.3 correspond to the zero eigenvalues, so no change is visible, except for a boundary effect in the figure at the bottom center. The two corresponding eigenvectors correspond to the deformation matrices

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

which represent unit horizontal scaling and shear, respectively. The two image windows on the right of figure 4.3, on the other hand, represent changes parallel to the eigenvectors corresponding to the large eigenvalues and to the vertical scaling and shear matrices

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

respectively.

For the blobs of figure 4.2, repeated at the top left of figure 4.4, the situation is quite different.

Here, the four deformation eigenvalues are of comparable size. Overall scaling (top right) is the direction of maximum confidence (second eigenvalue in the diagram), followed by vertical scaling (top center, first eigenvalue), a combination of vertical and horizontal shear (bottom right, fourth

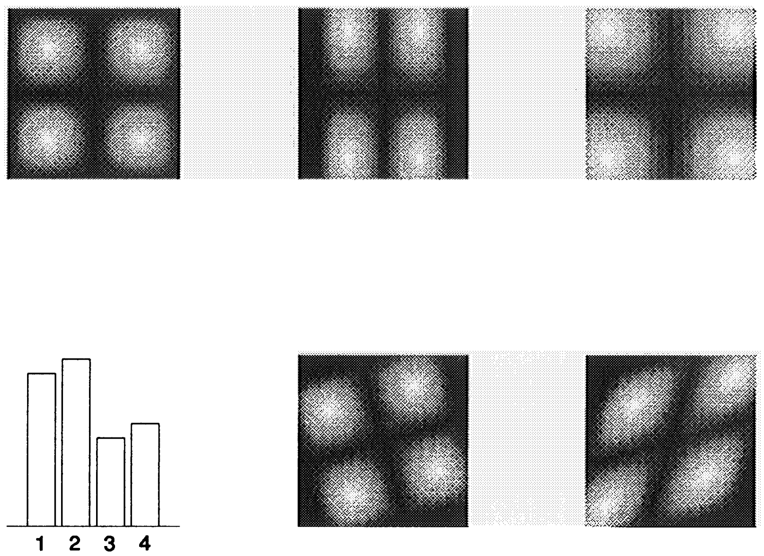

Figure 4.4: Effects of a horizontal scaling (top center), horizontal shear (bottom center), vertical scaling (top right) and vertical shear (bottom right) for the image at the top left. The diagram shows the eigenvalues of the deformation matrix.

eigenvalue), and rotation (third eigenvalue, bottom center). Because all eigenvalues are high, the deformation matrix can be computed reliably with the method described in chapter 3.

Chapter 5

Dissimilarity

A feature with a high measure of confidence, as defined in the previous chapter, can still be a bad feature to track. For instance, in an image of a tree, a horizontal twig in the foreground can intersect a vertical twig in the background. This intersection, however, occurs only in the image, not in the world, since the two twigs are at different depths. Any selection criterion would pick the intersection as a good feature to track, and yet there is no real world feature there to speak of. The problem is that image feature and world feature do not necessarily coincide, and an image window just does not contain enough information to determine whether an image feature is also a feature in the world.

The measure of *dissimilarity* defined in equation (3.1), on the other hand, can often indicate that something is going wrong. Because of the potentially large number of frames through which a given feature can be tracked, the dissimilarity measure would not work well with a pure translation model. To illustrate this, consider figure 5.1, which shows three out of 21 frame details from Woody Allen's movie, *Manhattan*.

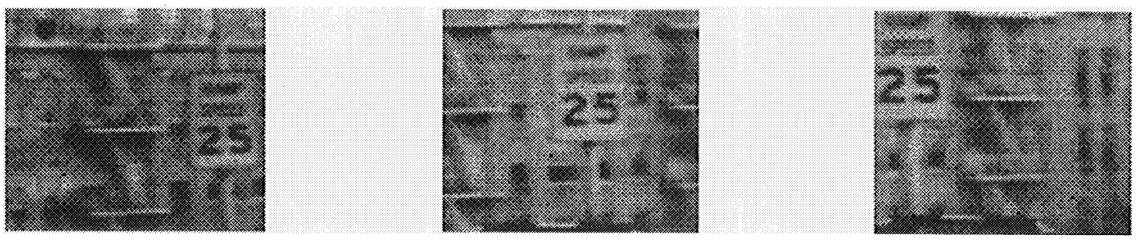

Figure 5.1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 5.2 shows the results of tracking the traffic sign in this sequence.

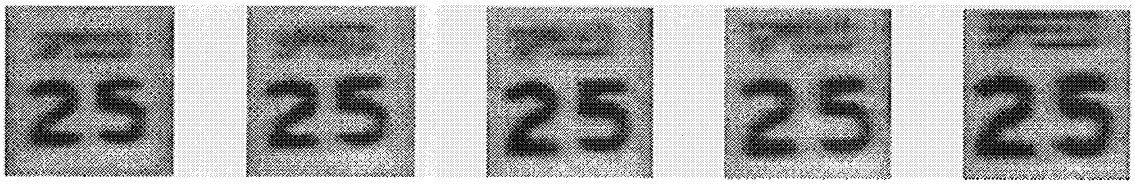

Figure 5.2: A window that tracks the traffic sign visible in the sequence of figure 5.1. Frames 1,6,11,16,21 are shown here.

While the inter-frame changes are small enough for the pure translation tracker to work, the cumulative changes over 25 frames are too large. In fact, the size of the sign increases by about 15 percent, and the dissimilarity measure (3.1) increases rather quickly with the frame number, as shown by the dashed line of figure 5.3.

