Global Methods for Image Motion Analysis

by

Venkataraman Sundareswaran

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science
New York University
October, 1992

Approved:__________________________________.
Robert Hummel
(Faculty adviser)

To
My Parents

ACKNOWLEDGEMENTS

I wish to thank my advisor Bob Hummel for his advice and guidance, in particular for teaching me the mathematical approach to solving problems in vision, and for showing me how to present the results. I would like to thank Stephane Mallat for introducing me to the field of visual motion analysis and for his encouragement and support. The experience with Rob Kelly and his group at Grumman and the support that they provided were valuable. Ken Perlin's interest in my work, his encouragement, and cheerfulness inspired me to tide many a hard day.

I would like to thank all my friends for all the inspiration, the encouragement, and all the good times we had together. In particular, my thanks go to Pankaj who inspired me in research, to Prasad from whom I have learned so much, and to Ronie and Laureen, the memories of whose company during my stay here I will cherish forever.

Support from Air Force contract F33615-89-C-1087 and ONR grant N00014-91-J-1967 is gratefully acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preliminaries

Visual experience forms a tremendously large part of our everyday sensory perception. Visual information is utilized to a surprisingly high extent to conduct everything from daily chores to complicated activities like flying a fighter aircraft. It has even been shown that suggestions from the visual experience supersede those from other sources like inertial inputs [40]. A vast amount of information is contained in the visual imagery obtained by motion. It is the processing of visual motion information that is the central concern of this thesis.

Let us consider visual motion perception. The motion of a sensor alone or its motion combined with the motion of other objects in the world gives rise to imagery that is rich in information about the world and the motion itself. Seeing an object hurled towards the observer provides enough data for the observer to avoid or catch the object! In less dramatic circumstances, we are able to navigate around obstacles by observing their relative motion as we walk.

Human beings and other organisms equipped with visual perception utilize changing information in the environment to judge self-motion [50] and to characterize the objects in the scene [76,34]. Technologically, the interest is to be able to provide robots with such a visual capability to enable them to navigate in known as well as unknown environments. This requires the ability to process the changing image of the environment. Thus the field of visual motion analysis is about analyzing sequences of images to determine egomotion and to extract information from the scene.

Research in motion analysis has been focussed on the problems of estimating the structure of the environment (structure from motion) and in computing the three-dimensional motion of the sensor (egomotion estimation, or passive navigation). The analysis is typically broken down into two stages. In the first stage, the motion of intensity points or feature points is computed. For this purpose, one can talk about small-scale motion and large-scale motion. Small-scale motion is the case where the motion in the image is small, i.e., comparable to the inter-pixel distance on a discrete image. On the other hand, large displacements result in large-scale motion. It has been suggested that the human visual system might have two different mechanisms to deal with these two different kinds of motion [73].

For the small-scale motion, the instantaneous motion of the intensity patterns can be defined; the vector field denoting the motion is termed the *optical flow*. For the large-scale motion, one can compute the displacements of important *features* in the image, and is termed the *correspondence*. The second stage of the analysis makes use of one of these two (optical flow or the correspondence) to estimate the structure and the motion parameters.

In this thesis, we are concerned with the situation where the optical flow has already been computed and we are interested in determining the egomotion parameters. The problem is rendered difficult because of the following two reasons [4]:

- The relationship between the three-dimensional motion parameters and the optical flow is nonlinear, and

- the space of unknowns has a dimensionality of five.

In other words, the straightforward methods to solve the problem are non-linear and involve minimization in a five-dimensional space. In developing our algorithms, we exploit the fact that even though the relationship between the parameters and the optical flow is nonlinear, it is actually bilinear (in some parameters). Certain observations about the structure of the equations that describe this relationship enable us to reduce the dimensionality of the search space to two. We also provide approximate algorithms that involve no search. One important feature of all the algorithms is that they are *global*; that is, they use all the available information in the image optical flow data to compute the parameters of motion.

1.2 Global methods

Methods to compute useful information from images can be either local or global. Local methods use local information, i.e., from a pixel or from a small neighborhood of a pixel, in order to determine the desired information. For example, edge detectors (algorithms to compute "edge points" in an image) use local information such as the pointwise derivatives of the image intensity function. On the other hand, Hough transform techniques use global information to determine the parameters of interest. Global methods are more robust, compared to local methods. We proceed to consider some standard image processing techniques.

Edge detection: This is a local process. Typically, first order derivatives of the image intensity function are used to determine the magnitude and the orientation of the edge. The process is sensitive to noise and can give very unstable results unless the image has been presmoothed.

Hough transform: This is a global process in which the parameters of a curve that we are looking for are obtained by using all available data. It is a very robust method that has been used in numerous applications.

Contour following: Local process; unstable.

Optical flow using the motion constraint equation: Local process; very unstable. It requires global smoothing techniques to obtain stable (but possibly incorrect) solutions.

Motion from correspondence: Linear algorithms exist but are very unstable. Use of global information makes it stable, but not completely.

Structure from motion: Local process; very unstable.

Stereo depth computation: Local process; very sensitive to noise unless there is a prior model of the environment.

Relaxation methods: Local iterative process that eventually integrates global information. For typical parameter values, the process is global, and produces stable results.

Calibration: Global processes used in calibration are robust, but tedious.

Object recognition: Global methods, such as geometric hashing, have also been shown to perform well [58].

Surface reconstruction: Global methods that use regularization are robust, but local methods are not.

Least squares methods: Least squares error methods which use redundant (globally) well-distributed data produce more robust results than those that use the minimum data required, with noisy data.

In summary, techniques that are global are more tolerant to noisy inputs, as opposed to local techniques which tend to be sensitive. This is only to be expected because errors arising due to (unsystematic) noise tend to cancel out when a large region of the image is used in the analysis.

For the case of a sensor moving in a static environment, information about the motion parameters is contained throughout the image. A change in the parameters affects the perceived motion everywhere. Thus instead of moving forwards, if the sensor moves sideways, the perception changes completely everywhere. On the other hand, structure (depth) information is contained locally; i.e., the depth to a scene structure affects the perceived motion (on the image plane) in a local fashion, namely, only the part of the image onto which the structure is projected. Thus, it is necessary to do local processing if the goal is to compute the structure of the scene. However, if we need to compute the motion parameters, it is to our advantage to make use of all the available information; such a global process will be more robust than a local process that uses only a few points.

Note that if self-moving objects are present in the scene, the algorithms presented here are not directly applicable. There is a need to segment the image (flow) into regions of same relative motion. This problem is not addressed in this thesis. However, if the segmented flow field is available, the algorithms can be applied to each of the regions to obtain the relative motion parameters.

1.3 Overview of the thesis

In this thesis, we present the results of algorithms to compute the parameters of the sensor motion. In particular, we describe the *flow circulation algorithm* to determine the rotational parameters using the curl of the *optical flow field* (a vector field of the instantaneous image motion), which under many conditions is approximately a linear function. The coefficients of the linear function are the desired rotational parameters. Instead of the curl values, we can use *circulation values*, defined to be contour integrals of the optical flow field on the image plane, resulting in robustness. We also describe a second algorithm that determines the translational parameters of the motion. The inner product of the optical flow field and a certain circular vector field gives rise to a scalar function that is of a particular quadratic polynomial form when the center of the circular field is chosen appropriately. This correct choice of the center is related to the translational parameters and can be found by projecting the inner product function onto suitable subspaces determined by the quadratic polynomial form.

A wire diagram representing the contents of this thesis is shown in Fig. (1.1). The motion parameter estimation is shown divided into two: rotational parameter estimation and translational parameter estimation. The different methods developed here appear at the leaves of the wire diagram.

In Chapter 2, we present a review of existing methods for motion analysis. In Chapter 3, we develop the relationship between the motion of a point in the three-dimensional scene and the instantaneous vector that is observed on the image plane. In Chapter 4, we present the flow circulation algorithm, a method to determine the rotational motion of the sensor. In Chapter 5, we describe methods to estimate the translational motion of a sensor; these methods constitute the FOE search algorithm. In Chapter 6, we analyze the applicability of the various methods presented in the thesis, considering various scene and motion situations. In Chapter 7, experimental results with synthetic data are presented. In Chapter 8, we present experimental results using real data,

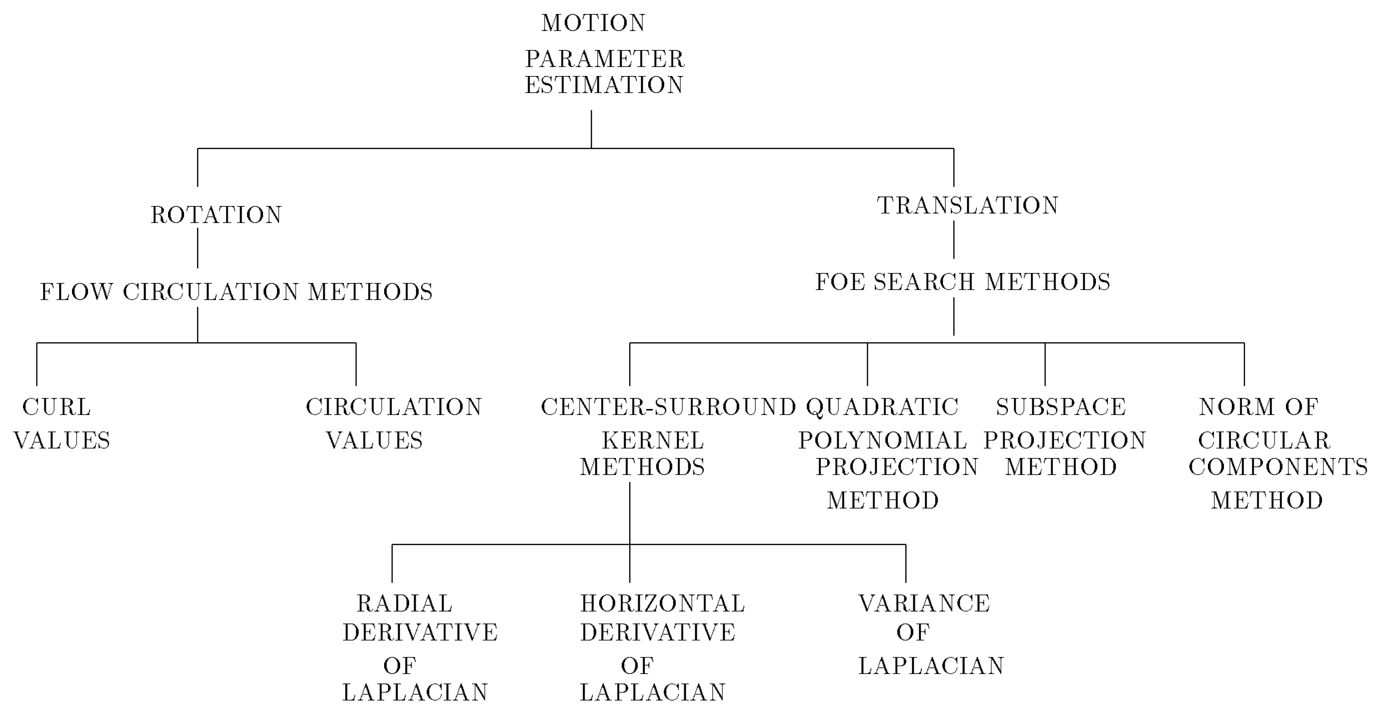

MOTION PARAMETER ESTIMATION
ROTATION
FLOW CIRCULATION METHODS
CURL VALUES
CIRCULATION VALUES
TRANSLATION
FOE SEARCH METHODS
CENTER-SURROUND KERNEL METHODS
QUADRATIC POLYNOMIAL PROJECTION METHOD
SUBSPACE PROJECTION METHOD
NORM OF CIRCULAR COMPONENTS METHOD
RADIAL DERIVATIVE OF LAPLACIAN
HORIZONTAL DERIVATIVE OF LAPLACIAN
VARIANCE OF LAPLACIAN

Figure 1.1: Thesis contents: a wire diagram.

and finally provide a summary and directions for possible future work in Chapter 9.

