Shape and Motion from Image Streams: a Factorization Method—Part 3

Detection and Tracking of Point Features

Carlo Tomasi Takeo Kanade

April 1991

CMU-CS-91-132

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597.

Keywords: computer vision, motion, shape, time-varying imagery

Abstract

The factorization method described in this series of reports requires an algorithm to track the motion of features in an image stream. Given the small inter-frame displacement made possible by the factorization approach, the best tracking method turns out to be the one proposed by Lucas and Kanade in 1981.

The method defines the measure of match between fixed-size feature windows in the past and current frame as the sum of squared intensity differences over the windows. The displacement is then defined as the one that minimizes this sum. For small motions, a linearization of the image intensities leads to a Newton-Raphson style minimization.

In this report, after rederiving the method in a physically intuitive way, we answer the crucial question of how to choose the feature windows that are best suited for tracking. Our selection criterion is based directly on the definition of the tracking algorithm, and expresses how well a feature can be tracked. As a result, the criterion is optimal by construction.

We show by experiment that the performance of both the selection and the tracking algorithm are adequate for our factorization method, and we address the issue of how to detect occlusions. In the conclusion, we point out specific open questions for future research.

Preface

In principle, the stream of images produced by a moving camera allows the recovery of both the shape of the objects in the field of view, and the motion of the camera. Traditional algorithms recover depth by triangulation, and compute shape by taking differences between depth values. This process, however, is sensitive to noise for distant scenes.

To overcome this problem, we have developed a factorization method to decompose an image stream directly into object shape and camera motion, without computing depth as an intermediate step. To explore this new method, we designed a series of eleven technical reports, as shown in figure 1, going from basic theory to implementation.

The first two report, already published as CMU-CS-90-166 and CMU-CS-91-105, explore the idea in the case of planar motion, in which images are single scanlines, and then for unrestricted, 3D motion and complete images.

The method used to select and track points from frame to frame is described in detail in the present report, number 3.

As to the future reports, number 4 and 5 address shape and motion reconstruction and tracking for line features, rather than points. Reports number 6 and 7 deal with perspective effects and degenerate motion. Occlusion can be handled by our method, as shown in report number 8, while number 9 examines how to detect cases of spurious, that is, non-rigid, motion. The factorization algorithm in report number 2 deals with the whole stream of images at once; report number 10, in contrast, proposes an implementation that can work with an indefinitely long stream of images.

Report number 11 considers a more radical departure from the assumption of a static scene than spurious motion, that of several bodies moving independently in the field of view.

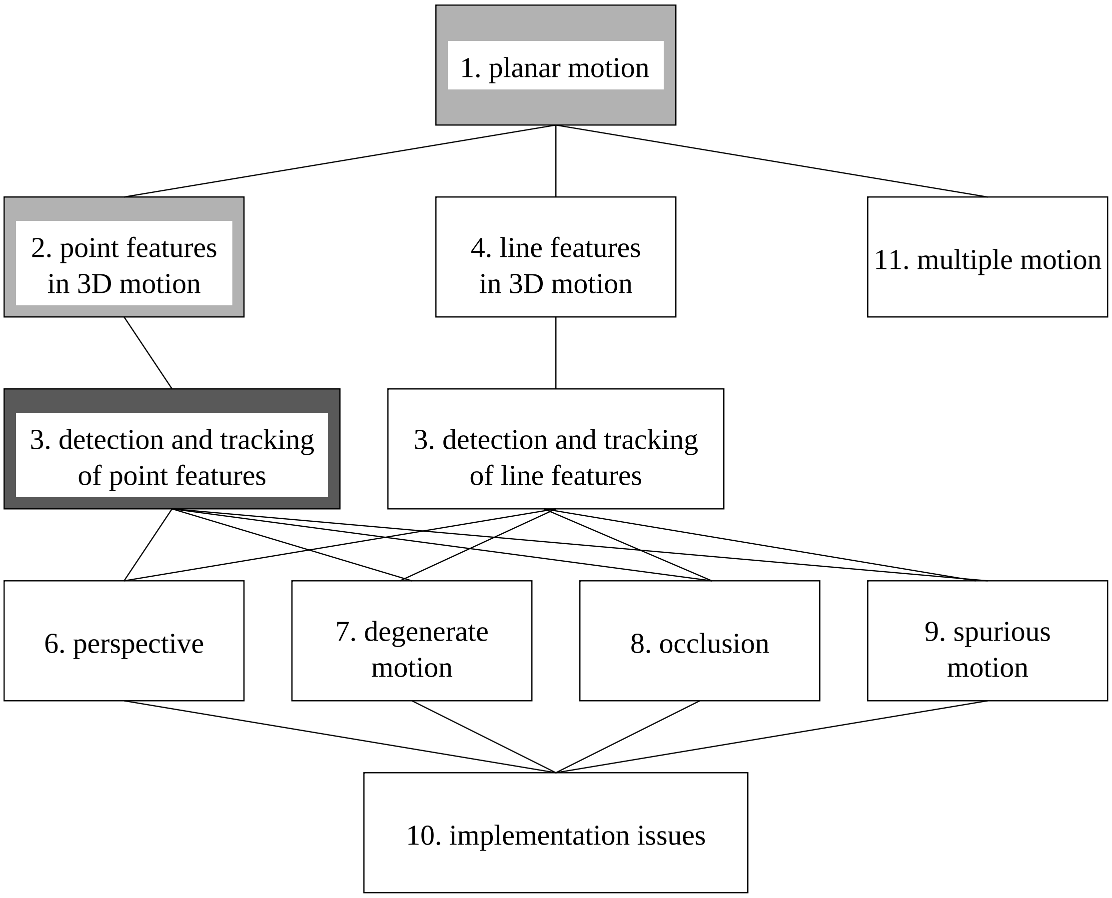

Figure 0.1: The reports in the series.

Chapter 1

Introduction

The factorization method introduced in reports 1 and 2 of this series [Tomasi and Kanade, 1990] [Tomasi and Kanade, 1991] requires selecting and tracking of features in an image stream. In this report we address the issues involved, and present our algorithm.

In general, two basic questions must be answered: how to select the features, and how to track them from frame to frame. We base our solution to the tracking problem on a previous result by Lucas and Kanade [Lucas and Kanade, 1981], who proposed a method for registering two images for stereo matching.

Their approach is to minimize the sum of squared intensity differences between a past and a current window. Because of the small inter-frame motion, the current window can be approximated by a translation of the old one. Furthermore, for the same reason, the image intensities in the translated window can be written as those in the original window plus a residue term that depends almost linearly on the translation vector. As a result of these approximations, one can write a linear 2×2 system whose unknown is the displacement vector between the two windows.

In practice, these approximations introduce errors, but a few iterations of the basic solution step suffice to converge. The result is a simple, fast, and accurate registration method.