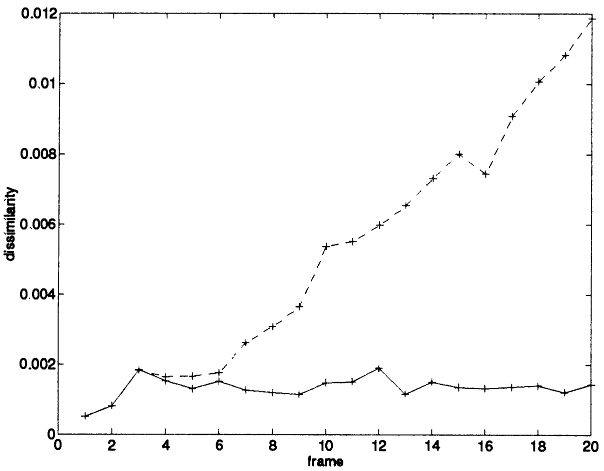

Figure 5.3: Pure translation (dashed) and affine motion (solid) dissimilarity measures for the window sequence of figure 5.2.

The solid line in the same figure, on the other hand, shows the dissimilarity measure when also deformations are accounted for, that is, if the entire system (3.8) is solved for $\mathbf{z}$. As expected, this new measure of dissimilarity remains small and roughly constant. Figure 5.4 shows the same windows as in figure 5.2, but warped by the computed deformations. The deformations make the five windows virtually equal to each other.

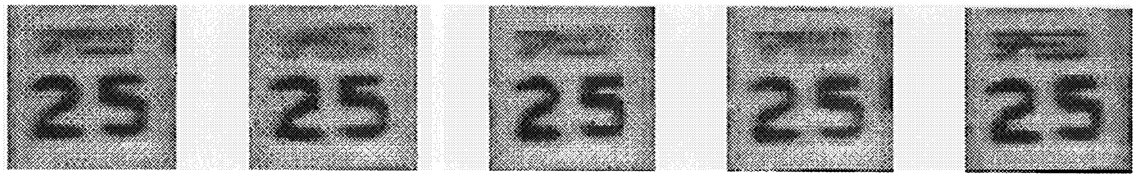

Figure 5.4: The same windows as in figure 5.2, warped by the computed deformation matrices.

Let us now look at a sequence from the same movie where something goes wrong with the feature. In figure 5.5, the feature tracked is the bright, small window on the building in the background. Figure 5.6 shows the feature window through five frames. Notice that in the third frame the traffic sign occludes the original feature.

The circled curves in figure 5.7 are the dissimilarity measures under affine motion (solid) and pure translation (dashed). The sharp jump in the affine motion curve around frame 4 indicates the occlusion. After occlusion, the first and current windows are too dissimilar from each other. Figure 5.8 shows that the deformation computation attempts to deform the traffic sign into a window.

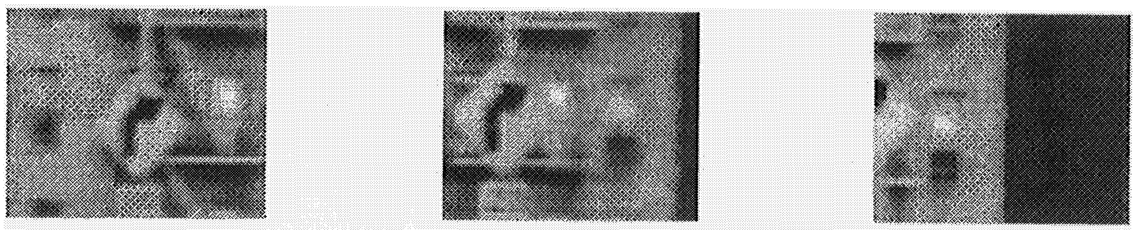

Figure 5.5: Three frame details from Woody Allen's *Manhattan*. The feature tracked is the bright window on the background, just behind the fire escape on the right of the traffic sign.

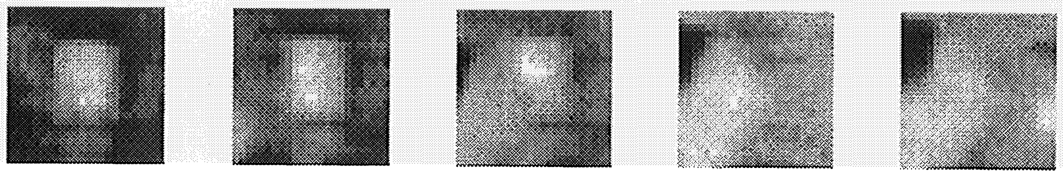

Figure 5.6: The bright window from figure 5.5, visible as the bright rectangular spot in the first frame (a), is occluded by the traffic sign (c).

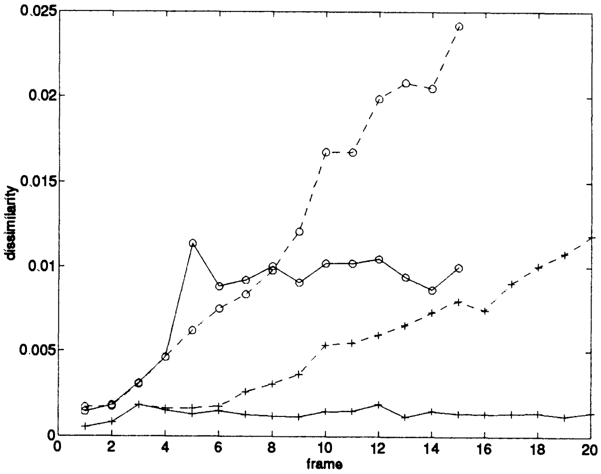

Figure 5.7: Pure translation (dashed) and affine motion (solid) dissimilarity measures for the window sequence of figure 5.1 (plusses) and 5.5 (circles).