Chapter 2

Review of Previous Work

2.1 Introduction

The problem of analyzing visual motion has attracted the attention of a lot of researchers. It will be noted that a great majority of the work in visual motion analysis has been with the extraction of structural information (structure from motion, or SFM). It is difficult to do justice to the numerous publications in this area. We will briefly survey the results that are relevant to the material presented in this thesis. An extensive review of visual motion algorithms can be found in [3]. A recent analysis of optical flow computation techniques is in Barron et. al. [7].

2.2 Problem statement

Our concern here is the situation where the sensor is in motion and the objects in the environment are rigid and stationary; we would like to determine the motion of the camera. The other problem of interest is to determine the structure of the environment. A more general scenario is one in which not only the sensor is in motion, but some objects in the scene are also moving, as in the case of driving on a busy highway; in addition, the objects in the scene could be non-rigid, like clouds or people. It has been noted [72] that the imposition of the rigidity constraint is one way of dealing with the ill-posedness of the problem. In our algorithms, we will restrict our attention to the case of rigid, static objects.

Formally, the problem is, given a sequence of images, we would like to determine the motion parameters (the translational and rotational velocities of the sensor). This is typically done in two stages. In the first stage, a representation of the motion of image features is computed. This could be in the form of a dense vector field denoting the instantaneous motion of intensity points. This vector field is the *optical flow field*. Alternatively, one could compute the corresponding points in two consecutive images. This is usually a sparse representation. In the second stage, the parameters of motion and the structure of the scene are computed, using the representation produced by the first stage. We begin by reviewing the methods for computing the optical flow; then we look at

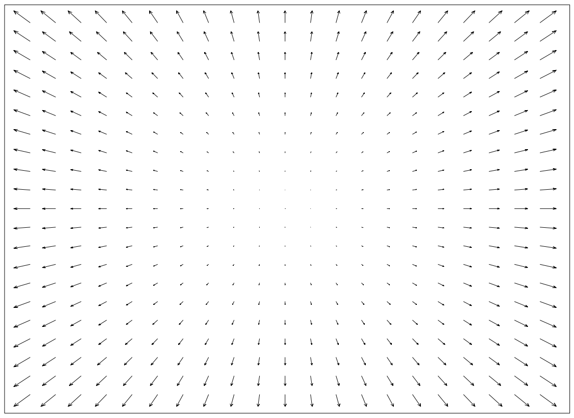


Figure 2.1: Optical flow field created due to forward motion of the sensor.

methods that use optical flow or correspondence to compute motion and structure.

2.3 Optical flow computation

Optical flow is an instantaneous representation. It is a two-dimensional vector field denoting the instantaneous motion of intensity points in the image. For instance, a forward motion of the camera results in an optical flow field such as the one shown in Fig. (2.1). In the next chapter, we will illustrate the relationship between the motion of the camera and the resulting optical flow. Of particular interest is the Focus of Expansion (FOE) which is a point in the image towards (or from) which features in the image seem to move. In Fig. (2.1), it is located at the middle of the image. Several methods are available for the computation of optical flow and for finding correspondence. We will review some of them here. Our interest in optical flow data arises since the algorithms proposed in this thesis make use of the optical flow field in order to determine the camera motion. There are gradient-based methods and energy model-based methods to compute optical flow.

2.3.1 Gradient-based methods

The gradient-based methods depend on the image motion constraint equation. If the intensity function is represented by $E(x, y, t)$, assuming that the motion can be viewed as a local translation of intensity patches, we have, for such a patch,

$$\frac{dE}{dt} = 0 \tag{2.1}$$

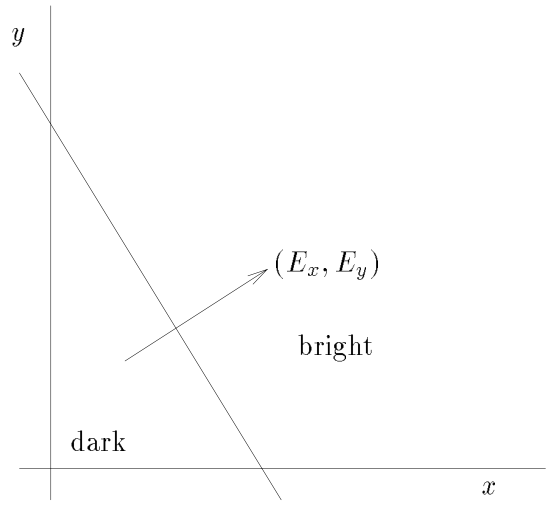


Figure 2.2: The gradient direction

Equivalently,

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0.$$

Or, simply,

$$E_x u + E_y v = -E_t \tag{2.2}$$

where E_x, E_y and E_t are the partial derivatives of E and (u, v) is the optical flow at a point. This is the motion constraint equation [29]. Two assumptions have been made in writing down Eqn. (2.2). The first one is that the intensity function can be approximated well by the linear terms in its Taylor series expansion (in other words, E is locally planar). We note here that this assumption is violated at locations with strong intensity gradients that occur in places like object boundaries and textured regions. The second assumption is that the motion can be locally modeled as translation. Again, violations occur at locations of motion transparency (two motions at the same point) and near motion boundaries.

From Eqn. (2.2) we can compute the projection of the optical flow in the direction (E_x, E_y) (see Figure 2.2) because we can readily estimate E_x, E_y and E_t from two or more consecutive image frames.

There are two unknowns u and v, but there is only one equation. This is one manifestation of the *aperture problem* which says that only that component of velocity in the direction (E_x, E_y) of the intensity gradient can be estimated. The other component – the one in the orthogonal direction – cannot be computed directly. This is a loss of information. Thus, what we have is a *raw* optical flow (also called *normal flow*) which needs to be processed further to get the *actual* optical flow. One can try to recover this lost information by making some assumptions on the structure of the solution. One such assumption is that the velocity can only vary smoothly. Horn and Schunk

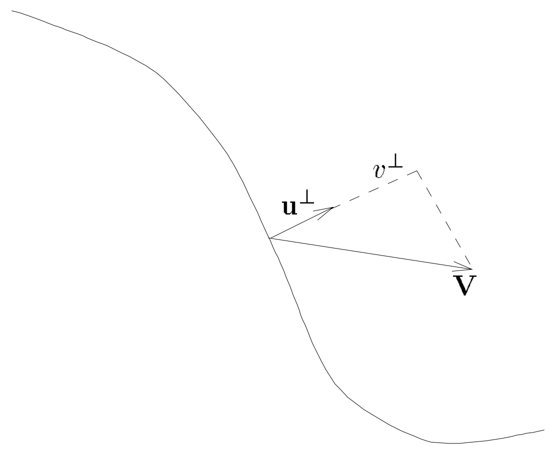


Figure 2.3: Normal component along a contour

minimize the function

$$\epsilon^2 = \int\int[(E_x u + E_y v + E_t)^2 + \alpha^2((\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2)]dxdy. \tag{2.3}$$

This function is a measure of the deviations from both the smoothness and the motion constraints; α^2 controls the relative cost of the two. The hope is that minimizing ϵ^2 would recover the correct velocity. This results in a pair of equations for each point in the image and an iterative method is used to obtain a solution. Horn and Schunk [30] show some experimental results using simulated data.

Hildreth [27] first finds contours in the image. Then she calculates the optical flow only along these contours using Eqn. (2.2). The resulting optical flow is smoothed along the contours using a conjugate gradient technique. The assumption here is that the velocity is smooth along the boundary of a body.

The contours are found by convolving with a $\nabla^2 G$ operator and selecting the zero-crossing contours. The *normal* component of velocity, namely the one that can be found using Eqn. (2.2), is computed along points on these contours. The next step is to compute the *actual* flow. This is done by assuming that the velocity varies smoothly along the contours. Also, the resulting velocity field should have normal components that are as close to the computed values as possible. These two constraints are captured in the function

$$\Theta = \int[(\frac{\partial \mathbf{V_x}}{\partial \mathbf{s}})^2 + (\frac{\partial \mathbf{V_y}}{\partial \mathbf{s}})^2]ds + \beta\int[\mathbf{V}\cdot\mathbf{u}^{\perp} - v^{\perp}]^2 ds$$

where the symbols have meanings as shown in Figure 2.3. Here, $\mathbf{V} = (\mathbf{V_x}, \mathbf{V_y})$ is the velocity vector field. The partial derivatives of the function are with respect to the arclength parameter s that is along the contour. That is, the partial derivatives give the rate with which the velocity components change as one marches along the contour. The symbol $\mathbf{u}^{\perp}$ is the unit vector normal

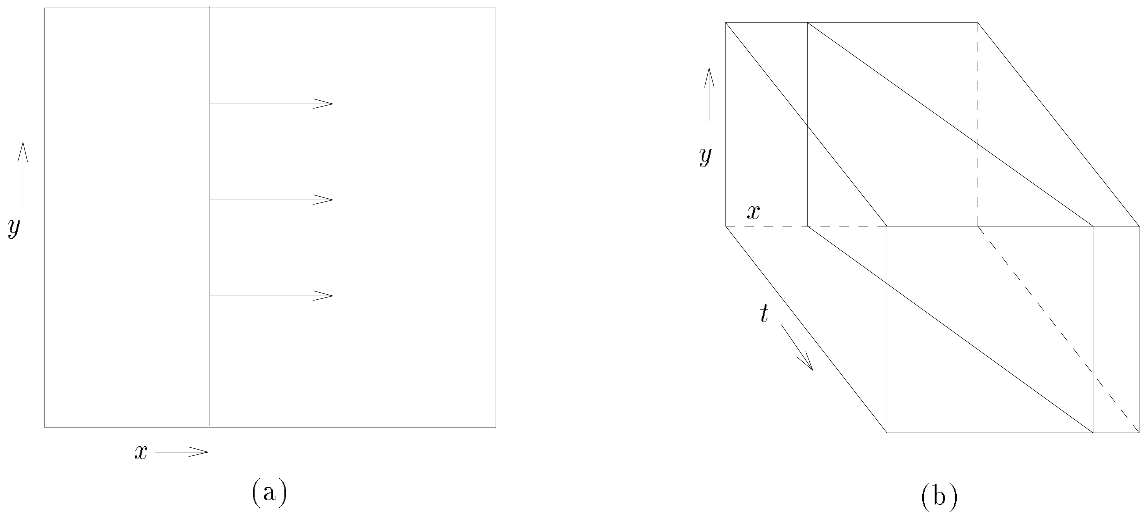


Figure 2.4: Orientation in spatiotemporal domain

to the contour and $\mathbf{v}^{\perp}$ is the computed normal component. Thus, the first integral measures the departure from smoothness and the second integral measures the deviation from fidelity to the computed normal components. The value β is a weighting factor which can be adjusted depending on the confidence on the computed normal components.

Hildreth [27] gives a discrete formulation of this function and uses a conjugate gradient algorithm to minimize this function. Some examples using real images are given. She also demonstrates that the algorithm is consistent with psychophysical results by applying the algorithm on examples from perceptual experiments. Other gradient-based methods can be found in [44,49].

2.3.2 Energy model-based methods

Adelson and Bergen [1] (also see [12]) point out that translation appears as orientation in the x-y-t (spatiotemporal) domain, and that the translation can be detected as such. This is quite easy to see. Consider the picture of a line moving to the right, as shown in Figure 2.4(a). Figure 2.4(b) shows the orientation in the x-y-t space. The velocity varies inversely with the slope of this plane. So, by measuring this slope, or orientation, we can compute the velocity. Filters can be designed to detect any particular orientation. However, it is desirable that such filters be spatiotemporally separable; these are simpler not only for processing images on a machine but also to construct out of simple neural mechanisms. In particular, Adelson and Bergen [1] develop the model with the latter goal in mind.