The first question posed above, however, was left unanswered in [Lucas and Kanade, 1981]: how to select the windows that are suitable for accurate tracking. In the literature, several definitions of a "good feature" have been proposed, based on an *a priori* notion of what constitutes an "interesting" window. For example, Moravec and Thorpe propose to use windows with high standard deviations in the spatial intensity profile [Moravec, 1980], [Thorpe, 1984], Marr, Poggio, and Ullman prefer zero crossings of the Laplacian of the image intensity [Marr *et al.*, 1979], and Kitchen, Rosenfeld, Dreschler, and Nagel define corner features based on first and second derivatives of the image intensity function [Kitchen and Rosenfeld, 1980], [Dreschler and Nagel, 1981].

In contrast with these selection criteria, which are defined independently of the registration algorithm, we show in this report that a criterion can be derived that explicitly optimizes the tracking performance. In other words, we define a feature to be good if it can be tracked well.

In this report, we first pose the problem (chapter 2), and rederive the equations of Lucas and Kanade in a physically intuitive way (chapter 3). Chapter 4 introduces the selection criterion. We then show by experiment (chapter 5) that the performance of both selector and tracker is satisfactory in a wide variety of situations, and discuss the problem of detecting feature occlusion. Finally, in chapter 6, we close with a discussion of the suitability of this approach to our factorization method for the computation of shape and motion, and point out directions for further research.

Chapter 2

Feature Tracking

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 and y as well as 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:

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

in plain English, 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 $\mathbf{d} = (\xi, \eta)$ is called the *displacement* of the point at $\mathbf{x} = (x, y)$ between time instants t and $t + \tau$, and is in general a function of x, y, t, and τ.

Even in a static environment under a constant lighting, the property described by equation (2.1) is violated in many situations. For instance, at occluding boundaries, points do not just move within the image, but appear and disappear. Furthermore, the photometric appearance of a region on a visible surface changes when reflectivity is a function of the viewpoint.

However, the invariant (2.1) is by and large satisfied at surface markings, and away from occluding contours. At locations where the image intensity changes abruptly with x and y, the point of change remains well defined even in spite of small variations of overall brightness around it.

Surface markings abound in natural scenes, and are not infrequent in man-made environments. In our experiments, we found that markings are often sufficient to obtain both good motion estimates and relatively dense shape results. As a consequence, this report is essentially concerned with surface markings.

The Approach

An important problem in finding the displacement $\mathbf{d}$ of a point from one frame to the next is that a single pixel cannot be tracked, unless it has a very distinctive brightness with respect to all of its neighbors. In fact, the value of the pixel can both change due to noise, and be confused with adjacent pixels. As a consequence, it is often hard or impossible to determine where the pixel went in the subsequent frame, based only on local information.

Because of these problems, we do not track single pixels, but *windows* of pixels, and we look for windows that contain sufficient texture. In chapter 4, we give a definition of what sufficient texture is for reliable feature tracking.

Unfortunately, different points within a window may behave differently. The corresponding three-dimensional surface may be very slanted, and the intensity pattern in it can become warped from one frame to the next. Or the window may be along an occluding boundary, so that points move at different velocities, and may even disappear or appear anew.

This is a problem in two ways. First, how do we know that we are following the same window, if its contents change over time? Second, if we measure "the" displacement of the window, how are the different velocities combined to give the one resulting vector? Our solution to the first problem is residue monitoring: we keep checking that the appearance of a window has not changed too much. If it has, we discard the window.

The second problem could in principle be solved as follows: rather than describing window changes as simple translations, we can model the changes as a more complex transformation, such as an affine map. In this way, different velocities can be associated to different points of the window.

This approach was proposed already in [Lucas and Kanade, 1981], and was recently explored in a more general setting in [Rehg and Witkin, 1991].

We feel, however, that in cases where the world is known to be rigid the danger of over-parametrizing the system outweighs the advantages of a richer model. More parameters to estimate require the use of larger windows to constrain the parameters sufficiently. On the other hand, using small windows implies that only few parameters can be estimated reliably, but also alleviates the problems mentioned above.

We therefore choose to estimate only two parameters (the displacement vector) for small windows. Any discrepancy between successive windows that cannot be explained by a translation is considered to be error, and the displacement vector is chosen so as to minimize this residue error.

Formally, if we redefine $J(\mathbf{x}) = I(x, y, t+\tau)$, and $I(\mathbf{x}-\mathbf{d}) = I(x-\xi, y-\eta, t)$, where the time variable has been dropped for brevity, our local image model is

$$J(\mathbf{x}) = I(\mathbf{x}-\mathbf{d}) + n(\mathbf{x}) \ , \tag{2.2}$$

where n is noise.

The displacement vector $\mathbf{d}$ is then chosen so as to minimize the residue error defined by the following double integral over the given window $\mathcal{W}$:

$$\epsilon = \int_{\mathcal{W}} [I(\mathbf{x}-\mathbf{d}) - J(\mathbf{x})]^2 w \, d\mathbf{x} \ . \tag{2.3}$$

In this expression, w is a weighting function. In the simplest case, w could be set to 1. Alternatively, w could be a Gaussian-like function, to emphasize the central area of the window. The weighting function w could also depend on the image intensity pattern: the relation (3.3) holds for planar patches, and w could be chosen, as suggested in [Lucas and Kanade, 1981], to de-emphasize regions of high curvature.

Several ways have been proposed in the literature to minimize this residue (see [Anandan, 1989] for a survey). When the displacement $\mathbf{d}$ is much smaller than the window size, the linearization method presented in [Lucas and Kanade, 1981] is the most efficient way to proceed.

In the next chapter, we rederive this method, and explain it in a physically intuitive way. Then, in chapter 4, we show that the registration idea can be extended also to *selecting* good features to track. As a consequence, feature selection is no longer based on an arbitrary criterion for deciding what constitutes a feature. Rather, a good feature is defined as one that can be tracked well, in a precise mathematical sense.

Chapter 3

Solving for the Image Displacement

In the previous chapter, we justified our local model of image changes as a simple translation, plus some noise (equation (2.2)), and we posed the registration problem as the minimization of the error residue defined by equation (2.3). In this chapter, we show that if the inter-frame displacement is sufficiently small with respect to the texture fluctuations within the window, the displacement vector itself can be written approximately as the solution to a 2×2 linear system of equations.

When the displacement vector is small, the intensity function can be approximated by its Taylor series truncated to the linear term:

$$I(\mathbf{x} - \mathbf{d}) = I(\mathbf{x}) - \mathbf{g} \cdot \mathbf{d} \ ,$$

and we can write the residue defined in equation (2.3) as

$$\epsilon = \int_{\mathcal{W}} [I(\mathbf{x}) - \mathbf{g} \cdot \mathbf{d} - J(\mathbf{x})]^2 w \, d\mathbf{x} = \int_{\mathcal{W}} (h - \mathbf{g} \cdot \mathbf{d})^2 w \, d\mathbf{x} \ , \tag{3.1}$$

where $h = I(\mathbf{x}) - J(\mathbf{x})$.