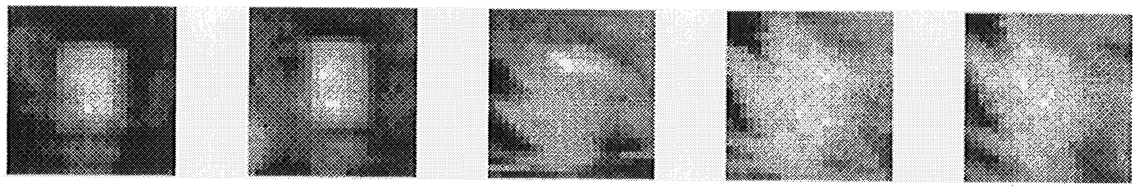

Figure 5.8: The same windows as in figure 5.6, warped by the computed deformation matrices.

Chapter 6

Simulations

In this chapter, we present simulation results to show that if the affine motion model is correct then the tracking algorithm presented in chapter 3 converges even when the starting point is far removed from the true solution.

The first series of simulations are run on the four circular blobs we considered in chapter 4 (figure 4.2). The following three motions are considered:

$$D_1 = \begin{bmatrix} 1.4095 & -0.3420 \\ 0.3420 & 0.5638 \end{bmatrix} , \quad \mathbf{d}_1 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 0.6578 & -0.3420 \\ 0.3420 & 0.6578 \end{bmatrix} , \quad \mathbf{d}_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$D_3 = \begin{bmatrix} 0.8090 & 0.2534 \\ 0.3423 & 1.2320 \end{bmatrix} , \quad \mathbf{d}_3 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

To see the effects of these motions, compare the first and last column of figure 6.1. The images in the first column are the feature windows in the first image (equal for all three simulations in this series), while the images in the last column are the images warped and translated by the motions specified above and corrupted with random Gaussian noise with a standard deviation equal to 16 percent of the maximum image intensity. The images in the intermediate columns are the results of the deformations and translations to which the tracking algorithm subjects the images in the leftmost column after 4, 8, and 19 iterations, respectively. If the algorithm works correctly, the images in the fourth column of figure 6.1 should be as similar as possible to those in the fifth column, which is indeed the case.

A more quantitative idea of the convergence of the tracking algorithm is given by figure 6.2, which plots the dissimilarity measure, translation error, and deformation error as a function of the frame number (first three columns), as well as the intermediate displacements and deformations (last two columns). Deformations are represented in the fifth column of figure 6.2 by two vectors each, corresponding to the two columns of the transformation matrix $A = \mathbf{1} + D$. Displacements and displacement errors are in pixels, while deformation errors are the Frobenius norms of the difference between true and computed deformation matrices. Table 6.1 shows the numerical values of true and computed deformations and translations.

Figure 6.3 shows a similar experiment with a more complex image, the image of a penny (available in MATLAB). Finally, figure 6.4 shows the result of attempting to match two completely different images: four blobs (leftmost column) and a cross (rightmost column). The algorithm tries to do its best by rotating the four blobs until they are aligned (fourth column) with the cross, but the dissimilarity (left plot in the bottom row of figure 6.4) remains high throughout.

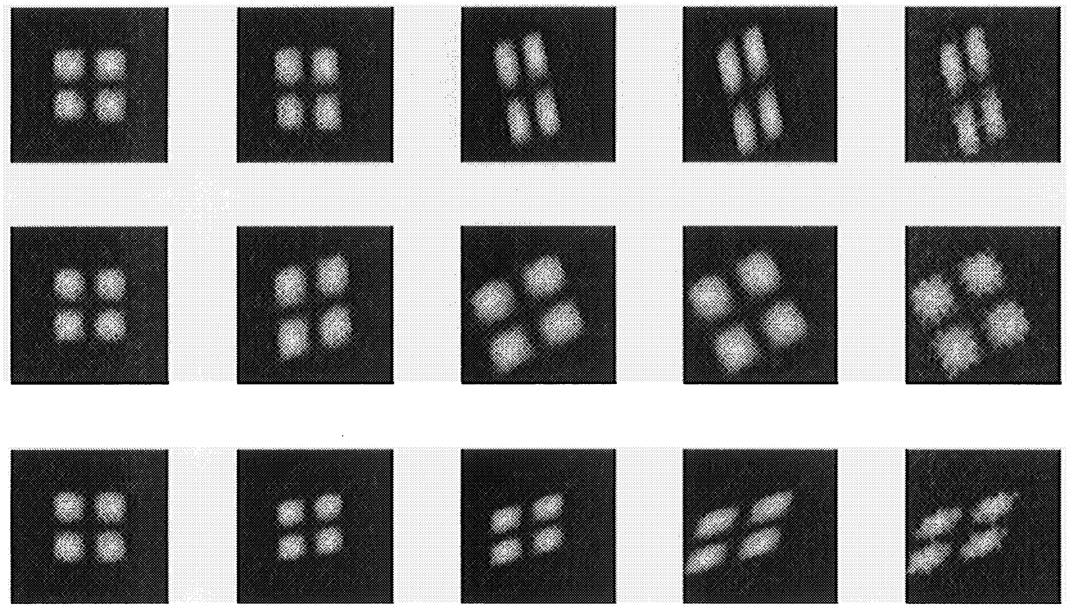

Figure 6.1: Original image (leftmost column) and image warped, translated and corrupted by noise (rightmost column) for three different motions. The intermediate columns are the images in the leftmost column deformed and translated by 4,8,and 19 iterations of the tracking algorithm.