A problem with spatiotemporally oriented filters is that they are phase-sensitive. This means that an input containing a drifting sine wave would induce an output that swings between positive, zero and negative values, even though the sine wave is moving with a constant velocity. This would be the case with any linear filter. A way of overcoming this problem is by using two linear filters whose responses are 90 degrees out of phase – called a *quadrature pair* – and by taking the sum of squares of their outputs. An example of such a quadrature pair is a set of oriented Gabor filters, one with a cosine phase and the other with a sine phase.

The key observation behind another energy model is that the Fourier transform energy of a translating image lies on a plane whose slope depends on the velocity of the movement. This can be derived quite easily [60]. The idea behind Heeger's algorithm [22] is to estimate the orientation of this plane by sampling the energy of the Fourier transform and by fitting a plane.

The sampling is done by Gabor filters. A one-dimensional sine-phase Gabor filter is given by

$$g(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp^{(\frac{-t^2}{2\sigma^2})} \sin(2\pi\omega t)$$

The power spectrum is a pair of gaussians centered at ω and $-\omega$. Convolving this filter with an one-dimensional function is equivalent to multiplying their Fourier transforms. The result is a sampling of the power spectrum of that function around ω and $-\omega$, yielding the *Gabor energy.* Such a filter is called a Gabor-energy filter.

A sine-phase filter and a cosine-phase filter are convolved with the image sequence. The power outputs are squared and added. This *quadrature pair* is employed to avoid the phase-dependency, as outlined earlier in this section. The model uses a family of Gabor-energy filters; the motion energy is predicted by computing the response of the filters to a random-dot texture translating with velocity (u, v). This is an analytical expression involving u and v. The computed motion energy has to be close to this predicted value if the pattern is moving with velocity (u, v). Thus, a least-squares estimate for (u, v) can be found by minimizing some measure of the difference between the computed and predicted motion energies. This amounts to finding the plane that best fits the computed energy in the three dimensional Fourier space.

Other methods using spatiotemporal filtering can be found in [62,15,18].

2.4 Correspondence computation

Many algorithms have been developed to solve the matching problem in stereo vision, where there are two pictures of the same scene taken from two different camera positions. This can be considered equivalent to a camera motion; that is, the camera took the picture from one position and then *moved* to the other position to take the second picture. However, this is only a particular case of camera motion. In general, the camera could have a rotational movement; and it might translate *towards* the objects. Also, objects in the scene might move. Thus, when compared to stereo, the matching problem for motion applications is more difficult. Matching can be done over two frames at a time or over more frames. We describe a correlation based scheme [5] that computes dense correspondence (i.e., for each pixel). We make use of an implementation of this scheme to compute the optical flow field data for our motion computation.

Anandan's algorithm [5] matches features at multiple resolutions using a *Laplacian pyramid* [11,10]. He first matches at the coarsest level where even a large-scale movement will be seen as a sub-pixel displacement. The result is then passed onto finer levels by an *overlapped pyramid projection* scheme. The coarse estimate is used at the finer level to get a better estimate of the motion. The final estimates are obtained at the finest level.

The Laplacian pyramid [10] can be constructed by first building a Gaussian low-pass-filter pyramid from the input image and then by computing the difference between adjacent levels of the Gaussian pyramid. The result is a set of band-pass-filtered images. The matching starts from the coarsest level where the finer details are not present. The matching is done using a correlation scheme based on the minimization of the sum of squared differences (SSD) [59]. For every pixel of the first image, a candidate pixel on the other image is found by using the results from the coarser level. The SSD for this pair is a Gaussian weighted sum of the squared differences between the values of the corresponding pixels in the 5×5 windows centered around the source and the candidate pixels.

A pixel at level l in the pyramid passes its estimates to all the pixels in a 4×4 area in the next finer level $l+1$. Thus each pixel receives estimates from 4 *parents* above (hence the name *overlapped pyramid projection* scheme) , and the SSD measure is minimized over these. A confidence measure is established by observing the following: the location of the minimum of the SSD which is supposed to give the disparity is frequently not at the exact disparity. Its distance from the actual disparity depends on the type of surface being matched: the best is for unoccluded corner points and the worst for homogeneous surfaces and occluded corner points. The accuracy of the match can be estimated by finding the local curvature of the SSD surface. The larger the curvature, the more accurate the answer is. This confidence measure, which is useful in the smoothing process, consists of two magnitudes ($c_{\max}$ and $c_{\min}$) and two direction vectors ($\mathbf{e}_{\max}$ and $\mathbf{e}_{\min}$, which are the unit vectors along the principal axes of the SSD surface). The magnitudes are given by

$$c_{\max} = \frac{C_{\max}}{k_1 + k_2 S_{\min} + k_3 C_{\max}}$$

and

$$c_{\min} = \frac{C_{\min}}{k_1 + k_2 S_{\min} + k_3 C_{\min}}$$

where k_1, k_2 and k_3 are normalization parameters, $S_{\min}$ is the SSD value corresponding to the best match, and $C_{\max}$ and $C_{\min}$ are the curvatures of the SSD surface along the axes of maximum and minimum curvature.

If we assume rigid object motion, we can *alter* the displacements with low confidence measure by getting information from those with high confidence measure. The idea is to find a vector field $\{\mathbf{u}\}$ such that the quadratic functional

$$E(\{\mathbf{u}\}) = E_{sm}(\{\mathbf{u}\}) + E_{ap}(\{\mathbf{u}\})$$

is minimized. The smoothing part is based on the error formulation

$$E_{sm}(\{\mathbf{u}\}) = \int\int [(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2)] dxdy. \qquad (2.4)$$

where $\{\mathbf{u}\}$ is the set of the displacement vectors $\mathbf{u}(x,y) = (u(x,y), v(x,y))$. The function in Eqn. (2.4) is the same as in Eqn. (2.3), except that $E_{sm}(\{\mathbf{u}\})$ contains only the smoothness constraint. The approximation error E_{ap} measures how well the computed field approximates the measured displacement field:

$$E_{ap}(\{\mathbf{u}\}) = \sum_{x,y}[c_{\max}(\mathbf{u}\cdot\mathbf{e}_{\max} - \mathbf{d}\cdot\mathbf{e}_{\max})^2 + c_{\min}(\mathbf{u}\cdot\mathbf{e}_{\min} - \mathbf{d}\cdot\mathbf{e}_{\min})^2]$$

where $\mathbf{d}$ gives the match estimate at each point. Minimizing the error function $E(\{\mathbf{u}\})$ in discrete form results in a system of coupled linear equations and is solved using the Gauss-Seidel relaxation algorithm. After the displacements and the associated confidences are computed within each level, the displacement field is smoothed before it is projected to the next level in the pyramid.

This algorithm gives good qualitative results in most regions of images [5], but has problems with occluded regions and at motion boundaries. For instance, displacements in areas of background near the boundary of a moving object are influenced by the moving object; occluded areas do not have reliable local estimates, and so the more reliable estimates from the neighborhood are propagated into these areas.

Barnard and Thompson [6] give an iterative relaxation algorithm which tries to find the best possible match for two sets of points (features) in two images. First, a set of candidate points are chosen from each image by using some interest point detection technique, such as the Moravec's operator [48]. The next step is to determine the matching. For each point P_i in the first image, a set of candidate points are chosen from the second image as those within some distance from P_i (compute this distance by imagining that P_i is in the second image). With each P_i, we associate a set L_i of labels. Each label l is either a disparity vector or a distinguished symbol l^*. Each label is a potential disparity for that point; l^* means that no match in the second image and hence no disparity can be assigned to the point. Initially L_i contains only a l^*. In addition, there is a number $p_i(l)$ associated with each label l which can be interpreted as the probability that the point P_i has disparity l. A relaxation procedure iteratively modifies all these probabilities so that for each point one of the probabilities, say, that of label l_1 is expected to dominate while the others tend to zero. This suggests that l_1 is the disparity for that point.

The algorithm proposed by Sethi and Jain [61] uses more than two images in a time sequence to find matching points. The main idea is *path coherence* which says that the trajectory traced by a point is smooth. The trajectory of a point is the curve connecting it through all its matching points in the sequence. The fundamental assumption here is that objects are in constant motion, without undergoing sudden changes in their movements. A measure of the smoothness of the trajectory is defined and it is minimized to get the best match satisfying path coherence.

An algorithm that combines both a gradient-based method and a feature-based method has been proposed by Heitz and Bouthemy [25]. It is based on a Bayesian formulation using Markov random fields. The motion constraint in Eqn. (2.2) is used to compute flow, simultaneously with a Moving Edge constraint described in [8]. Validation factors are defined for each of these constraints.

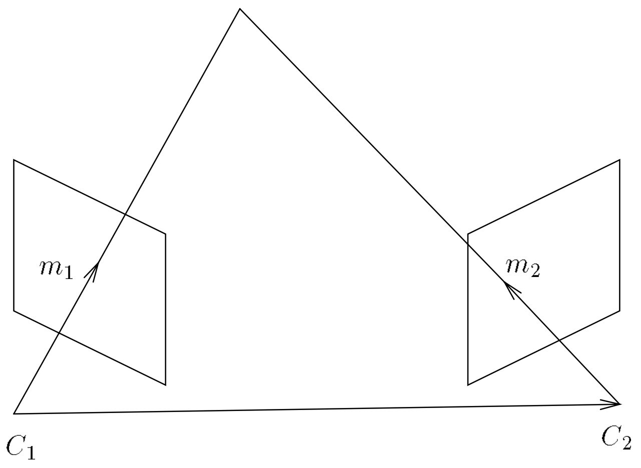


Figure 2.5: The Planarity Constraint

The distribution of observations (provided by the constraints) and motion labels (the flow field and motion discontinuities) is specified using a coupled Markov Random Field (MRF) model. Associated with the distribution is the *energy function* of the MRF. Finding the Maximum A Posteriori (MAP) estimate is equivalent to minimizing this global energy function. The energy minimization is done using a deterministic relaxation procedure. Extension to multiresolution and experimental results are presented in [25]. An implementation of this algorithm has been used to compute optical flow fields used in some of the experiments whose results are presented in this thesis. An empirical comparison of various optical flow computation techniques can be found in [7].

2.5 Parameter estimation: previous work

One of the most important applications of motion analysis is considered to be the computation of the depths of the various points in the scene. The interest in reconstructing the structure of the scene from motion information has captured the attention of lot of researchers. It is well known that with a single camera, it is only possible to estimate depths up to a scale factor because two features, one twice as big as the other, twice as far, and twice as fast, will give rise to the same motion (on the image plane) under perspective projection.

It is of equal interest to determine the parameters of motion of the sensor. Two approaches are possible in this regard. The first is to attempt to recover the structure of the scene; this involves local processing because depth information is local. We will note that most algorithms take this route. The second approach is to compute the motion parameters (the translation and rotation of the sensor) first. This is non-local information; so we can expect to obtain more robust methods when compared to the first approach. Examples of the second approach are the methods presented in this thesis.

Most algorithms assume that either the correspondence or the optical flow field is available. The algorithm of Longuet-Higgins [41] (see also [71],[13] and [80]) assumes that the correspondence of points between two consecutive frames is available. The quantities computed are the translation $\vec{t}$ and the rotation R for moving the camera from C_1 to C_2 (see Fig. (2.5)). The constraint for every matching pair (for instance, m_1 and m_2 in Fig. (2.5)) is that the lines of projection for these two image points meet in space (simply because they are the images of the same world point). This *planarity* constraint can be expressed as follows:

$$\vec{C_1 m_1} \cdot \vec{t} \times R\vec{C_2 m_2} = 0 \tag{2.5}$$

Equivalently,

$$\vec{C_1 m_1} \cdot T_x R \vec{C_2 m_2} = 0$$

where T_x is a matrix formed out of the elements of $\vec{t}$. The constraint, applied to each pair of corresponding points, results in an equation linear in the elements of the matrix $E = T_x R$. With correspondence for eight points, we obtain eight equations as in Eqn. (2.5) and we can solve for the elements of the matrix E. From E, we can compute the translation vector $\vec{t}$ and the rotation matrix R, as described in [41] or in [13]. The algorithm in [13] is more robust because it uses redundancy to combat noise. That is, it uses more than eight points in solving for E. A cogent presentation of the method can be found in [80].