This residue is a quadratic function of the displacement $\mathbf{d}$. As a consequence, the minimization can be done in closed form. Differentiating the last expression of the residue ϵ in equation 3.1 with respect to $\mathbf{d}$ and setting the result equal to zero yields the following vector equation:

$$\int_{\mathcal{W}} (h - \mathbf{g} \cdot \mathbf{d}) \mathbf{g} \, w \, dA = 0 \ .$$

Since $(\mathbf{g}\cdot\mathbf{d})\mathbf{g} = (\mathbf{g}\mathbf{g}^T)\mathbf{d}$, and $\mathbf{d}$ is assumed to be constant within $\mathcal{W}$, we have

$$\left(\int_{\mathcal{W}} \mathbf{g}\mathbf{g}^T\, w\, dA\right)\mathbf{d} = \int_{\mathcal{W}} h\mathbf{g} w\, dA \; .$$

This is a system of two scalar equations in two unknowns. It can be rewritten as

$$G\mathbf{d} = \mathbf{e} \; , \tag{3.2}$$

where the coefficient matrix is the symmetric, 2×2 matrix

$$G = \int_{\mathcal{W}} \mathbf{g}\mathbf{g}^T\, w\, dA \; ,$$

and the right-hand side is the two-dimensional vector

$$\mathbf{e} = \int_{\mathcal{W}} (I - J)\mathbf{g} w\, dA \; .$$

In the last expression, we wrote h explicitly as the difference between the two frames I and J.

Equation (3.2) is the basic step of the tracking procedure. For every pair of adjacent frames, the matrix G can be computed from one frame, by estimating gradients and computing their second order moments. The vector $\mathbf{e}$, on the other hand, can be computed from the difference between the two frames, along with the gradient computed above. The displacement $\mathbf{d}$ is then the solution of system (3.2).

Physical Interpretation

To understand the meaning of this solution, we rederive the expression (3.1) of the residue ϵ in a physically more intuitive way.

Consider the intensity function within the window $\mathcal{W}$. Figure 3.1 shows an example. Make a second copy of it, and superimpose it on the first. There is no space between the two intensity surfaces. If you now move the copy by a small horizontal displacement, a gap forms between the two surfaces.

The width of the gap, measured horizontally, is a function of the displacement between the two intensity patches. When measured vertically, on the other hand, the width of the gap is just the difference between the values of the two intensity profiles.

Figure 3.1: Example of image intensity function within a small window.

In the following, we show that for small displacements the horizontal and the vertical width of the gap at a given point in the image are related to each other through the image gradient at that point. As a consequence, we can write the infinitesimal volume of the gap in a neighborhood of a given point in two different ways. One is a function of the displacement, the other is not.

We then look for the displacement that makes the difference between the two expressions as small as possible in the Least Squared Error sense and over the entire window $\mathcal{W}$. This yields, by a different route, the last expression given in equation (3.1) for the residue ϵ.

Figure 3.2 shows a small patch of the intensity function $I(\mathbf{x})$ and the corresponding translated patch behind it. The same figure shows also a cross section of the two patches along the direction of the image gradient.

The displacement vector $\mathbf{d}$ is in general in a different direction than the image gradient $\mathbf{g} = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$. If the gradient is expressed as

$$\mathbf{g} = g\mathbf{u} \ ,$$

where g is the magnitude of $\mathbf{g}$ and $\mathbf{u}$ is a unit vector, then the displacement Δ measured along the gradient direction is the projection of $\mathbf{d}$ along $\mathbf{u}$:

$$\Delta = \mathbf{d} \cdot \mathbf{u} \ .$$

From the right part of figure 3.2, we see that the vertical gap width $h = I - J$ is

$$h = \Delta \tan \alpha \ ,$$

where α is the maximum slope of the patch. Since the tangent of α is equal to the magnitude g of the gradient, we can write

$$h = \Delta\, g = \mathbf{d} \cdot \mathbf{u}\, g = \mathbf{d} \cdot \mathbf{g} \ .$$

By equating the first and last term, we obtain the following equation relating the image gradient $\mathbf{g}$, the inter-frame displacement $\mathbf{d}$, and the difference h between image intensities:

$$\mathbf{g} \cdot \mathbf{d} = h \ . \tag{3.3}$$

This is a scalar equation in the two-dimensional unknown $\mathbf{d}$. The image gradient $\mathbf{g}$ can be estimated from one image, while the difference h is easily computed from both.

Figure 3.2: Two corresponding image intensity patches (left), and a section of the same through the direction of the gradient (right). Δ is the projection of the displacement $\mathbf{d}$ along the gradient $\mathbf{g}$.

The fact that the number of unknowns in equation (3.3) exceeds the number of constraints is called the *aperture problem* in the literature [Hildreth, 1983], [Horn and Schunck, 1981]: if we just look at that patch (as if through a small aperture), we cannot determine the displacement $\mathbf{d}$, but at most one component of it.

If we now consider the whole window $\mathcal{W}$, however, different patches may allow us to compute different components of the displacement vector, assumed to be constant within $\mathcal{W}$. To combine these measurements, we observe that if the displacement $\mathbf{d}$ is assigned a wrong value, there will be a difference between the left and the right-hand side of equation (3.3). The best value for $\mathbf{d}$ can be chosen as the one that minimizes the square of that difference, integrated over the entire window. In other words, we minimize the weighted residue

$$\int_{\mathcal{W}} (h - \mathbf{g} \cdot \mathbf{d})^2 w \, dA$$

with respect to $\mathbf{d}$. By comparing with equation (3.1), we see that the expression above is equal to the residue ϵ, as desired.

Equation (3.3) holds exactly either when the displacement $\mathbf{d}$ approaches zero, or when the image intensities are linear functions of the image coordinates x and y. In fact, this equation assumes that the patches in figure 3.2 be planar. For finite displacements, the approximation will cause some error on $\mathbf{d}$ especially at high-curvature points in the image intensity function.

As a consequence, the solution $\mathbf{d}$ to equation (3.2) will usually contain some error. However, the images can be approximately registered by using this solution, and the basic step (3.2) can be repeated. At every iteration step, images are resampled by bilinear interpolation to achieve subpixel accuracy. The closer we are to the solution, the better the approximations underlying equation (3.2). In practice, we found that very few iterations usually suffice for convergence. We discuss some experiments in chapter 5.

Not all parts of an image lend themselves equally well to this tracking method. For instance, when the intensity pattern I is constant, the matrix G is null, and the displacement $\mathbf{d}$ is undefined. In the next chapter, we address the problem of how to select good windows to track.

Chapter 4

Feature Selection

Regardless of the method used for tracking, not all parts of an image contain motion information. Similarly, along a straight edge, we can only determine the motion component orthogonal to the edge.

In general terms, the strategy for overcoming these difficulties is to use only regions with a rich enough texture. In this spirit, researchers have proposed to track corners, or windows with a high spatial frequency content, or regions where some mix of second-order derivatives was sufficiently high.

All these definitions usually yield trackable features. However, these "interest operators" are often based on a preconceived and sometimes 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 come with no guarantee of being the best for the tracking algorithm to produce good results.

Instead, we propose a more principled approach. Rather than defining our notion of a good window *a priori*, we base our definition on the method we use for tracking. A good window is one that can be tracked well. With this approach, we know that a window is omitted only if it is not good enough for the purpose: the selection criterion is optimal by construction.