Simulation Number	True Deformation	Computed Deformation	True Translation	Computed Translation
1	$\begin{bmatrix} 1.4095 & -0.3420 \\ 0.3420 & 0.5638 \end{bmatrix}$	$\begin{bmatrix} 1.3930 & -0.3343 \\ 0.3381 & 0.5691 \end{bmatrix}$	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 3.0785 \\ -0.0007 \end{bmatrix}$
2	$\begin{bmatrix} 0.6578 & -0.3420 \\ 0.3420 & 0.6578 \end{bmatrix}$	$\begin{bmatrix} 0.6699 & -0.3432 \\ 0.3187 & 0.6605 \end{bmatrix}$	$\begin{bmatrix} 2 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 2.0920 \\ 0.0155 \end{bmatrix}$
3	$\begin{bmatrix} 0.8090 & 0.2534 \\ 0.3423 & 1.2320 \end{bmatrix}$	$\begin{bmatrix} 0.8018 & 0.2350 \\ 0.3507 & 1.2274 \end{bmatrix}$	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 3.0591 \\ 0.0342 \end{bmatrix}$

Table 6.1: Comparison of true and computed deformations and displacements (in pixels) for the three simulations illustrated in figures 6.1 and 6.2.

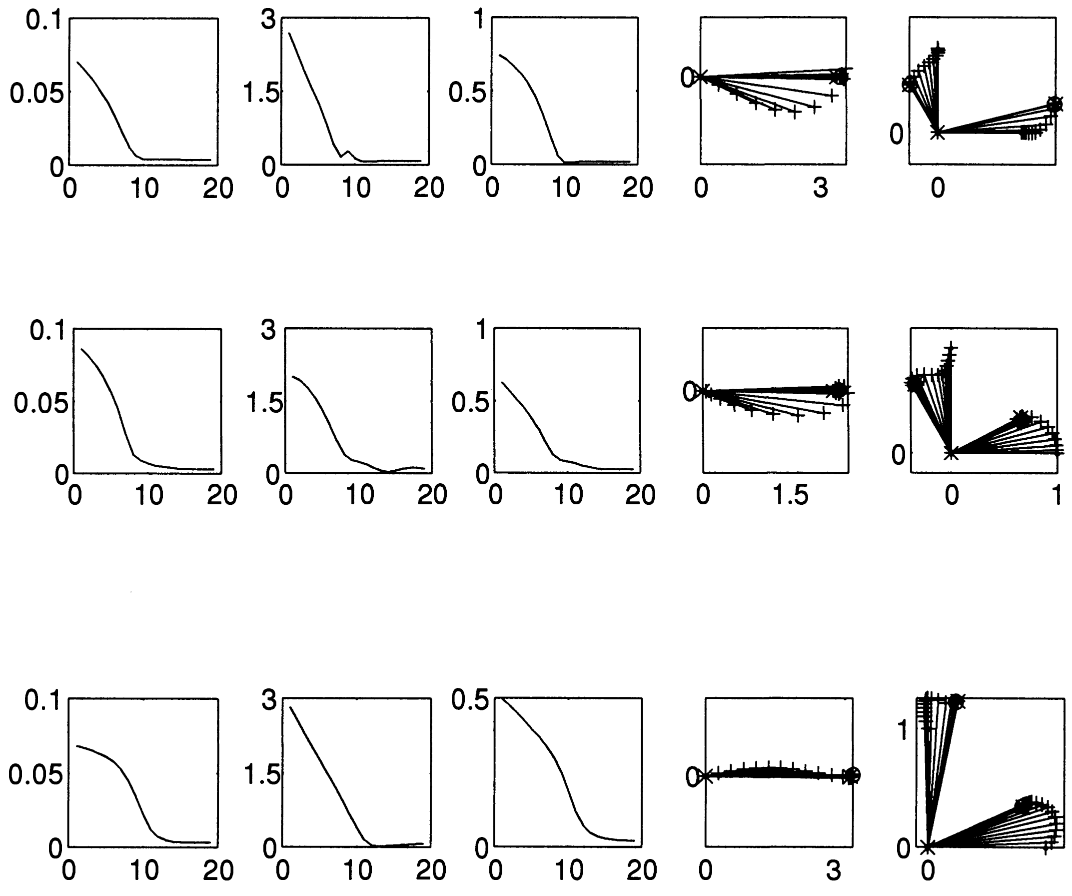

Figure 6.2: The first three columns show the dissimilarity, displacement error, and deformation error as a function of the tracking algorithm's iteration number. The last two columns are displacements and deformations computed during tracking, starting from zero displacement and deformation.

Chapter 7

More Experiments on Real Images

The experiments and simulations of the previous chapters illustrate specific points and issues concerning feature monitoring and tracking. Whether the feature selection and monitoring proposed in this report are useful, on the other hand, can only be established by more extensive experiments on more images and larger populations of features. The experiments in this chapter are a step in that direction.