Another algorithm, based on optical flow field input, is described by Heeger and Jepson [24]. This makes use of the bilinear form of the optical flow equations:

$$\vec{\theta}(x, y) = A(x, y, T)p(x, y) + B(x, y)\vec{\omega} \tag{2.6}$$

where $\theta(\vec{x, y})$ is the optical flow at (x, y), T is the three parameters of the translation of the camera and $\vec{\omega}$ is the vector of rotations with respect to the three axes. The 2×3 matrix B depends only on the image position (x, y) and the 2×1 matrix A depends only on the image position and the translation T. The inverse depth is denoted by $p(x, y)$. By choosing the optical flow at five points, the Eqn. (2.6) can be rewritten in the form

$$\vec{\theta} = C(T)\vec{r}$$

where $\vec{\theta}$ is a 10×1 vector of optical flow at the five points, $C(T)$ is a 10×8 matrix depending on the translation T and $\vec{r}$ is a 8 element vector consisting of the inverse depths of the five points and the three axis rotations. Since $\vec{\theta}$ has to be in the *range* of the matrix $C(T)$, the appropriate value of T can be determined by minimizing the projection of $\vec{\theta}$ on the space which is the orthogonal complement to the range of $C(T)$. Knowing T, the rotations can also be computed [23]. We will return to this algorithm again to point out the similarity it bears to some methods described in this thesis.

Analysis of the optical flow field, also called the motion parallax field, has received considerable research attention. J. J. Gibson, in 1950, discussed the motion parallax field, and defined and discussed the importance of the focus of expansion [16]. Even earlier, Helmhotz had noted that the image flow field contained information about the depths to the objects [75]. Subsequent work by Gibson considered further the extraction of structural information of the scene from the flow field; for example, for the case of a pilot landing a plane, there is essentially no rotational component, and so the extraction of structure from the flow field due to translation is possible [17]. In a series of papers, Koenderink and Van Doorn study properties of the image flow field [36,37]. Their work begins the practice of analyzing the flow field induced on imaged surfaces of particular form, such as planar surfaces or gaussian protuberances. The goal is to produce local measurements that are invariantly related to properties of the surface shape.

In a famous paper, Longuet-Higgins and Prazdny[43] show that, in principle, the motion parameters and local surface curvature of an imaged surface may be determined from the local flow values and values of the spatial derivatives of the local flow field up to second order. The unknown parameters are the translation and rotation velocities and the surface normal at any given point. The focus of expansion is located using the residual flow field after subtracting the flow field due to rotation. There are special difficulties with the solution method when the imaged surface is planar, and it is clear that measurements of the second order derivative, in particular, will be noisy. However, since the computations at each point should yield the same motion parameters (and varying surface normals), the computations are redundant, and thus one can hope that stabilized algorithms are possible.

In a series of subsequent papers, Prazdny offered a variety of other algorithms for motion determination. In [54], a precursor to the FOE search algorithms presented in this thesis (also see [31,66]) appears. In this algorithm, the best rotation parameters are sought such that the flow field that remains after subtracting the vector field corresponding to the rotation parameters yields a pure expansion or pure contraction field, as will occur for the flow field due to translation only. The algorithm then simply involves a nonlinear minimization. More recent work by Burger and Bhanu [9] extends the Prazdny search algorithm to search for a "fuzzy" focus of expansion region. In [53], Prazdny shows how the computation of the translational and rotational velocities can be obtained from the flow velocities at a collection of distinct points (five of them are required), by solving a system of three cubic polynomial equations in three unknowns. An iterative method is used, and a good initial guess is required. The intention is that the processing should be local, although the equations hold for any distinct set of five points. A good survey and history of results until that point is provided in a separate paper by Prazdny [55]. The idea of searching over the possible focus of expansion points, instead of searching over the rotation parameters, as in the FOE search algorithms presented in this thesis, was introduced by Adiv [2]. The algorithm presented by Heeger and Jepson [24,19,21,32] uses a variant of the same error function as in Adiv [2]. Another related approach was presented by Maybank [46,45].

Nearly all researchers in motion parameter estimation realize that once one has *some* information, such as the location of the focus of expansion, or the value of the rotation parameters, all other parameters are easily obtained. Hence there is considerable motivation for separating the flow field due to rotational parameters from the flow field due to translational parameters. Already noted by Helmholtz [75] and mentioned by Longuet-Higgins and Prazdny[43] is the idea of making use of either depth discontinuities or motion parallax of a translucent surface (such as a dusty window). In these cases, the difference or jump in the flow velocities cancels the flow component due to the rotational parameters, leaving a flow dependent only on translational parameters. Providing there are enough such points, then the focus of expansion, and hence all other parameters, may be determined. Lawton and Rieger exploit this idea to build a system based on differences of neighboring flow velocities [57]. Unfortunately, noise tends to make this method rather unreliable. Lawton built another system that assumes that the rotational parameters are nearly zero, and thus finds the focus of expansion by means of a "Hough transform" technique [39]. A solution method that assumes that the translational parameters are zero would be quite easy; the flow circulation method presented in this thesis (also in [68]) provides an exact procedure, and other methods are straightforward.

In a series of papers, Waxman and collaborators revisited the problem of motion parameter estimation and local surface structure determination from local flow parameters (i.e., values of the flow velocities and derivatives of the flow velocities through second order). A solution method is presented for the case of planar surface patches and quadratic flow velocity fields [65] together with an analysis of the ambiguities, followed by a new method for quadratic surfaces and quadratic velocity fields [77] which improves upon an earlier method of Waxman and Ullman [78]. More detail and extensions to binocular image flows are given in [82]. In all of these works, the structure of the surface and the motion parameters are considered as the unknowns relative to a single image point. That is, the analysis is local. Measurements must be given of derivatives of flow velocity values through second order, in which case it is possible to solve for local surface structure up to a second order Taylor expansion. The curl of the flow field is one of the twelve "deformation parameters" (D_6 to be exact), and the "kinematic relation" for this parameter is precisely the equation that we require for the flow circulation algorithm. However, since the problem that Waxman addresses is the exact computation of motion parameters coupled with surface parameters based upon local data (deformation data at a single point), the approximations and the global method leading to the flow circulation algorithm is missed. Using instead correspondences of curves, the local quadratic nature of the velocity field can be obtained providing a sufficient number of curves are matched, as studied by Wohn [79,82] (who also makes use of some temporal smoothing). Solution methods based on the use of a sufficient number of correspondences of points, without involving explicit derivatives of the flow field, and without explicit representation of surface parameters, are provided by Jerian and Jain [33]; their work, like that of Tsai and Huang [71] is directed for the case of determining rotation and translation parameters from correspondences, as opposed to rotation and translation *velocities* from an image velocity flow field. An approach to factor a matrix of motion measurements into two matrices that represent shape and motion, for the case of orthographic projection has been proposed by Tomasi and Kanade [70]. Direct methods for recovering the motion parameters have been described by Horn and coworkers [28,52]. The direct methods apply only for restricted kinds of motion, such as pure translation.

Clearly, the surface parameters, to the extent that they can be recovered, must be based on local measurements. The motion parameters, however, are global. Most researchers have noted that their methods provide redundant computation of motion parameters, providing a test for the rigidity assumption. Unfortunately, many of the algorithms forgo the stability that can be obtained by deriving the motion parameters from an integrative approach i.e., by making use of the constancy over the image. Of course, if one's focus is on surface parameter reconstruction, then local processing is essential. However, if one first derives the motion parameters, and then uses knowledge of the motion parameters to assist in surface depth estimates, then global methods may be used for motion parameter estimation. Methods that can potentially make use of distributed information include those of Prazdny [53], Adiv [2], and Heeger and Jepson [21].

The curl of the flow field, which is the basis of our flow circulation algorithm, has long been recognized as an important property of the motion field. However, the observation that the curl of the field is approximated by a linear function, and the use of that approximation to determine the rotational parameters, appears to be new. Koenderink and Van Doorn [36] calculate explicitly the curl (and other functionals). We begin with their computation of the curl (converted to planar image coordinates as opposed to polar coordinates), but make use of the function to solve for the rotational parameters. They instead studied the properties of these elementary fields in the case of an observer moving with respect to a plane [37,38]. The existence of receptive fields sensitive to the curl (and also to the divergence) was hypothesised by Koenderink and Van Doorn [36] and Longuet-Higgins and Prazdny [43], but the motivation is for surface structure determination, and not for global synthesis of motion parameters. Regan and Beverly [56] followed up on the hypothesis of Longuet-Higgins and Prazdny by conducting psychophysical experiments, and concluded that the existence of vorticity receptors is plausible. Cell recordings in the dorsal part of MST of Macaque monkeys suggest cells tuned to expansion/contraction and other cells sensitive to rotation [69]. More recently, Werkhoven and Koenderink [81] have considered methods for directly computing flow field invariants, including the curl, from time-varying image irradiance data. The considerable interest and evidence for the importance of the curl of the flow field, or equivalently, the circulation values, lends credence to the flow circulation algorithm presented in this thesis.

In the next chapter we relate the optical flow field to the motion parameters of the sensor. The model so obtained would enable us to point out how one can estimate the motion parameters by suitably processing the optical flow field.

Chapter 3

The Optical Flow Equations

3.1 Introduction

We consider the imagery that is produced by a moving sensor. The sensor produces images whose contents are altered over time due to either the motion of the sensor, or the motion of objects in the scene, or both. Thus, two kinds of motion are possible: sensor motion and object motion. The sensor could move about in its environment. A camera mounted on an Autonomous Land Vehicle is one example. On the other hand, objects in the scene could move. In general, both kinds of motion can happen simultaneously. Mathematically, it is natural to consider the *relative motion* between the sensor and each of the objects in the scene. The prerequisite for using relative motion in image processing is the availability of segmented images, namely, the segregation of the different objects in each image. This would conceivably be done at an earlier stage of processing, using either a static segmentation technique operating on each image in turn or a dynamic method using the motion information obtained from more than one image.

Optical sensors like the retina or a camera film are sensitive to illumination; so, the motion information is available as changes in the illumination intensity patterns falling on the sensor array. The instantaneous movement of the intensity pattern on the sensor array can be represented by a vector field; the vector field value at location P represents the instantaneous motion of the pattern at the point P. The vector field is referred to as the *optical flow field*, or simply, the *flow field*. It is customary to assume that the flow field represents a projection of the instantaneous three-dimensional motion even though this is not always true [74].

We need to model the flow field in terms of the parameters of the real world motion and the structure of the environment, in order to interpret the flow field. The aim of this chapter is to present a model of the flow field as a set of equations that relate the flow field to the motion parameters and the scene structure, and to make certain observations about the form of these equations.

One of the issues in the modeling is the choice of a coordinate system. The coordinate system can be either ego-centric or exo-centric. The ego-centric system is one in which the coordinate system is fixed with respect to the sensor (typically, the origin will be located *within* the sensor). This means that the coordinates of a point that is not rigidly attached to the sensor will change with time as the sensor moves. In the exo-centric system, the coordinate system is located outside the sensor, usually in a static location or object.

A second, less critical choice concerns the representation which is typically either cartesian or spherical. The cartesian coordinate system is a natural choice for cameras with a planar imaging array, and the spherical coordinate system is the best choice for the human eye because the retina is roughly spherical.

In this chapter, we derive the optical flow equations for an ego-centric cartesian coordinate system and observe some of the properties of these equations. The results could be derived in other coordinate systems, but would be more complicated.