With the formulation of tracking introduced in the previous section, this concept is easy to formalize. In fact, we can track a window from frame to frame if the system (3.2) represents good measurements, and if it can be solved reliably.

This means that the 2×2 coefficient matrix G of the system must be both above the image noise level and well-conditioned. In turn, the noise requirement implies that both eigenvalues of G 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 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 G 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 G 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 gives us 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.

Chapter 5

Experiments

In this chapter, we evaluate the performance of both feature selection and tracking on real images. To this end, we use a stream of 100 frames, showing surfaces of several different types: a furry puppet, a cylindrical and glossy mug with strong surface markings, an artichoke, a flat model street sign. Figures 5.1 and 5.2 show the first and the last frame of the stream, respectively. Between frames, the camera was translated to the right, producing a displacement of about one pixel per frame.

5.1 Feature Selection

Figure 5.3 shows an intensity encoding of the value of the smaller of the two eigenvalues of the tracking matrix G (see equation 3.2) for all the square windows of size 15 in the first frame. We call this the *minor* eigenvalue.

Figure 5.4 shows a histogram of the eigenvalues displayed in figure 5.3. For feature detection, we choose a threshold somewhere in the large gap between the near-zero and the higher cluster. Because of the size of that gap, the threshold value is not critical. We select a value of 10.

The feature selection algorithm sorts the minor eigenvalues in decreasing order, and picks feature coordinates from the top of the sorted list. Every time a coordinate pair is selected, it is assigned a new feature number. To obtain non-overlapping features, all the features in the list that overlap the window centered at the selected pair are deleted. Figure 5.5 shows the feature windows computed from the frame in figure 5.1. If desired, the requirement

Figure 5.1: The first frame of the stream used in the experiments.

Figure 5.2: The last frame of the stream. The total image displacement from first to last frame is about 100 pixels (one pixel per frame).

of zero overlap can be relaxed by enforcing a minimum distance between window centers smaller than the window size.

From figure 5.5, we see that the eigenvalue criterion selects the corners on the mug as well as fuzzier features on the puppet, both along the edges of the spots, and elsewhere. Also, a considerable number of features is found on the artichoke, where the intensity patterns are very irregular. The background, as well as the relatively uniform areas on the pedestrian sign and on the mug, contain no features. It is doubtful that any useful motion information can be extracted from those areas.

No features are found along the straight edges on the mug. These edges are characterized by a nearly-zero minor eigenvalue, and are good examples of regions suffering from the so-called "aperture problem" discussed in chapter 3.

Figures 5.6 and 5.7 show four sample feature windows. Each pair of illustrations shows the grey values within a window, and its isometric plot. All feature windows have substantial variations of intensity, but can hardly be classified as "corners".

An interesting phenomenon can be noticed in figure 5.5 concerning features at the corners on the mug: these windows are almost invariably positioned so that the corner in each of them is at the very edge of the window, which is filled by the brighter side of the corner (see for instance feature number 1).

This phenomenon is due to the fact that the intensity variations in the bright regions, albeit very small, are larger than those in the dark regions. The odd placement of the corner windows, then, indeed maximizes the intensity variations within the windows.

Although this phenomenon presented no difficulty in our experiments, it is possible that with noisier images the "interesting" part of the feature is lost from one frame to the next because it is too close to the window boundary. We leave the exploration of this conjecture to future work.

5.2 Tracking

Figure 5.8 shows the last of the 100 frames in the sequence, with the superimposed features, as tracked by the algorithm. Each feature required typically fewer than five iterations of the basic tracking step (see equation 3.2) to

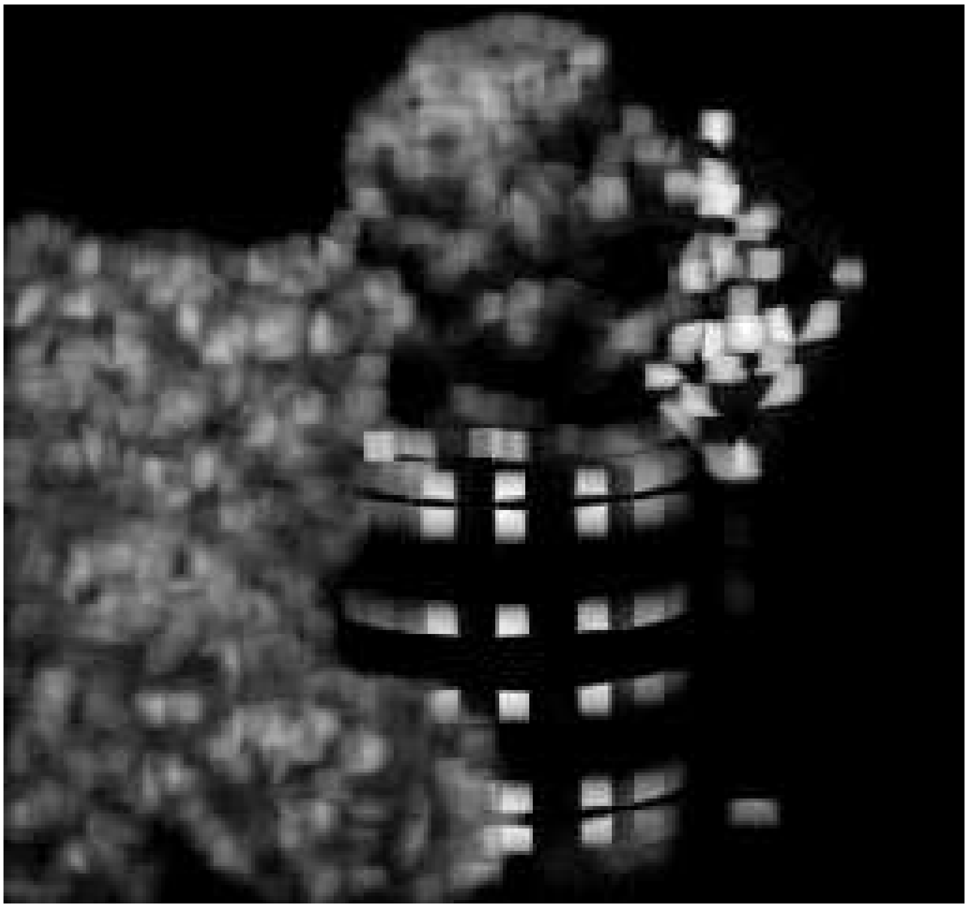

Figure 5.3: The minor eigenvalue of G (see equation (3.2)) for the first frame in the stream (figure (5.1)). Brighter areas correspond to higher values. The intensities in this display were compressed logarithmically. The square patterns reflect the shape of the 15×15 window used by the detector (and the tracker).