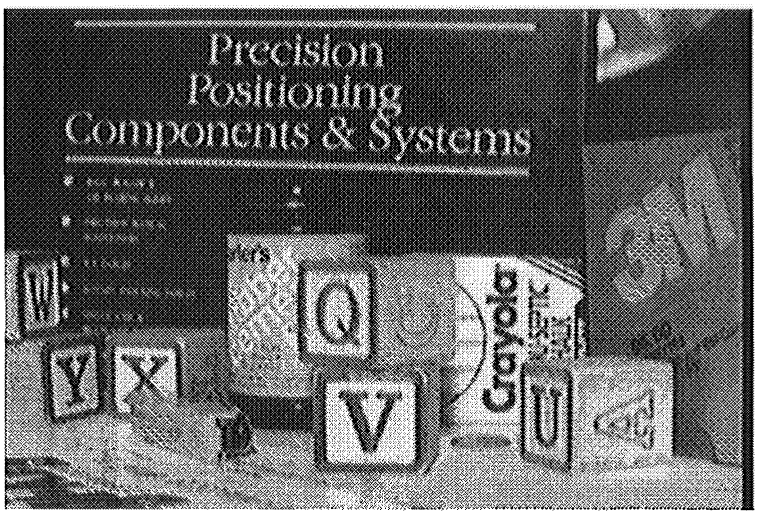

Figure 7.1: The first frame of a 26 frame sequence taken with a forward moving camera.

Figure 7.1 shows the first frame of a 26-frame sequence. The Pulnix camera is equipped with a 16mm lens and moves forward 2mm per frame. Because of the forward motion, features loom larger from frame to frame. The pure translation model is sufficient for inter-frame tracking but not for a useful feature monitoring, as discussed below. Figure 7.2 displays the 102 features selected according to the criterion introduced in chapter 4. To limit the number of features and to use each portion of the image at most once, the constraint was imposed that no two feature windows can overlap in the first frame. Figure 7.3 shows the dissimilarity of each feature under the pure translation motion model, that is, with the deformation matrix D set to zero for all features. This dissimilarity is nearly useless: except for features 58 and 89, all features have comparable dissimilarities, and no clean discrimination can be drawn between good and bad features.

From figure 7.4 we see that features 58 is at the boundary of the block with a letter **U** visible in the lower right-hand side of the figure. The feature window straddles the vertical dark edge of the

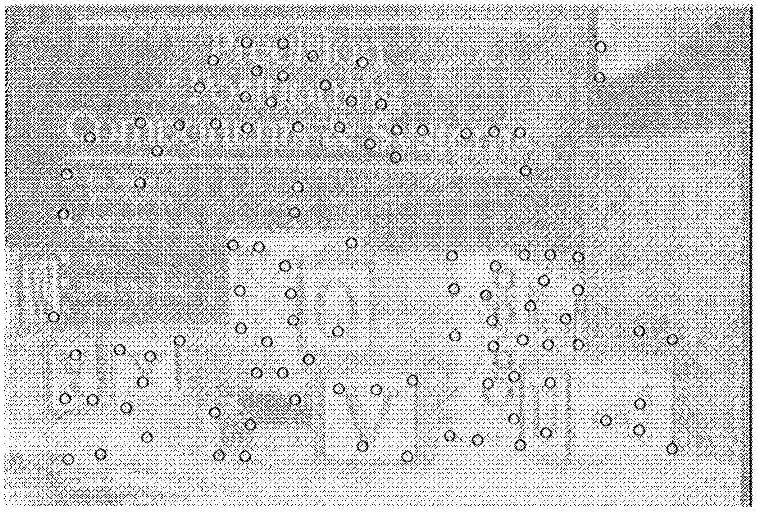

Figure 7.2: The features selected according to the texturedness criterion of chapter 4.

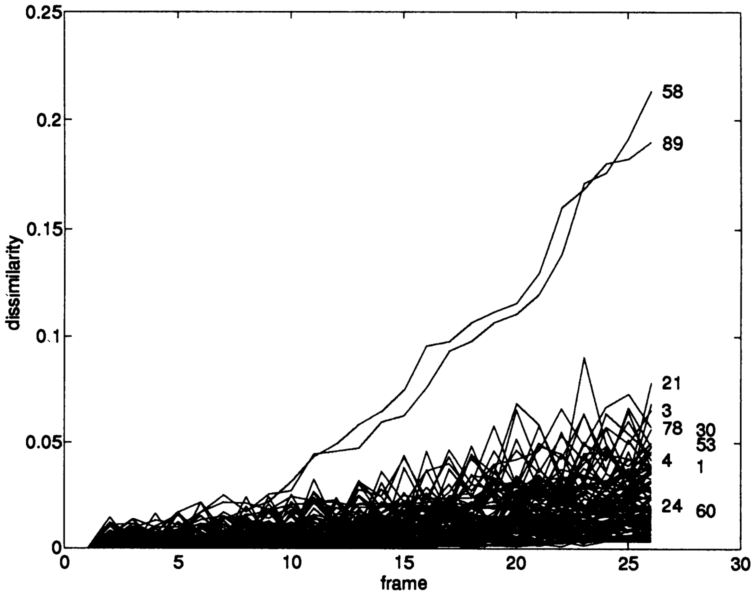

Figure 7.3: Pure translation dissimilarity for the features in figure 7.2. This dissimilarity is nearly useless for feature discrimination.

block in the foreground as well as parts of the letters **Cra** in the word "Crayola" in the background. Six frames of this window are visible in the third row of figure 7.5. As the camera moves forward, the pure translation tracking stays on top of approximately the same part of the image. However, the gap between the vertical edge in the foreground and the letters in the background widens, and a deformation of the current window into the window in the first frame becomes harded and harder, leading to the rising dissimilarity. The changes in feature 89 are seen even more easily. This feature is between the edge of the book in the background and a lamp partially visible behind it in the top right corner of figure 7.4. As the camera moves forward, the shape of the glossy reflection on the lamp shade changes and is more occluded at the same time: see the last row of figure 7.5.