3.2 The coordinate systems

The optical flow equations for the perspective projection of a scene on a planar imaging sensor are well-known and understood. We provide here an independent derivation, similar to the development in [21,29,43]. We will adopt the cartesian coordinate system and the motion parameters as shown in Fig. (3.1). The three-dimensional coordinate system is centered at the camera's center of projection and the coordinates are denoted using upper-case letters X, Y and Z. This is the sensor coordinate system. The imaging surface is a plane located at $Z = f$, f being the focal length. The x and y coordinate axes lie on this plane, with the origin at the center of the image. The transformation from spatial coordinates to the image coordinates is given by the equations of perspective projection which can be easily derived using a pin-hole approximation to the lens:

$$x = fX/Z, \quad y = fY/Z$$

where $(X, Y, Z) = (X(x,y), Y(x,y), Z(x,y))$ is the position of the point in three-dimensional space that is imaged at (x, y). We assume that the objects in the scene are fixed and rigid. We want to determine the flow field which is observed due to the motion of the sensor, as a function of the image coordinates x and y.

The camera moves with a translational velocity of $T = (v_1, v_2, v_3)$ and a rotational velocity of $\omega = (\omega_1, \omega_2, \omega_3)$. The values v_1, v_2 and v_3 are the X, Y and Z components of the instantaneous vector describing the translation of the sensor. The rotation that the sensor coordinate system undergoes can be described either using a rotation matrix R, or equivalently, by the angular velocity magnitude M_ω about an axis of rotation A_ω. In the latter case, one can simplify the details by defining an *angular velocity vector* ω that has magnitude M_ω and direction A_ω. Again, ω_1, ω_2 and ω_3 are the components of ω.

The optical flow $V = (u, v)$ at the image point (x, y) can be obtained by projecting the three-dimensional relative velocity of the feature point $\mathbf{X}$ imaged at (x, y) onto the image plane. The

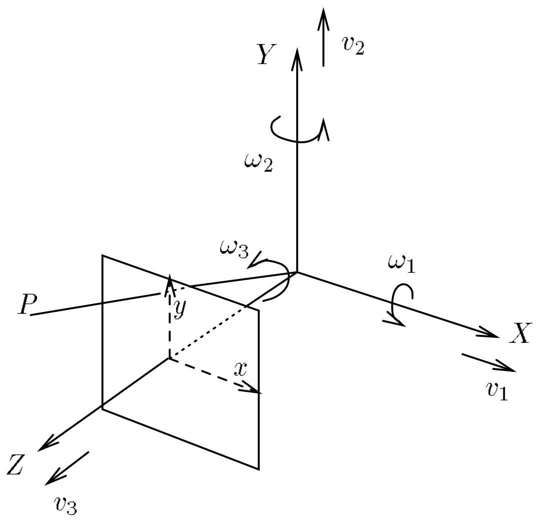


Figure 3.1: The coordinate systems and the parameters

relative velocity of a point $\mathbf{X}$ with respect to the coordinate system has two parts; one is due to the translation T of the coordinate system, and the other is due to the angular velocity vector ω. The three-dimensional velocity of the feature point $\mathbf{X} = (X, Y, Z)$ can be easily seen [63] to be

$$\dot{\mathbf{X}} = -T - \omega \times \mathbf{X}. \tag{3.1}$$

Here we use the convention $\dot{\mathbf{X}}$ to denote the temporal derivative of $\mathbf{X}$ (i.e., $\frac{\partial \mathbf{X}}{\partial \mathbf{t}}$). In terms of the component values,

$$\begin{aligned} \dot{X} &= -v_1 - \omega_2 Z + \omega_3 Y, \\ \dot{Y} &= -v_2 - \omega_3 X + \omega_1 Z, \\ \dot{Z} &= -v_3 - \omega_1 Y + \omega_2 X. \end{aligned}$$

The flow on the image plane is

$$(u, v) = (\dot{x}, \dot{y}).$$

Thus,

$$\begin{aligned} u &= \frac{f\dot{X}}{Z} - \frac{fX\dot{Z}}{Z^2} \\ &= (-\frac{fv_1}{Z} - \omega_2 f + \omega_3 y) - x(-\frac{v_3}{Z} - \frac{\omega_1 y}{f} + \frac{\omega_2 x}{f}), \\ v &= \frac{f\dot{Y}}{Z} - \frac{fY\dot{Z}}{Z^2} \\ &= (-\frac{fv_2}{Z} - \omega_3 x + f\omega_1) - y(-\frac{v_3}{Z} - \frac{\omega_1 y}{f} + \frac{\omega_2 x}{f}). \end{aligned}$$

Making explicit the dependence on the image coordinates and rearranging terms, we get

$$\begin{aligned} u(x,y) &= \tfrac{1}{Z(x,y)}\left[-fv_1 + xv_3\right] + \omega_1\left[\tfrac{xy}{f}\right] - \omega_2\left[f + \tfrac{x^2}{f}\right] + \omega_3 y, \\ v(x,y) &= \tfrac{1}{Z(x,y)}\left[-fv_2 + yv_3\right] + \omega_1\left[f + \tfrac{y^2}{f}\right] - \omega_2\left[\tfrac{xy}{f}\right] - \omega_3 x. \end{aligned} \tag{3.2}$$

Here, $u(x,y)$ and $v(x,y)$ are the x and y components of the optical flow field $V(x,y)$.

In a similar fashion, one can derive the equations of the flow field for a spherical imaging surface. All procedures that work in the cartesian coordinate system can be transformed to work in the spherical coordinate system.

3.3 Observations on the equations

The equations in (3.2) have certain properties that we note here. The flow equations may be grouped into the sum of two terms, as is noted, for example, in [43]: the first term gives the flow field due to the translational components, and is modulated by the inverse depths, and the second term is a flow field due to the rotational components, and is independent of the depths. Thus,

$$V(x,y) = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = V_v(x,y) + V_\omega(x,y), \tag{3.3}$$

with (for the case $v_3 \neq 0$)

$$\begin{aligned} V_v(x,y) &= v_3\rho(x,y)\begin{bmatrix} x-\tau \\ y-\eta \end{bmatrix}, \\ \tau = \frac{fv_1}{v_3},\ \eta &= \frac{fv_2}{v_3},\ \rho(x,y) = \frac{1}{Z(x,y)} \end{aligned} \tag{3.4}$$

and

$$V_\omega(x,y) = \omega_1\begin{bmatrix} \frac{xy}{f} \\ f + \frac{y^2}{f} \end{bmatrix} + \omega_2\begin{bmatrix} -f - \frac{x^2}{f} \\ \frac{-xy}{f} \end{bmatrix} + \omega_3\begin{bmatrix} y \\ -x \end{bmatrix}. \tag{3.5}$$

The flow due to the translational components has a radial structure, expanding or contracting about a *focus of expansion* at location (τ,η) and with a magnitude modulated by the distance from the focus of expansion, the component of translation in the viewing direction (v_3), and the inverse depth to the point imaged at each pixel, $\rho(x,y) = 1/Z(x,y)$. In the case $v_3 = 0$, the situation is nearly the same, except that V_v is now a parallel vector field:

$$V_v(x,y) = -f\rho(x,y)\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \qquad (v_3 = 0), \tag{3.6}$$

in the direction of the translational velocity, modulated by inverse depths as before. In both cases, the flow due to the rotational components is the linear combination of three fixed flow fields, weighted by the angular (rotational) velocity components.

The rotational part, namely V_ω, is linear (see Eqn. (3.5)) in the rotational parameters $\omega = (\omega_1, \omega_2, \omega_3)$. On the other hand, the translational part $V_v(x, y)$ is *bilinear*, i.e., linear in $\rho(x, y)$ for fixed translation T and linear in T for fixed inverse depths $\rho(x, y)$. In essence, the translational parameters appear in product form with the inverse depth values (note the forms $\frac{v_1}{Z}$, $\frac{v_2}{Z}$ and $\frac{v_3}{Z}$ appearing in Eqn. (3.2)). We can multiply both the translation vector T and the depth function $Z(x, y)$ by the same factor, say a constant k, and the equations (3.2) will still hold. Thus, it is not possible to recover the absolute values of the translational parameters and the absolute inverse depths. They can be estimated only up to a scale factor; only *relative depths* can be computed from a monocular image sequence, and only two translational parameters can be recovered. There is a need to choose a unit of measure. A good choice for the unit of distance measure is the translation vector; the depth value corresponding to a feature point in units of the translation vector indicates how long it will take for the camera to hit that feature point if moving towards it. For such a choice, the translation vector is necessarily of unit length (i.e., with its tip on a unit sphere). Heeger and Jepson [20] make use of this normalization and search for the translation vector by tessellating the surface of a unit sphere.

Another possible normalization is to set one of the motion parameters to be unity. For instance, if we choose v_3 to be unity (for $v_3 \neq 0$), the other two are measured in terms of v_3. The focus of expansion (FOE) is precisely this. Recall from Eqn. 3.4 that the FOE is defined by ($\tau = \frac{fv_1}{v_3}$, $\eta = \frac{fv_2}{v_3}$), for the case $v_3 \neq 0$, and ($\tau = fv_1$, $\eta = fv_2$), for $v_3 = 0$. We will adopt this choice and in a later chapter present methods to compute the FOE.

If the structure of the scene (and hence $\rho(x, y)$) is known, the optical flow equations are linear in all the motion parameters and can be easily solved, given a sufficient number of flow vectors. However, it is seldom the case that $\rho(x, y)$ is known except in certain controlled situations. Alternatively, if the translation T is known, then one can solve for the rotational parameters ω and the scene structure because the equations will now be linear in the unknowns. In a general situation where none of these quantities is known, it is possible to consider a collection of flow vectors resulting in a system of nonlinear equations and solve for the unknowns. Note that we obtain two equations per flow vector. The unknowns are the five motion parameters and one inverse depth per flow vector position. Thus, given the flow vectors at five or more points, it is possible to solve for all the unknowns. However, solving the set of nonlinear equations is non-trivial and the performance of procedures to do this rely on a good initial guess. Some methods along these lines are presented by Horn [29].

The final comment here concerns the appropriateness of using measured optical flow in conjunction with the model in Eqns. (3.2). The optical flow that is induced on the imaging surface is treated as a vector field that is a projection of the three-dimensional velocity of the feature in motion. The optical flow does not necessarily always correspond to such a projection. One example where such a correspondence does not hold is that of a rotating, featureless sphere. Optically, no motion is seen even though there is physical movement. A formal treatment of the distinction between the projection and the optical flow was done by Verri and Poggio [74]. They use the term "motion field" to denote the projection of the three-dimensional velocity vectors, and point out that the motion field and the optical flow are same only under certain conditions such as high contrast. However, note that only the optical flow can be computed from the intensity images whereas the motion field is the one that is easier to model. Also, modeling the optical flow requires assumptions about illumination and surface reflectance properties; it is hard to choose the right assumptions. Therefore, it is customary to assume that the motion field and the optical flow are the same, as we do here.

In the approaches presented in this thesis, we will attempt to cancel out the contribution from either the translation or the rotation, thus enabling us to estimate the other. The *flow circulation algorithm* presented in the next chapter eliminates the translational part of the flow field in order to estimate the rotational parameters. The *FOE search algorithms* presented in a later chapter cancel the rotational part to estimate the focus of expansion. The focus of expansion is directly related to the translational parameters.

Chapter 4

Rotational parameter estimation

4.1 Introduction

An object in motion can have both translational and rotational motion. For many applications, it is necessary to determine the translational and rotational velocity. For instance, the sensor on a spinning satellite may need to determine the angular velocity to adjust the motion of the satellite. The source of a sensor's rotational velocity may be due to an intentional movement, or due to vibrations. A helicopter in a turning flight has a rotational velocity that is nonzero. A camera mounted on a vehicle that is moving on a rough terrain could experience vibrations that produce an effect as though the camera is rotating about an axis that is changing rapidly. In either case, it is important to determine the instantaneous rotational parameters. In the case of the helicopter, the rotational parameters provide valuable information about the motion itself while in the latter case, the estimated rotational parameters will be useful to eliminate the "jitter" produced by the vibrations. In general, it is well known that estimation of the rotational parameters helps to eliminate the rotational component of the flow field; this leaves behind the translational component from which the translational parameters and the depth map can be computed by simple algorithms. While this observation is theoretically sound, errors introduced in the estimation of one set of parameters compound the errors that appear in the second stage of processing, namely in estimating the other set of parameters.