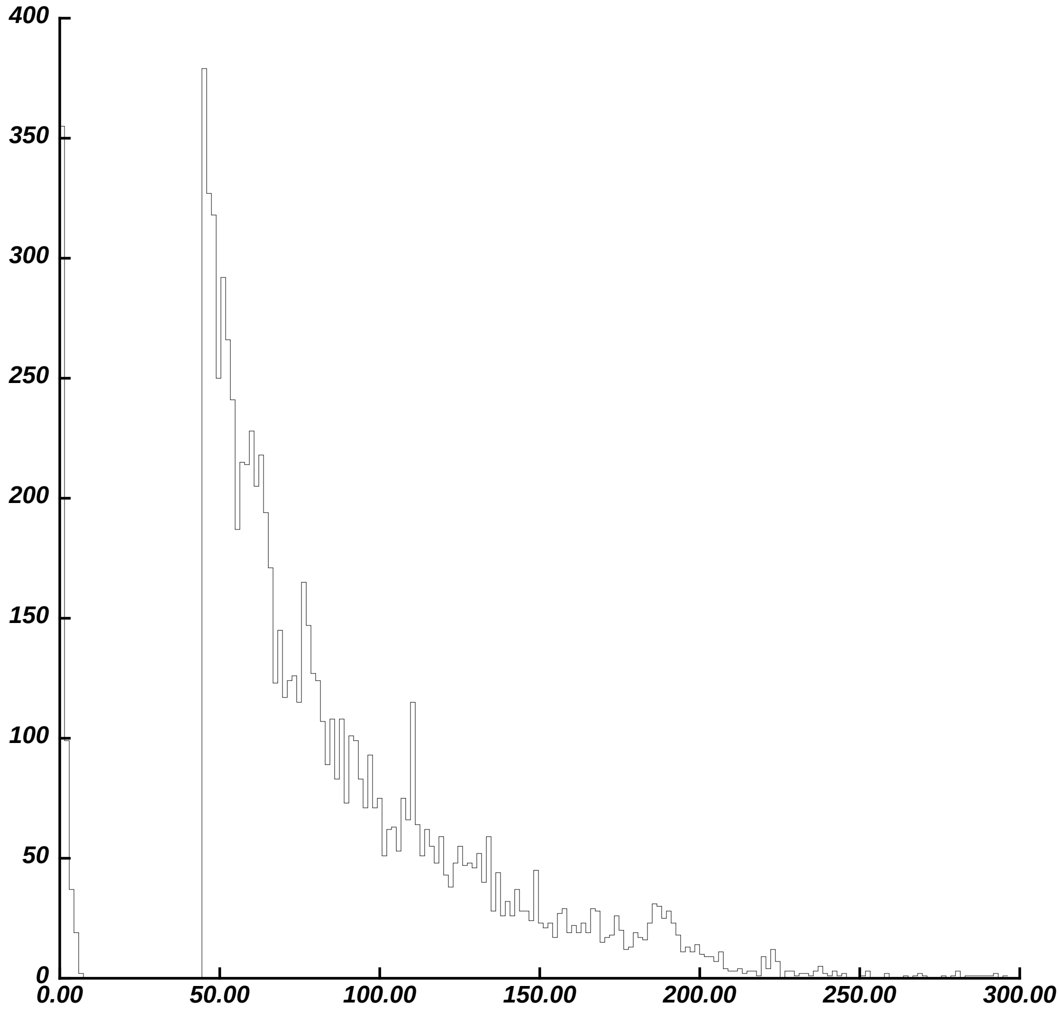

Figure 5.4: Histogram of the minor eigenvalues. Notice the wide gap between the near-zero values, corresponding to the background and to uniform regions in frame 1, and the upper cluster.

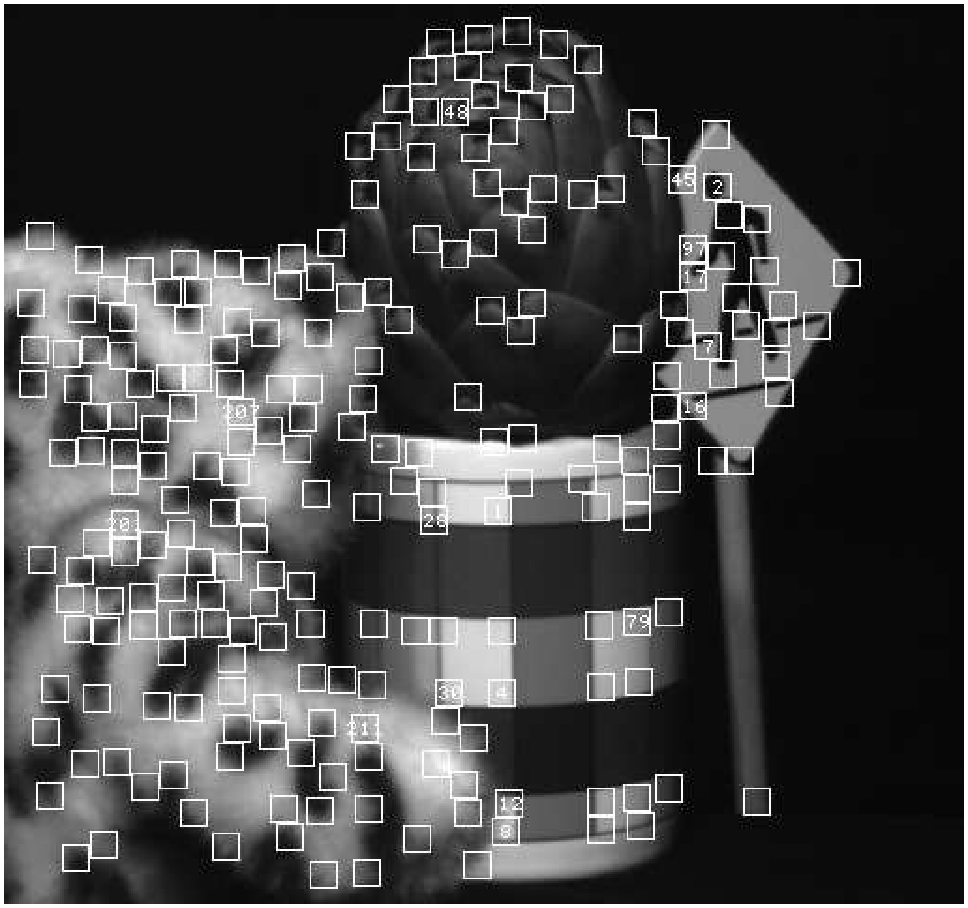

Figure 5.5: The non-overlapping features produced by thresholding the eigenvalues of figure 5.3 with a value of 10. Some features are numbered for reference in the text.