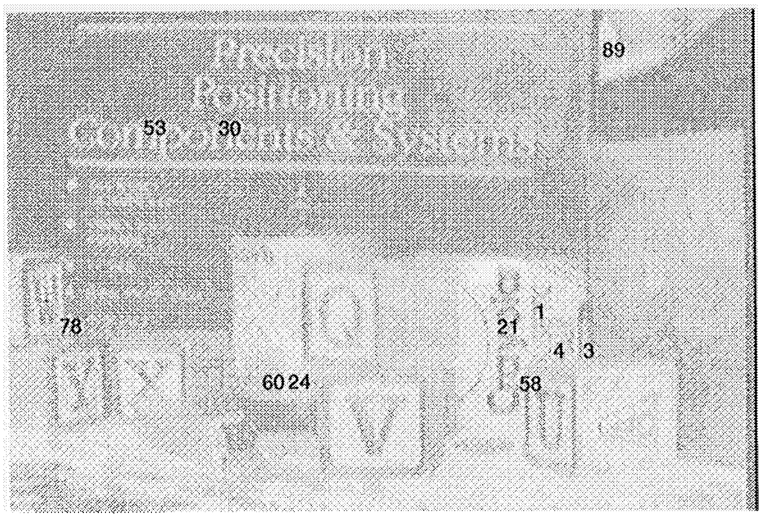

Figure 7.4: Labels of some of the features in figure 7.2.

However, many other features are bad in this frame. For instance, feature 3 in the lower right of figure 7.4 is affected by a substantial disocclusion of the lettering on the Crayola box by the **U** block as the camera moves forward, as well as a slight disocclusion by the "3M" box on the right. Yet the dissimilarity of feature 3 is not substantially different from that of all the other features in figure 7.3. This is because the degradation of the dissimilarity caused by the camera's forward motion is dominant, and reflects in the overall upward trend of the majority of curves in figure 7.3. Similar considerations hold, for instance, for features 78 (a disocclusion), 24 (an occlusion), and 4 (a disocclusion) labeled in figure 7.4.

Now compare the pure translation dissimilarity of figure 7.3 with the affine motion dissimilarity of figure 7.6. The thick stripe of curves at the bottom represents all good features, including features 1,21,30,53. From figure 7.4, these four features can be seen to be good, since they are immune from occlusions or glossy reflections: 1 and 21 are lettering on the "Crayola" box (the second row of figure 7.5 shows feature 21 as an example), while features 30 and 53 are details of the large title on the book in the background (upper left in figure 7.4). The bad features 3,4,58,78,89, on the other hand, stand out very clearly in figure 7.6: discrimination is now possible.

Features 24 and 60 deserve a special discussion, and are plotted with dashed lines in figure 7.6. These two features are lettering detail on the rubber cement bottle in the lower center of figure 7.4. The fourth row of figure 7.5 shows feature 60 as an example. Although feature 24 has an additional slight occlusion as the camera moves forward, these two features stand out from the very beginning, that is, even for very low frame numbers in figure 7.6, and their dissimilarity curves are very erratic throughout the sequence. This is because of aliasing: from the fourth row of figure 7.5,

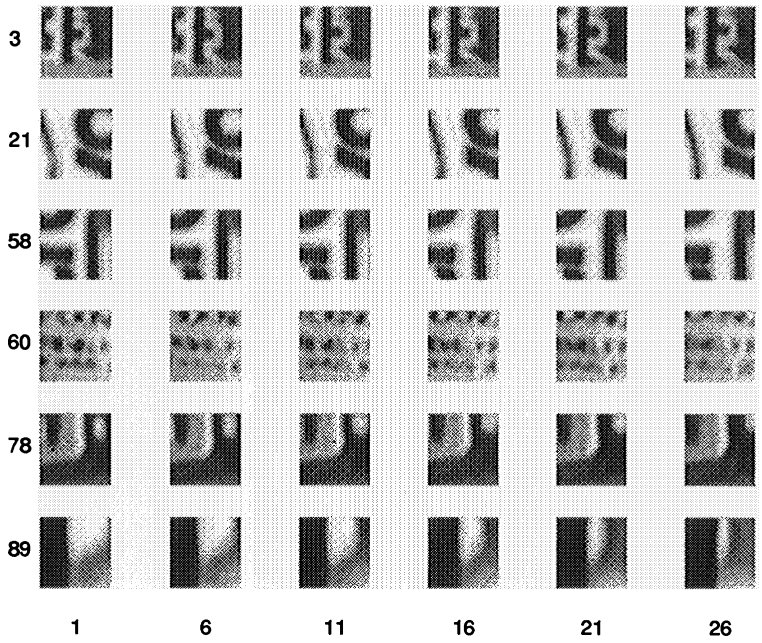

Figure 7.5: Six sample features through six sample frames.