Presented in this thesis are two independent algorithms, one to estimate the rotational parameters and the other to estimate the translational parameters, providing independent pathways to estimate the different parameters, thus eliminating the compounding of errors. We begin by first presenting an algorithm to determine the rotational parameters.

We should note here that the rotational parameters depend on the choice of the origin. If the location of the motive power for the physical rotation is known, one can choose that location to be the origin of the model. In such a case, pure rotation (i.e., no translation) will be estimated as pure rotation with respect to an axis passing through the chosen origin. However, if the mechanism for the source of rotation is unknown, the natural choice for the origin is the focal point of the sensor.

In such a case, if the rotation axis does not pass through the origin, the rotation will be seen as a combination of a rotation with respect to an axis through the chosen origin and a translation. Note that once the quantities are estimated with respect to one choice of the origin, they can always be transformed for another choice; the judgement of egomotion is thus unaffected by the choice.

The angular velocity ω of the sensor induces an image flow field that depends only on the physical location of the image points and not on their depths. It is the flow field due to the translation that depends on the depth to scene points. Looking at the equations of optical flow,

$$\begin{array}{rcl} u(x,y) & = & \frac{1}{Z(x,y)}\left[-fv_1 + xv_3\right] + \omega_1\left[\frac{xy}{f}\right] - \omega_2\left[f + \frac{x^2}{f}\right] + \omega_3 y, \\ v(x,y) & = & \frac{1}{Z(x,y)}\left[-fv_2 + yv_3\right] + \omega_1\left[f + \frac{y^2}{f}\right] - \omega_2\left[\frac{xy}{f}\right] - \omega_3 x, \end{array} \tag{4.1}$$

we again note that the optical flow field is a vector sum of two flow fields, one arising due to the translation and the other arising due to the rotation:

$$V(x,y) = V^v(x,y) + V^\omega(x,y).$$

Also, note the linearity of the flow field in ω. The method described here attempts to cancel the contribution from translation and the depth values, while still retaining the linearity in ω. We show here that under certain assumptions, this is exactly possible, and under other conditions, it is achieved in a global approximate sense. We provide the background mathematics, the technical derivation and a description of the algorithm which will be called the *flow circulation algorithm.* An analysis of the effect of violations of the assumptions is contained in Chapter 6. Experimental results are presented in Chapters 7 and 8.

4.2 Curl and circulation values

The main observation behind the algorithm is that the curl of the flow vector field produces two terms: A term that is linear in the rotational parameters, and a term that is dependent on the translational parameters as well as the depth gradient. Under suitable conditions, we argue that the latter contribution can be ignored, yielding an approximate linear method to determine the rotational parameters.

We begin by reviewing the curl of a vector function and a related theorem which will be useful to us. The curl of a two-dimensional vector field $\mathbf{V}$ (with $\mathbf{u}$ and $\mathbf{v}$ as the x and y components), also denoted by $\nabla \times \mathbf{V}$, is defined as

$$\nabla \times \mathbf{V} = \frac{\partial \mathbf{v}}{\partial \mathbf{x}} - \frac{\partial \mathbf{u}}{\partial \mathbf{y}}.$$

To be precise, this is the pointwise magnitude of a vector field in a direction orthogonal to the two-dimensional plane containing the vector field $\mathbf{V}$. Since the whole (curl) field is pointed in the same direction, we will use the curl as though it is a scalar function. If the vector field $\mathbf{V}$ lies on a spherical retina instead of a planar surface, then the curl values will point in a direction orthogonal to the spherical surface. Here we are concerned only with the planar imaging surface.

An important property of the curl of a vector field is provided by Stoke's theorem [26]:

$$\iint_D \nabla\times V dx\, dy = \oint_{\partial D} V \cdot d\mathbf{s}, \tag{4.2}$$

where the right hand side, which we will call the *circulation* of the flow about the circuit ∂D, denotes the contour integral of the vector field about the boundary of D. The theorem is quite general, requiring only certain differentiability conditions on V and the boundary of the domain D. Note that the contour integral is independent of the parameterization of the contour. For example, when D is a disk of radius r about an image point (x_0, y_0), the circulation may be calculated from

$$\oint_{\partial D} V \cdot d\mathbf{s} = \int_0^{2\pi} V(x_0 + r\cos\theta, y_0 + r\sin\theta)\cdot(-r\sin\theta, r\cos\theta)d\theta.$$

Mitiche, Grisell and Aggarwal [47] provide a discrete version of Stoke's theorem, and discuss its use for smoothing flow field data given time-varying intensity image data; we need only the continuous version, for smoothing *the curl* of the flow field information.

Interestingly, by using Stoke's theorem, we will not lose any precision, nor do we require any approximations, in order to take advantage of the smoothing. Although the circulation feature values are obtained more stably than would be curl (vorticity) values of the flow field, they are just as useful to the flow circulation algorithm.

4.3 Rotation from curl

Applying the curl to Eqn. (4.1),

$$\nabla\times V(x,y) = v_3 \cdot \left[\frac{\partial\rho}{\partial x}\cdot(y-\eta) - \frac{\partial\rho}{\partial y}\cdot(x-\tau)\right] - \frac{1}{f}\left[x\omega_1 + y\omega_2 + 2f\omega_3\right]. \tag{4.3}$$

We want to show that the first term on the right hand side of Eqn. (4.3) can be neglected, and thus the curl values lie on a linear surface defined by the function $g(x,y) = -(x\omega_1 + y\omega_2 + 2f\omega_3)/f$. We begin by computing the curl that results at a point where an analytic surface element is imaged. Suppose that (x_0, y_0) is the image of an analytic surface point $P(X_0, Y_0, Z_0)$ and the analytic surface is described (implicitly) by the equation

$$n_1(X-X_0) + n_2(Y-Y_0) + n_3(Z-Z_0) + \sum_{|\alpha|\geq 2}(X-X_0)^{\alpha_1}(Y-Y_0)^{\alpha_2}(Z-Z_0)^{\alpha_3} = 0.$$

In this, α is a multi-index with integer components α_i, and $|\alpha|$ is the order of the multi-index, i.e., the sum of the components. Any analytic surface can be locally described this way. The tangent

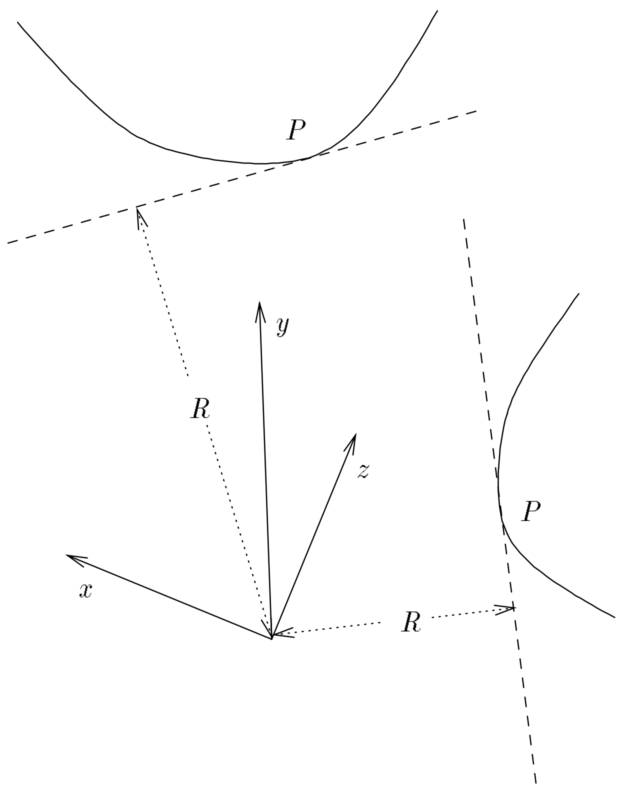


Figure 4.1: Two analytic surfaces and their associated R values

plane to this surface has the normal $\mathbf{n} = (n_1, n_2, n_3)$, which we may assume is a unit normal. We set

$$R = n_1 X_0 + n_2 Y_0 + n_3 Z_0,$$

which is the distance, at the point of closest approach, between the tangent plane to the surface and the focal point of the imaging system (see Fig. 4.1).

Using the projective geometry equations, and the definition that $\rho(x, y) = 1/Z(x, y)$, one can easily show that

$$\frac{\partial \rho}{\partial x}(x_0, y_0) = \frac{n_1}{R}, \ \frac{\partial \rho}{\partial y}(x_0, y_0) = \frac{n_2}{R}.$$

After substitution into Eqn. (4.3), we obtain

$$\nabla \times V(x_0, y_0) = v_3 \left[\frac{n_1}{R}(y_0 - \eta) - \frac{n_2}{R}(x_0 - \tau) \right] - \frac{1}{f} \left[x_0 \omega_1 + y_0 \omega_2 + 2 f \omega_3 \right]. \qquad (4.4)$$

Regrouping terms and using the definitions of τ and η, we get

$$\begin{aligned}\nabla \times V(x_0, y_0) \ &= \\ -\frac{1}{f}\left[x_0\left(\omega_1 + \frac{n_2 v_3}{R}\right)\right. &+ y_0\left(\omega_2 + \frac{n_1 v_3}{R}\right) + \left.\left(2f\omega_3 + \frac{n_1 v_2 - n_2 v_1}{R}\right)\right]. \end{aligned} \quad (4.5)$$

If we now allow (x_0, y_0) to vary and denote the point by (x, y), regarding (n_1, n_2, n_3) as a function of (x, y), we conclude from Eqn. (4.4) and the equivalent Eqn. (4.5) that the curl

$$\nabla \times V(x, y) = \frac{-1}{f}\left(x\omega_1 + y\omega_2 + 2f\omega_3\right), \quad (4.6)$$

whenever any of the following conditions holds true:

- If $v_1 = v_2 = v_3 = 0$, then Eqn. (4.6) holds everywhere. Note that this is a particularly trivial case since the original equations for $u(x, y)$ and $v(x, y)$ (see Eqn. (4.1)) will themselves be linear in the rotational parameters.

- At any point (x, y) such that $n_1 = n_2 = 0$. The condition that $n_1 = n_2 = 0$ is equivalent to saying that the tangent plane to the surface imaged at (x, y) lies normal to the viewing direction. In particular, the condition holds true for any frontal planar surface.

- At any point where the vector to the focus of expansion $(\tau - x, \eta - y)$ is proportional to the tilt of the tangent plane to the surface imaged at (x, y) (the tilt is defined as (n_1, n_2)), then clearly the first term on the right hand side of Eqn. (4.4) vanishes, and Eqn. (4.6) holds.

Eqn. (4.6) will hold approximately if the distance R from the tangent plane to the focal point is large relative to the translational velocity $||\mathbf{v}||$ (note that the components of $\mathbf{n}$ satisfy $|n_i| \leq 1$, since $\mathbf{n}$ is a unit normal), or if the rotational components of velocity ω are quite large compared to the translational components $\mathbf{v}$, and assuming that the R value is bounded from below.

It is interesting to note that the R value becomes small whenever a surface lies nearby, and also whenever a surface is oriented so that the tangent plane passes close to the focal point. Even when the errors are such that Eqn. (4.6) does not hold approximately pointwise, it is still possible that Eqn. (4.6) holds in a globally approximate sense. This will happen, for example, if the surface tilts n_1 and n_2 are random and well distributed. Because the algorithm is global, the global approximation condition suffices.

Also, even though we have used an analytic surface to point out the conditions under which the algorithm is useful, note that the algorithm is as useful for general surfaces as long as Eqn. (4.6) is approximately true in a global sense.