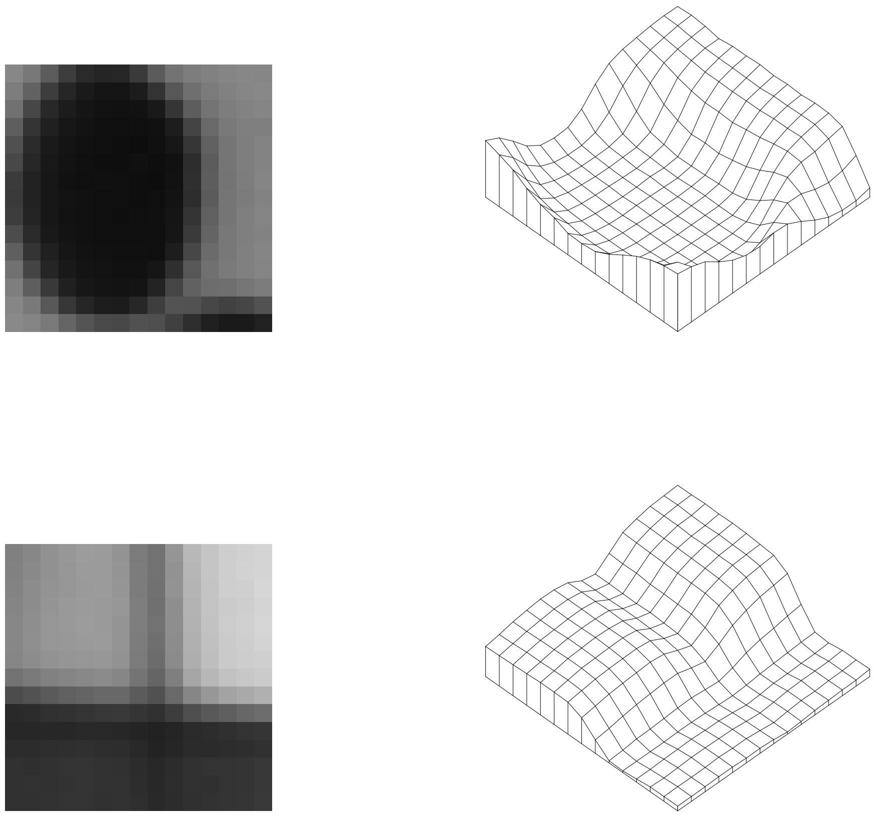

Figure 5.6: Two sample feature windows: the head of the man on the pedestrian sign (feature 2), and a corner on the mug (feature 28).

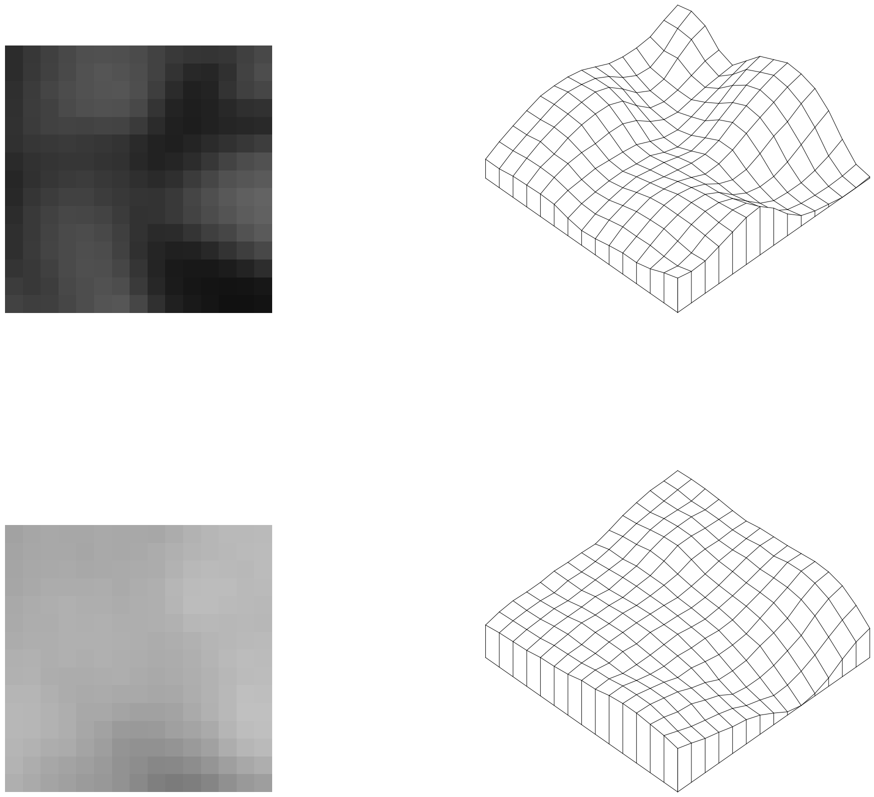

Figure 5.7: Two more sample feature windows: a detail of the artichoke (feature 48), and a spot on the puppet (feature 211).

stabilize the displacement estimate to within one hundredth of a pixel.

217 of the 226 features selected in the first frame survive tracking throughout the stream. No gross errors are made for any of the surviving features. Of the nine missing features at the end of the stream, six disappear off the right image boundary. Of the other three, two (201 and 207, on the fur of the puppet) are too weak to be tracked.

The ninth missing feature, number 79, on the right side of the mug in figure 5.5, is lost because in frame 40 the tracker did not converge within ten iterations. It would have taken 14 iterations for complete convergence, that is, to bring the change in displacement due to a new iteration below one hundredth of a pixel. The reason for the large number of iterations is that feature 79 is on top of a glossy surface viewed at a substantial slant angle. This causes the feature window to change its appearance substantially from frame to frame.

During tracking, a cumulative residue is computed for each feature window. This residue is defined as the root-mean-squared intensity difference between the first and the current window. The cumulative residue is plotted in figure 5.9 as a function of the frame number. Notice that most of the residue curves grow at the rate of about one intensity level per pixel every one-hundred frames. As discussed below, a larger residue may indicate occlusion.

Window Size and Occlusion

As discussed in chapter 3, smaller windows are more sensitive to noise. However, they are also less likely to straddle surface discontinuities, or to be affected by distortions due to changes of viewpoint. To illustrate this point, we compared tracking of feature number 2 with square windows 15 and 31 pixels wide. Feature number 2 is the head of the pedestrian on the sign (see figure 5.6). In figure 5.10, the tracks left by feature number 2 are shown for the two window sizes.

At the end of the tracking process, there is a difference between the results: the discrepancy is of about 3 pixels horizontally, and about 0.8 pixels vertically. One of the two final coordinate pairs must be wrong. The reason for the discrepancy can be seen from figure 5.11, which shows the first and last windows in the stream with the 31-pixel windows. Halfway through the stream, the edge of the artichoke appeared in the window, causing the error