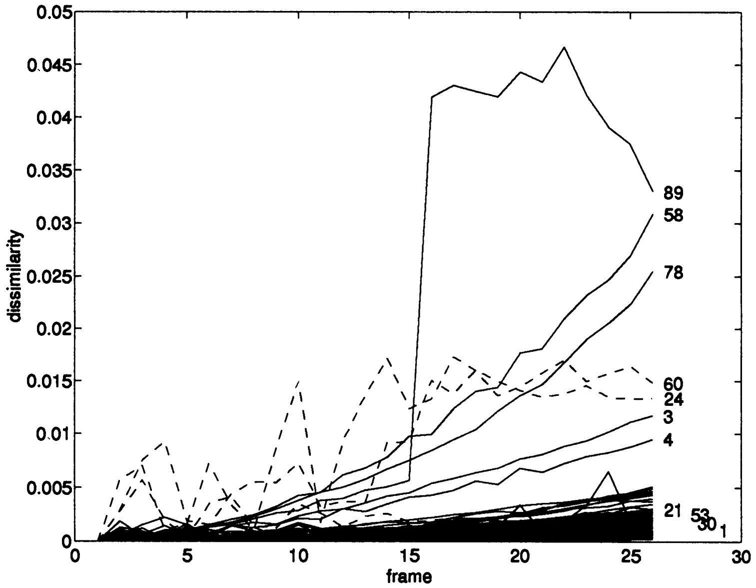

Figure 7.6: Affine motion dissimilarity for the features in figure 7.2. Notice the good discrimination between good and bad features. Dashed plots indicate aliasing (see text).

we see that feature 60 (and similarly feature 24) contains very small lettering, of size comparable to the image's pixel size (the feature window is 25×25 pixels). The matching between one frame and the next is haphazard, because the characters in the lettering are badly aliased. This behavior is not a problem: erratic dissimilarities indicate trouble, and the corresponding features ought to be abandoned.

Chapter 8

Conclusion

In this report, we have proposed both a method for feature selection and a technique for feature monitoring during tracking. Selection specifically maximizes the quality of tracking, and is therefore optimal by construction, as opposed to more *ad hoc* measures of texturedness. Monitoring is computationally inexpensive and sound, and makes it possible to discriminate between good and bad features based on a measure of dissimilarity that uses affine motion as the underlying image change model.

Of course, monitoring feature dissimilarity does not solve all the problems of tracking. In some situations, a bright spot on a glossy surface is a bad (that is, nonrigid) feature, but may change little over a long sequence: dissimilarity may not detect the problem. However, it must be realized that not everything can be decided locally. For the case in point, rigidity is not a local feature, so a local method cannot be expected to detect its violation. On the other hand, many problems can indeed be discovered locally and these are the target of the investigation in this report. The many illustrative experiments and simulations, as well as the large experiment of chapter 7, show that monitoring is indeed effective in realistic circumstances. A good discrimination at the beginning of the processing chain can reduce the remaining bad features to a few outliers, rather than leaving them an overwhelming majority. Outlier detection techniques at higher levels in the processing chain are then much more likely to succeed.

Bibliography

[Ana89] 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.

[BYX82] P. J. Burt, C. Yen, and X. Xu. Local correlation measures for motion analysis: a comparative study. In *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing*, pages 269–274, 1982.

[CL74] D. J. Connor and J. O. Limb. Properties of frame-difference signals generated by moving images. *IEEE Transactions on Communications*, COM-22(10):1564–1575, October 1974.

[CR76] C. Cafforio and F. Rocca. Methods for measuring small displacements in television images. *IEEE Transactions on Information Theory*, IT-22:573–579, 1976.

[DN81] L. Dreschler and H.-H. Nagel. Volumetric model and 3d trajectory of a moving car derived from monocular tv frame sequences of a street scene. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 692–697, Vancouver, Canada, August 1981.

[F̈87] W. Foerstner. Reliability analysis of parameter estimation in linear models with applications to mansuration problems in computer vision. *Computer Vision, Graphics, and Image Processing*, 40:273–310, 1987.

[FM91] C. S. Fuh and P. Maragos. Motion displacement estimation using and affine model for matching. *Optical Engineering*, 30(7):881–887, July 1991.

[FP86] W. Foerstner and A. Pertl. *Photogrammetric Standard Methods and Digital Image Matching Techniques for High Precision Surface Measurements.* Elsevier Science Publishers, 1986.

[KR80] L. Kitchen and A. Rosenfeld. Gray-level corner detection. Computer Science Center 887, University of Maryland, College Park, April 1980.

[LK81] 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*, 1981.

[MO93] R. Manmatha and John Oliensis. Extracting affine deformations from image patches - i: Finding scale and rotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 754–755, 1993.

[Mor80] H. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover.* PhD thesis, Stanford University, September 1980.

[MPU79] D. Marr, T. Poggio, and S. Ullman. Bandpass channels, zero-crossings, and early visual information processing. *Journal of the Optical Society of America*, 69:914–916, 1979.

[OK92] M. Okutomi and T. Kanade. A locally adaptive window for signal matching. *International Journal of Computer Vision*, 7(2):143–162, January 1992.

[RGH80] T. W. Ryan, R. T. Gray, and B. R. Hunt. Prediction of correlation errors in stereo-pair images. *Optical Engineering*, 19(3):312–322, 1980.

[TH86] Q. Tian and M. N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 35:220–233, 1986.

[Van92] C. Van Loan. *Computational Frameworks for the Fast Fourier Transform.* Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, 1992.

[Woo83] G. A. Wood. Realities of automatic correlation problem. *Photogrammetric Engineering and Remote Sensing*, 49:537–538, April 1983.