The algorithm will use feature data to fit a linear function of the form $g(x, y) = ax + by + c$ to the data, and then ω_1, ω_2, and ω_3 may be determined directly from a, b, c respectively. However, using Stoke's theorem, we show that we do not have to rely on samples of the curl of the flow field: we may instead use circulation values.

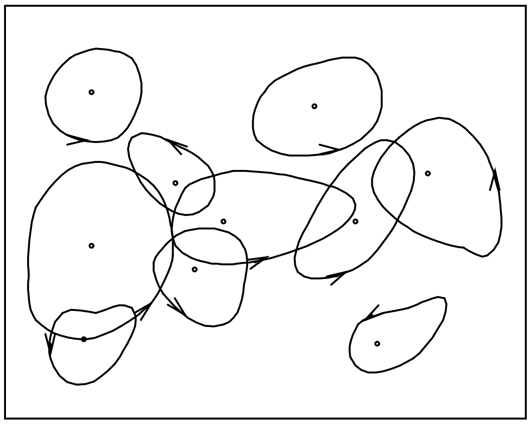


Figure 4.2: A collection of cycles and their centroids, where data for the flow circulation algorithm is collected. The algorithm will simply fit a linear function to the circulation values given at each centroid.

Consider a region of the image domain D (not necessarily circular), such that at most of the points (x, y) in D, one of the above conditions holds true. We do not require that D be a small region, nor do we require that the same condition hold true over all of D, nor are we concerned about substantial violations *as long as the average of the violations is small.* We will average the values of the curl over D:

$$\begin{aligned} \frac{1}{|D|}\iint_D \nabla \times V dx\, dy &\approx \frac{1}{|D|}\iint_D -\frac{1}{f}\left[x\omega_1 + y\omega_2 + 2f\omega_3\right] dx\, dy \\ &= -\frac{1}{f}\left[x_0\omega_1 + y_0\omega_2 + 2f\omega_3\right], \end{aligned} \tag{4.7}$$

where (x_0, y_0) is the centroid in the image plane of the region D. Here, $|D|$ refers to the area of D in the image domain. Applying Stoke's theorem to the left hand side of (4.7), we see that the circulation value of V about the boundary of D satisfies

$$\frac{1}{|D|}\oint_{\partial D} V \cdot dS \approx -\frac{1}{f}\left[x_0\omega_1 + y_0\omega_2 + 2f\omega_3\right]. \tag{4.8}$$

4.4 The flow circulation algorithm

We now state the algorithm. Suppose we have a collection of image domains D_i, $i = 1, \ldots, N$. Suppose that we have normalized the circulation values measured for each such domain:

$$\gamma_i = \frac{1}{|D_i|}\oint_{\partial D_i} V \cdot d\mathbf{s}.$$

Let us suppose that the centroid of each region D_i is known, and is denoted by (x_i, y_i) (see Fig. 4.2). From Eqn. (4.7), we know that the data (x_i, y_i, γ_i), for $i = 1, \ldots, N$, satisfies, approximately,

$$\gamma_i = ax_i + by_i + c,$$

where $a = -\omega_1/f$, $b = -\omega_2/f$, and $c = -2\omega_3$. We can determine the coefficients a, b, and c by fitting a linear function to the available data. Outliers, of course, can be discarded by standard regression methods, so that violations of the conditions are not consequential. Clearly, the rotational parameters are determined from a, b, and c.

If the collection of centroids of regions (x_i, y_i) is sufficiently dense and symmetrically distributed about the image plane origin, then the determination of the rotational parameters from the circulation data becomes especially simple, since then the parameters are coefficients of mutually orthogonal functions, Specifically, a will be proportional to the average value of $x_i \cdot \gamma_i$, and b will be proportional to the average value of $y_i \cdot \gamma_i$, and c will be equal to the average value of γ_i. Then $\omega_1 = -fa$, $\omega_2 = -fb$, and $\omega_3 = -c/2$. Once again, outliers may be discarded from the averages so as to improve the quality of the estimates.

Thus, the algorithm simply involves computing contour integrals (which might be directly provided by feature detectors) and fitting a linear surface to this data.

Chapter 5

Translational parameter estimation

5.1 Introduction

The translation of a sensor, $v = (v_1, v_2, v_3)$, is an important quantity to be determined in visual motion analysis. For instance, in navigation, it is important to determine the direction of movement in order to avoid obstacles in the path. In turn, knowledge of the translation helps to determine the scene structure and is useful in detecting obstacles. From monocular imagery, the translation can be estimated only up to a scale factor, due to the combined effect of translation and depth values on the flow field. One way to see this is to note that the same sequence can be produced by a nearby point moving slowly or a distant point moving (proportionally) fast. This scale factor can be absorbed into the unknown quantities in several ways. In particular, we will be interested in determining the focus of expansion (FOE) which is nothing but the projection of the translational velocity vector on the image plane.

The FOE is particularly easy to compute in the case where there is pure translation (i.e., zero angular velocity). In such a case, the optical flow field is radially expanding or contracting, and the FOE is found to be the point where the flow field vectors intersect. The simple-minded procedure of finding the FOE as the intersection point of the vectors does not work in practice for several reasons. First, the optical flow field cannot be determined exactly, due to the aperture problem. Secondly, noise adversely affects the flow field computation; and thirdly, pure translation is seldom "pure" — there is always some rotation due to vibrations resulting in a turning of the camera and other causes of rotation. Under such conditions, there is a need to somehow eliminate the contribution from the rotation because the flow field is no longer of the radially expanding type.

Determination of the translation enables one to solve for the rotation and the structure of the scene. Note that if the translation is known, then the optical flow equations become linear, and so a collection of flow vectors provides enough information to solve for the other unknown quantities using a linear algorithm. However, computation of the translation is made difficult by the fact that in the translational part of the optical flow in the equations

$$\begin{array}{rcl} u(x,y) & = & \frac{1}{Z(x,y)}\left[-fv_1 + xv_3\right] + \omega_1\left[\frac{xy}{f}\right] - \omega_2\left[f + \frac{x^2}{f}\right] + \omega_3 y, \\ v(x,y) & = & \frac{1}{Z(x,y)}\left[-fv_2 + yv_3\right] + \omega_1\left[f + \frac{y^2}{f}\right] - \omega_2\left[\frac{xy}{f}\right] - \omega_3 x, \end{array} \tag{5.1}$$

the translational parameters appear coupled with the inverse depth values. Thus, the equations in (5.1) are bilinear as functions of $\mathbf{T}$ and $\rho(x,y)$. In this chapter, we present a projection that achieves two goals: (1) eliminating the contribution due to the rotation, and (2) providing a function that has a minimum at the FOE. We present three methods that make use of this projection and point out the features of these methods. Analysis of the methods and experimental results are provided in separate chapters.

5.2 The circular-component function

We begin by defining a scalar function which is a particular projection of the optical flow field vectors. The projection is parameterized by the center of concentric circles drawn in the image plane. This projection, called the circular-component velocity function at a point (x,y) is obtained from the vector dot product of the velocity flow field and the circular vector field that is depicted in Fig. 5.1, centered on (x_0,y_0), and the magnitude of the vector field increases with the radius.

The formal description is as follows: For each point (x_0,y_0), we consider the circular component flow field about (x_0,y_0) defined by:

$$U_{(x_0,y_0)} = V(x,y)\cdot(-y+y_0, x-x_0). \tag{5.2}$$

The vector field $(-y+y_0, x-x_0)$ is shown in Fig. (5.1). Since $V = V_{\mathbf{v}} + V_\omega$, we further define

$$\begin{array}{rcl} U^v_{(x_0,y_0)}(x,y) & = & V_v(x,y)\cdot(-y+y_0, x-x_0), \\ U^\omega_{(x_0,y_0)}(x,y) & = & V_\omega(x,y)\cdot(-y+y_0, x-x_0), \end{array}$$

so that

$$U_{(x_0,y_0)}(x,y) = U^v_{(x_0,y_0)}(x,y) + U^\omega_{(x_0,y_0)}(x,y).$$

We calculate the second term first:

$$\begin{array}{rcl} U^\omega_{(x_0,y_0)} & = & \omega_1\left[-\frac{x_0}{f}y^2 + \frac{y_0}{f}xy + fx - fx_0\right] \\ & & +\omega_2\left[-\frac{y_0}{f}x^2 + \frac{x_0}{f}xy + fy - fy_0\right] \\ & & +\omega_3\left[-y^2 - x^2 + x_0x + y_0y\right]. \end{array} \tag{5.3}$$

The important thing to note is that for fixed ω and (x_0,y_0), this function is a quadratic polynomial in x and y. As for the first term, we have

$$U^v_{(x_0,y_0)}(x,y) = v_3\rho(x,y)\cdot\left[(y_0-\eta)x + (-x_0+\tau)y + \eta x_0 - \tau y_0\right]. \tag{5.4}$$

$U^v_{(x_0,y_0)}(x,y)$ is not, in general, a quadratic polynomial since the inverse depth function is not restricted. However, at the focus of expansion, when $(x_0, y_0) = (\tau, \eta)$,

$$U^v_{(\tau,\eta)}(x,y) = v_3\rho(x,y) \cdot \begin{bmatrix} x-\tau \\ y-\eta \end{bmatrix} \cdot (-y+\eta, x-\tau) = 0 \tag{5.5}$$

so that $U_{(x_0,y_0)} = U^\omega_{(x_0,y_0)}$ for $(x_0, y_0) = (\tau, \eta)$. Eqn. (5.5) is merely a result of the radial structure of the translational component of the flow field. Thus, $U_{(x_0,y_0)}$ will be a quadratic polynomial when (x_0, y_0) is at the focus of expansion (τ, η). At other points, $U_{(x_0,y_0)}$ is composed of the sum of a quadratic polynomial of the form (5.3) and the function $U^v_{(x_0,y_0)}$, which is in general not a quadratic polynomial.

The definition of the circular-component flow field function can be extended to the case where the candidate focus of expansion is a point at infinity (corresponding to forward velocity v_3 equal to zero). In this case, the candidate focus of expansion is a unit direction (v_1, v_2), and the constant vector field $(-v_2, v_1)$ may replace the circular field $(-y + y_0, x - x_0)$ in the definition of the circular component function, yielding a function $U^v_{(v_1,v_2,0)}$.

5.3 Finding the FOE

The algorithm is thus the following: every candidate (x_0, y_0) for the focus of expansion computes the function $U_{(x_0,y_0)}$ and then determines whether the resulting function is a quadratic polynomial of the form (5.3). Candidate foci of expansion at infinity can be included by considering the circular component flow function $U^v_{(v_1,v_2,0)}$ for unit vectors (v_1, v_2). That candidate for which the resulting function is a quadratic polynomial is declared to be the correct focus of expansion. This can be done in several ways. We present three methods here. The first two methods use ways to cancel any quadratic polynomial while the third one eliminates quadratic polynomials of the specific form.

The center-surround kernel method uses a convolution process to remove the contribution from the rotational velocity. Observe that $U^\omega_{(x_0,y_0)}$ (see Eqn. (5.3)), the rotational part of the circular-component function, is in a three-dimensional subspace of all quadratic polynomial functions. This subspace is spanned by the following functions:

$$\begin{aligned} \phi_1(x,y) &= -\frac{x_0}{f}y^2 + \frac{y_0}{f}xy + fx - fx_0, \\ \phi_2(x,y) &= -\frac{y_0}{f}x^2 + \frac{x_0}{f}xy + fy - fy_0, \\ \phi_3(x,y) &= -y^2 - x^2 + x_0x + y_0y. \end{aligned}$$

The space of all quadratic polynomial functions is, in turn, a subspace of the space of all polynomials, which in turn is a subspace of all functions. The three-dimensional subspace of $U^\omega_{(x_0,y_0)}$ is dependent on the choice of (x_0, y_0). Thus, in order to eliminate the rotational part, we could remove either the projection of the circular-component function on the space of quadratic polynomials in general (yielding the quadratic polynomial projection method) or the projection on the three-dimensional subspace dependent on (x_0, y_0) (the subspace projection method).