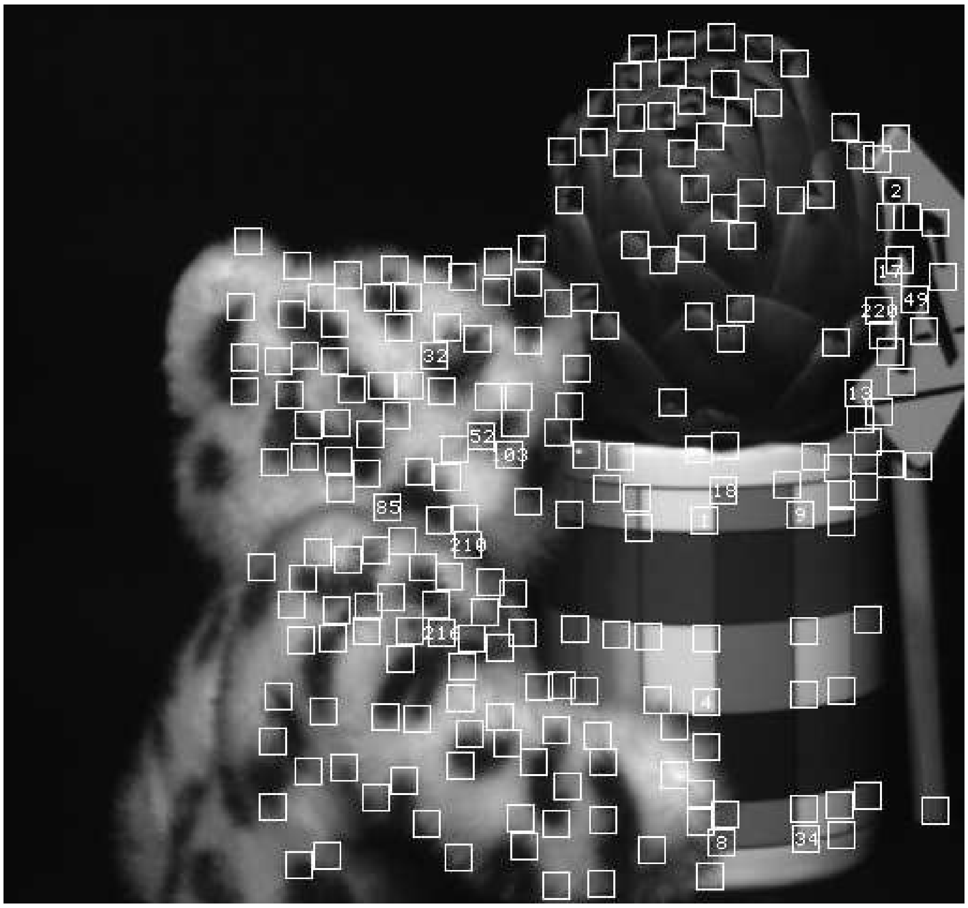

Figure 5.8: The features surviving through the 100 frames. Of the 226 starting features, only nine disappear, six of them off the right image boundary.

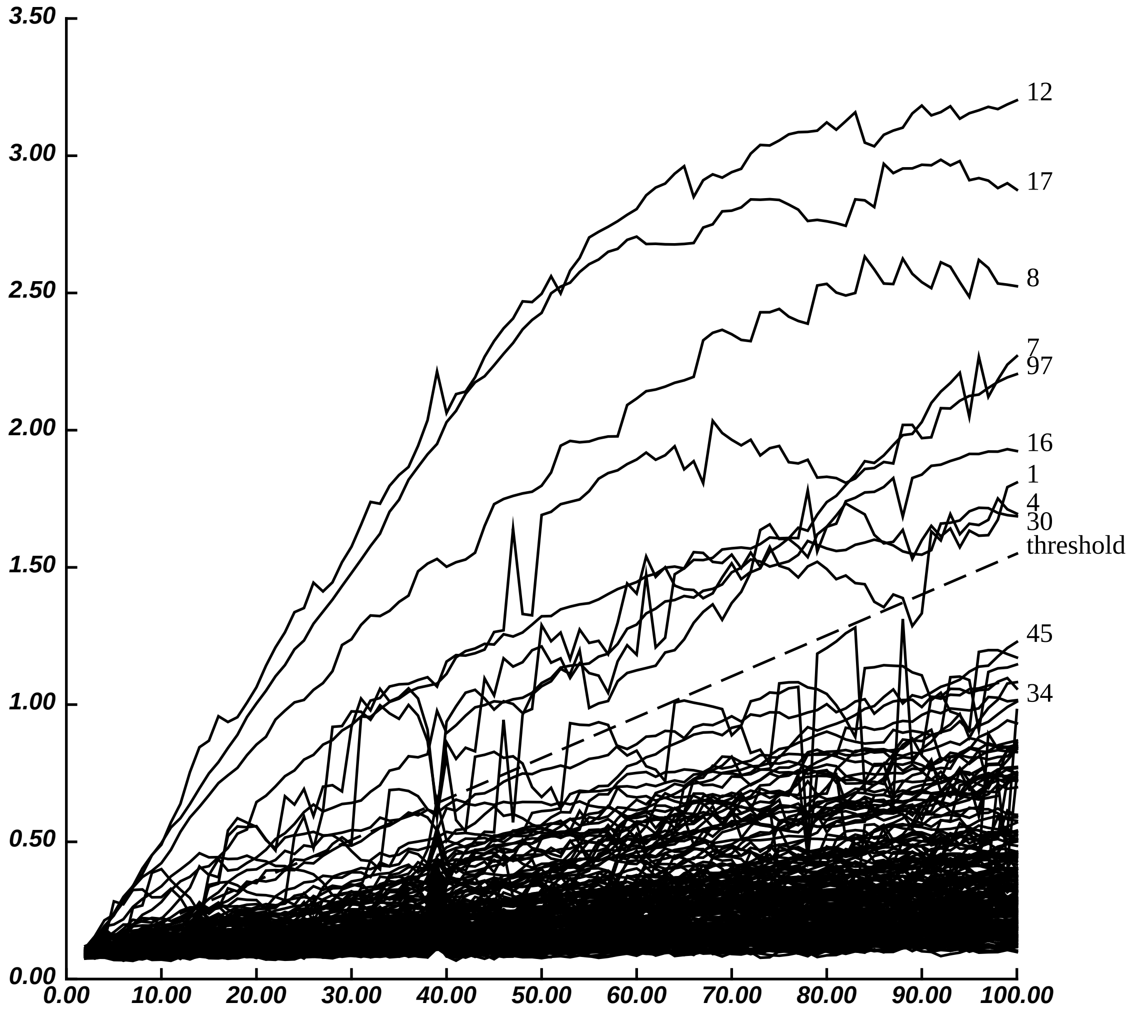

Figure 5.9: Cumulative root-mean-squared residues between the current and the first frame for all of the features selected by our method. Units are intensity levels per pixel, versus the frame number. Features 7, 8, 12, 16, 17, and 97 are occluded during the stream.

in the displacement values.

Small windows minimize these occlusion problems. On the other hand, they will occur no matter what the size of the window. The dashed line in figure 5.9 suggests a threshold on the cumulative residue for the detection of occlusions. Of the features above the threshold, those numbered 7, 8, 12, 16, 17, and 97 are occlusions. The other features, numbered 1, 4, 30, are in an area of the mug that receives strong reflections from the light source. As a result, the overall intensity pattern changes substantially from the first to the last frame, increasing the value of the residue even if the features are tracked well (compare figure 5.5 with the last frame, figure 5.8).

This simple occlusion detection method would identify most occlusions, at least for the sequence used for this experiment. It is possible that modeling window changes as affine transformations, rather than simple translations, increases the separation between good and bad residues, thus yielding a more reliable detection.

False Features

Other occlusion phenomena produce problems that are more difficult to detect. For instance, feature number 45 starts at the intersection of the right boundary of the artichoke with the upper left edge of the traffic sign (see figure 5.5). As the camera moves, the local appearance of that intersection does not change, but its position in space slides along both edges. The tracker cannot notice the problem, but the feature would create a bad measurement for any motion and shape method that assumes that features correspond to static points in the environment. However, this problem can be detected in three dimensions, after the motion and shape algorithm has been applied.