Appendix A

Derivation of the Tracking Equation

In this appendix, we derive the basic tracking equation (3.8) by rewriting equations (3.6) and (3.7) in a matrix form that clearly separates unknowns from known parameters. The necessary algebraic manipulation can be done cleanly by using the Kronecker product defined as follows [Van92]. If A is a $p \times q$ matrix and B is $m \times n$, then the *Kronecker product* $A \otimes B$ is the $p \times q$ block matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1q}B \\ \vdots & & \vdots \\ a_{p1}B & \cdots & a_{pq}B \end{bmatrix}$$

of size $pm \times qn$. For any $p \times q$ matrix A, let now $v(A)$ be a vector collecting the entries of A in column-major order:

$$v(A) = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{pq} \end{bmatrix}$$

so in particular for a vector or a scalar $\mathbf{x}$we have $v(\mathbf{x}) = \mathbf{x}$). Given three matrices A,X,B for which the product AXB is defined, the following two equalities are then easily verified:

$$A^T \otimes B^T = (A \otimes B)^T \tag{A.1}$$

$$v(AXB) = (B^T \otimes A)v(X)\ . \tag{A.2}$$

The last equality allows us to "pull out" the unknowns D and $\mathbf{d}$ from equations (3.6) and (3.7).

The scalar term $\mathbf{g}^T\mathbf{u}$ appears in both equations (3.6) and (3.7). By using the identities (A.1), (A.2), and definition 3.4, we can write

$$\begin{aligned} \mathbf{g}^T\mathbf{u} &= \mathbf{g}^T(D\mathbf{x} + \mathbf{d}) \\ &= \mathbf{g}^T D\mathbf{x} + \mathbf{g}^T\mathbf{d} \\ &= v(\mathbf{g}^T D\mathbf{x}) + \mathbf{g}^T\mathbf{d} \\ &= (\mathbf{x}^T \otimes \mathbf{g}^T)v(D) + \mathbf{g}^T\mathbf{d} \\ &= (\mathbf{x} \otimes \mathbf{g})^T v(D) + \mathbf{g}^T\mathbf{d}\ . \end{aligned}$$

Similarly, the vectorized version of the 2×2 matrix $\mathbf{g}\mathbf{x}^T$ that appears in equation (3.6) is

$$\begin{aligned} v(\mathbf{g}\mathbf{x}^T) &= v(\mathbf{g}\,1\,\mathbf{x}^T) \\ &= \mathbf{x} \otimes \mathbf{g}\ . \end{aligned}$$

We can now vectorize matrix equation (3.6) into the following vector equation:

$$\left(\int\int_W U(\mathbf{x})\, w\, d\mathbf{x}\right) v(D) + \left(\int\int_W V(\mathbf{x})\, w\, d\mathbf{x}\right) \mathbf{d} = \int\int_W \mathbf{b}(\mathbf{x})\, w\, d\mathbf{x} \tag{A.3}$$

where the rank 1 matrices U and V and the vector $\mathbf{b}$ are defined as follows:

$$\begin{aligned} U(\mathbf{x}) &= (\mathbf{x} \otimes \mathbf{g})(\mathbf{x} \otimes \mathbf{g})^T \\ V(\mathbf{x}) &= (\mathbf{x} \otimes \mathbf{g})\mathbf{g}^T \\ \mathbf{b}(\mathbf{x}) &= [I(\mathbf{x}) - J(\mathbf{x})]\, v(\mathbf{g}\mathbf{x}^T) \,. \end{aligned}$$

Similarly, equation (3.7) can be written as

$$\left(\int\int_W V^T(\mathbf{x})\, w\, d\mathbf{x}\right) v(D) + \left(\int\int_W Z(\mathbf{x})\, w\, d\mathbf{x}\right) \mathbf{d} = \int\int_W \mathbf{c}(\mathbf{x})\, w\, d\mathbf{x} \tag{A.4}$$

where V has been defined above and the rank 1 matrix Z and the vector $\mathbf{c}$ are defined as follows:

$$\begin{aligned} Z(\mathbf{x}) &= \mathbf{g}\mathbf{g}^T \\ \mathbf{c}(\mathbf{x}) &= [I(\mathbf{x}) - J(\mathbf{x})]\, \mathbf{g} \,. \end{aligned}$$

Equations (A.3) and (A.4) can be summarized by introducing the symmetric 6×6 matrix

$$\begin{aligned} T &= \int\int_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} w\, d\mathbf{x} \\ &= \int\int_W \begin{bmatrix} x^2 g_x^2 & x^2 g_x g_y & xy g_x^2 & xy g_x g_y & x g_x^2 & x g_x g_y \\ x^2 g_x g_y & x^2 g_y^2 & xy g_x g_y & xy g_y^2 & x g_x g_y & x g_y^2 \\ xy g_x^2 & xy g_x g_y & y^2 g_x^2 & y^2 g_x g_y & y g_x^2 & y g_x g_y \\ xy g_x g_y & xy g_y^2 & y^2 g_x g_y & y^2 g_y^2 & y g_x g_y & y g_y^2 \\ x g_x^2 & x g_x g_y & y g_x^2 & y g_x g_y & g_x^2 & g_x g_y \\ x g_x g_y & x g_y^2 & y g_x g_y & y g_y^2 & g_x g_y & g_y^2 \end{bmatrix} w\, d\mathbf{x} \,, \end{aligned}$$

the 6×1 vector of unknowns

$$\mathbf{z} = \begin{bmatrix} v(D) \\ \mathbf{d} \end{bmatrix} ,$$

and the 6×1 vector

$$\begin{aligned} \mathbf{a} &= \int\int_W \begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix} w\, d\mathbf{x} \\ &= \int\int_W [I(\mathbf{x}) - J(\mathbf{x})] \begin{bmatrix} x g_x \\ x g_y \\ y g_x \\ y g_y \\ g_x \\ g_y \end{bmatrix} w\, d\mathbf{x} \,. \end{aligned}$$

With this notation, the basic step of the iterative procedure for the computation of affine deformation D and translation $\mathbf{d}$ is the solution of the following linear 6×6 system:

$$T\mathbf{z} = \mathbf{a} \,.$$