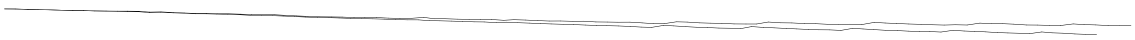

Figure 5.10: Results of tracking feature 2 (the man's head on the sign) with two different window sizes (square windows of 15 and 31 pixels). The distance between the right endpoints is about 3 pixels horizontally and 0.8 pixels vertically.

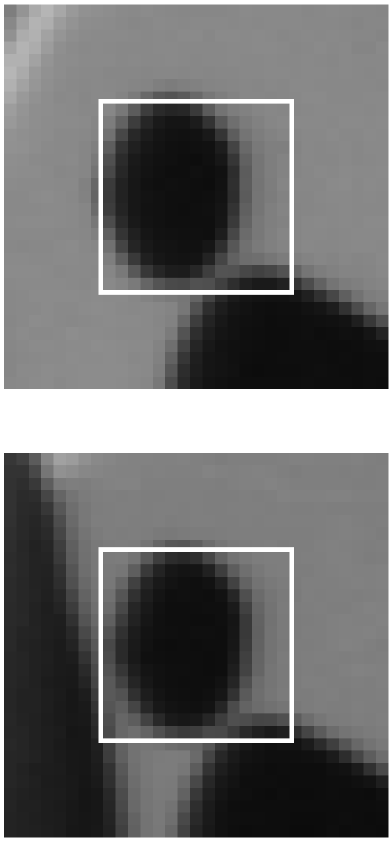

Figure 5.11: A large window is more likely to change dramatically during camera motion. Here, the boundary of the artichoke appears within the large 31×31 window of feature 2 (the head of the pedestrian on the sign) somewhere between frame 1 (top) and 100 (bottom), causing the error in figure 5.10. The smaller, 15×15, window (outlined in white) is less susceptible to the problem.

Chapter 6

Conclusion

The main focus of our research on visual motion is the reconstruction of three-dimensional shape and motion from the motion of features in an image sequence, as outlined in the first two reports of this series [Tomasi and Kanade, 1990], [Tomasi and Kanade, 1991]. Many algorithms proposed in the literature for similar purposes assume that feature points are available from some unspecified "previous processing".

Of course, the validity of these algorithms, including ours, depends critically on whether this preliminary processing can actually be done. In this technical report, we have shown that it is possible to go from an image stream to a collection of image features, tracked from frame to frame.

We chose to use the window-based technique proposed in [Lucas and Kanade, 1981], because it is simple, fast, and gives accurate results if windows are selected appropriately.

The need for a careful choice of the windows to track is crucial, and we proposed a direct and effective solution to this problem. The proposed criterion, based on the size of the smaller eigenvalue of the tracking matrix G, is well justified by the nature of the tracking method. Furthermore, it subsumes previous feature selection criteria, in that it detects corners equally well as regions with high spatial frequency content, or with high second-order derivatives, or high values of intensity variance.

The experiments outlined in chapter 5, as well as those described in other reports in this series, show that the overall performance is good.

Of course, this method does not settle the issue of motion detection in image sequences. Window tracking requires good surface markings, can give rise to false measurements from windows along occluding boundaries, and yields relatively sparse results.

However, false measurements should probably be detected at a higher level in the processing chain, when measurements are combined into three-dimensional motion and shape estimates.

Also, results from surface markings are very accurate, typically to within one tenth of a pixel or better, and are therefore well suited for motion and shape estimation.

As to sparsity, if rich shape results are the goal, the shape and motion method will have to be complemented with other techniques for a denser reconstruction of surfaces. However, the number of features obtained in typical scenarios [Tomasi and Kanade, 1991] is more than sufficient to obtain accurate motion results, and to initialize a dense shape map, to be used by other modules for a more detailed reconstruction of the visible surfaces.

Future Work

As an agenda for future work on the detection and tracking of features in a stream of images, we now summarize the issues we left open in this report.

Two parameters need to be specified for detection and tracking: the size of the window and the detection threshold. We have argued that windows should be as small as possible, compatibly with good noise rejection. However, it has been shown [Okutomi and Kanade, 1990] that a careful choice of the window size can improve performance considerably. It would be interesting to develop an inexpensive and automatic window size selection algorithm.

The feature detection threshold was chosen in this report based on a histogram of the minor eigenvalues for the entire image (figure 5.4). Also this parameter should, in the future, be selected automatically.

In chapter 2, we argued that for small windows a pure translation model of image changes gives more reliable results than a model with more parameters. On the other hand, in chapter 5, we also conjectured that an affine transformation model would improve the discrimination power of the occlusion detector. This suggests a combined strategy: the translation model is more adequate for the registration of adjacent frames, while a more sophisticated transformation model is probably required when comparing distant frames (in the first and the current image), as done for the detection of occlusions. How far the occlusion detector can be improved by the affine transformation model is an interesting open question.

In the discussion of figure 5.5, we pointed out the asymmetric behavior of the feature selector for corner-like regions: the corner almost always appears along the boundary of the window. It would be interesting to explore this phenomenon. First of all, it should be determined whether this is at all a problem, that is, if it can cause features to be lost when images are very noisy. If so, it should be possible to add a feature stability criterion to the maximization of $\min(\lambda_1, \lambda_2)$: choose features so as to maximize the minor eigenvalue *and* the spatial stability of the feature in the presence of image noise. The exact formalization of this stability criterion, as well as the recipe for combining it with the eigenvalue maximization rule, are open research questions.

Finally, a complete investigation of the detection and tracking methods presented in this report requires a more thorough performance evaluation. First, the methods should be compared experimentally with those previously proposed in the literature. Second, performance should be measured for a large number of sequences in a more diverse set of situations.

Bibliography

[Anandan, 1989]
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.

[Dreschler and Nagel, 1981]
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.

[Hildreth, 1983]
E. C. Hildreth. *The Measurement of Visual Motion.* PhD thesis, MIT, August 1983.

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

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

[Lucas and Kanade, 1981]
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.

[Marr *et al.*, 1979]
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.

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

[Okutomi and Kanade, 1990]
M. Okutomi and T. Kanade. A locally adaptive window for signal matching. In *Proceedings of the Third International Conference on Computer Vision*, pages 190–199, Osaka, Japan, December 1990.

[Rehg and Witkin, 1991]
J. Rehg and A. Witkin. Visual tracking with deformation models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 844–850, Sacramento, CA, April 1991.

[Thorpe, 1984]
C. E. Thorpe. *Vision and navigation for a robot rover.* PhD thesis, Carnegie Mellon University, December 1984.

[Tomasi and Kanade, 1990]
C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method - 1. planar motion. Technical Report CMU-CS-90-166, Carnegie Mellon University, Pittsburgh, PA, September 1990.

[Tomasi and Kanade, 1991]
C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method - 2. point features in 3d motion. Technical Report CMU-CS-91-105, Carnegie Mellon University, Pittsburgh, PA, January 1991.