5.3.1 The center-surround kernel method

We begin by presenting a method that uses convolution; a convolution procedure is desirable because of its simplicity and its well-studied properties. We note that the Laplacian of (5.3) is a constant:

$$\begin{aligned} \Delta U^{\omega}_{(x_0,y_0)}(x,y) &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) U^{\omega}_{(x_0,y_0)} \\ &= \frac{-2x_0}{f}\omega_1 - \frac{2y_0}{f}\omega_2 - 4\omega_3. \end{aligned}$$

Thus any derivative of the Laplacian of $U_{(x_0,y_0)}$ will give zero when $(x_0, y_0) = (\tau, \eta)$. Rather than taking derivatives, we advocate filtering $U_{(x_0,y_0)}$ by a convolution kernel. We suggest three possibilities. The first suggested kernel is $(\partial/\partial r)\Delta G_\sigma$, where G_σ is a Gaussian kernel with standard deviation σ, and r is a radial variable (we may regard G_σ as defined in polar coordinates). For fixed σ, in (x, y) coordinates, this kernel is proportional to

$$K(x,y) = (x^2+y^2)^{1/2}\left(4 - \frac{x^2+y^2}{\sigma^2}\right) \cdot e^{-(x^2+y^2)/2\sigma^2}. \tag{5.6}$$

A cross section of this kernel is shown in Fig. 5.2. The 2-D kernel is a rotationally symmetric version of the displayed function. The 2-D kernel is convolved with the circular-component flow field function, and yields a zero resulting function when the candidate position is located at the focus of expansion.

Another possibility is $(\partial/\partial x)\Delta G_\sigma$. This kernel will not be circularly symmetric, but can be implemented by differencing two horizontally displaced center-surround receptive fields, as might be found in a stereo imaging system. Convolution by this kernel is reminiscent of the receptive fields proposed by Nakayama and Loomis [51]. For this kernel, we have the (proportional) formula

$$K(x,y) = x\left(4 - \frac{r^2}{\sigma^2}\right) \cdot e^{-r^2/2\sigma^2} \tag{5.7}$$

A three-dimensional plot of this kernel is shown in Fig. (5.3). A third possibility is to filter $U_{(x_0,y_0)}$ by ΔG_σ, without any derivatives. This kernel is given by

$$K(x,y) = \left(2 - \frac{r^2}{\sigma^2}\right) \cdot e^{-r^2/2\sigma^2}, \tag{5.8}$$

which is the well-known Mexican-hat function. In this case, we search for a location (x_0, y_0) where the result of the filtering operation is a constant function. By filtering, we mean that a convolution

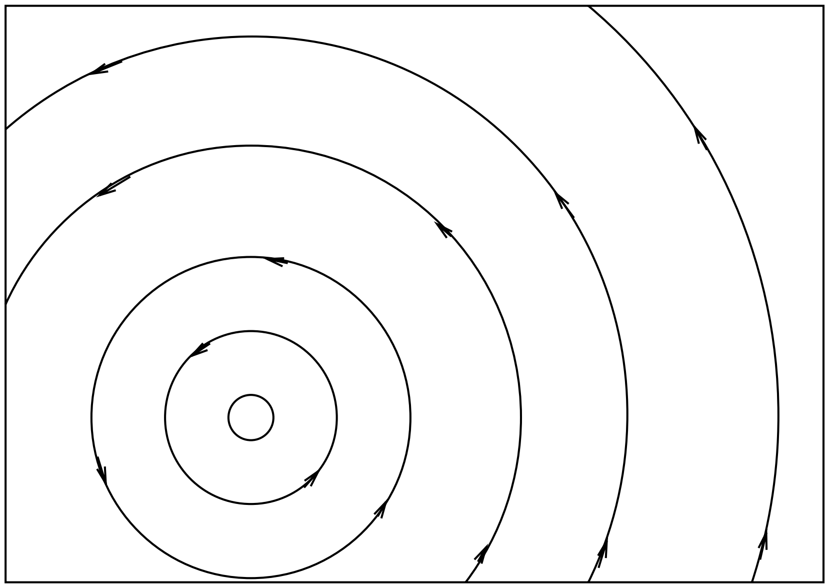


Figure 5.1: The circular vector field used in the computation of the circular-component velocity function

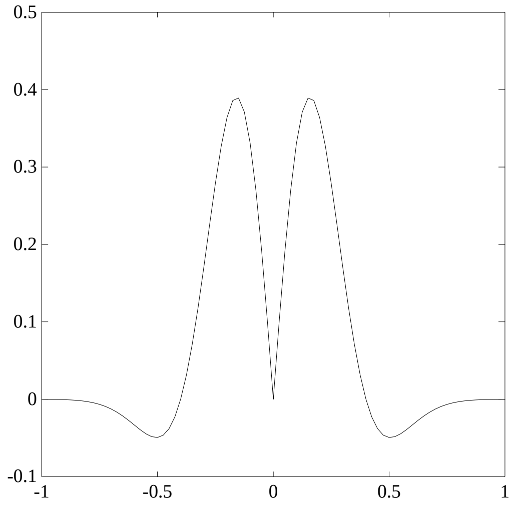


Figure 5.2: A slice of the kernel $(\partial/\partial r)\Delta G_\sigma$ used in the center-surround kernel method.

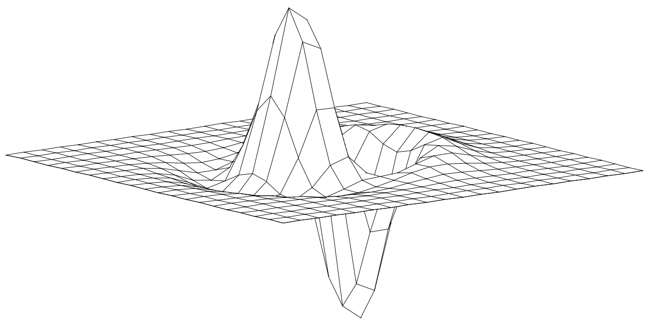


Figure 5.3: A plot of the kernel in Eqn. (5.7) used in the center-surround kernel method.

is desired:

$$\Phi_{(x_0,y_0)}(x,y) = K * U_{(x_0,y_0)}(x,y) = \iint K(x',y')U_{(x_0,y_0)}(x-x',y-y')dx'dy'. \tag{5.9}$$

We are assured that when $(x_0, y_0) = (\tau, \eta)$, $\Phi_{(x_0,y_0)}(x,y)$ is identically zero (the zero function) for the kernels (5.6) and (5.7) discussed above, and is constant for the kernel (5.8). A reasonable criterion would be to search for (x_0, y_0) such that

$$E(x_0,y_0) = \iint |\Phi_{(x_0,y_0)}(x,y)|^2 dxdy \tag{5.10}$$

is zero, in the case of kernels (5.6) and (5.7), and where $E(x_0, y_0) = \mathrm{Var}(\Phi_{(x_0,y_0)})$, the variance of $\Phi_{(x_0,y_0)}$, is zero in the case of kernel (5.8).

For the kernels in (5.6) and (5.7), the error surface $E(x_0, y_0)$ can be rewritten as a quadratic functional of the function $U_{(x_0,y_0)}(x,y)$. The quadratic functional has form

$$E(x_0,y_0) = \iiiint R(x,y,x',y') \cdot U_{(x_0,y_0)}(x,y)U_{(x_0,y_0)}(x',y')dxdydx'dy'. \tag{5.11}$$

Formulas are derived for R in the next section.

Although this provides a particularly attractive test for the focus of expansion, the problem with the center-surround kernel is that the kernel will yield a zero convolution against many functions other than quadratic polynomials. For example, if ρ is such that $U_{(x_0,y_0)}$ happens to be a harmonic function for some (x_0, y_0), then $E(x_0, y_0)$ will be zero. While it is not likely that ρ will yield a harmonic $U_{(x_0,y_0)}$ for general surface shapes, the test using the center-surround kernel is not specific to the form of the quadratic polynomial that arises at the focus of expansion, and thus can be considerably improved.

5.3.2 The quadratic functionals

In this section, we derive the formulas for the quadratic functionals. First, we want to show that, given two functions $K(\mathbf{x})$ and $U(\mathbf{x})$ of the multivariate $\mathbf{x}$, if $\Phi(\mathbf{x}) = K * U(\mathbf{x})$, then

$$\|\Phi\|^2 = \int (\check{K} * K * U(\mathbf{x})) \cdot U(\mathbf{x}) d\mathbf{x}, \tag{5.12}$$

where $\check{K}(\mathbf{x}) = K(-\mathbf{x})$. In order to prove this, first note that

$$< A * f, g > = < f, \check{A} * g >,$$

for functions A, f and g, and where $< f, g >$ denotes the inner product of f and g. Using this, we see that

$$\begin{aligned} \| \Phi \|^2 &= \|K * U\|^2 \\ &= < K * U, K * U > \\ &= < U, \check{K} * K * U > . \end{aligned}$$

Now, consider the kernel $(\partial/\partial r)\Delta G_\sigma$ (Eqn. (5.6)). This is a symmetric kernel ($\check{K}(x,y) = K(x,y)$). So, from Eqn. (5.12), we have

$$R(x, y, x', y') = S(x - x', y - y'),$$

where

$$\begin{aligned} S(x,y) &= K * K \\ &= (\partial^2/\partial r^2)\Delta^2 G_{\sqrt{2}\sigma}, \end{aligned}$$

for $r = \sqrt{x^2 + y^2}$. A plot of the $K * K$ kernel is shown in Fig. (5.4). For the asymmetric kernel K in Eqn. (5.7),

$$\begin{aligned} S(x,y) &= \check{K} * K \\ &= (\partial^2/\partial x^2)\Delta^2 G_{\sqrt{2}\sigma}. \end{aligned}$$

A plot of $\check{K} * K$ is shown in Fig. (5.5).

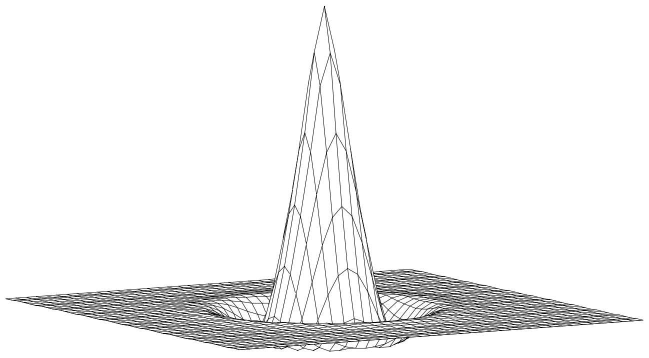


Figure 5.4: The autoconvolution kernel $K * K$, for the K in Eqn. (5.6), that may be used as the quadratic functional kernel in the center-surround kernel method to detect whether $U_{(x_0,y_0)}$ is a quadratic polynomial, and thus if (x_0, y_0) is the focus of expansion.

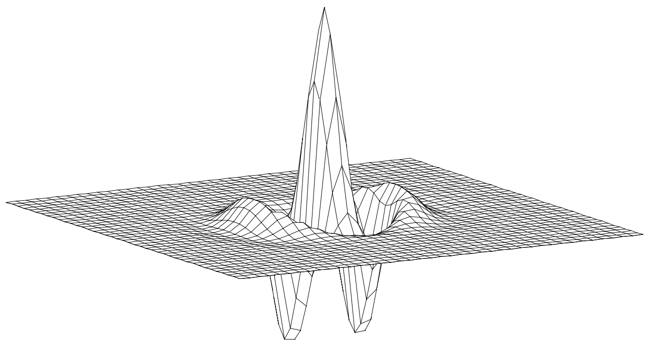


Figure 5.5: The $\check{K} * K$ that may be used as the quadratic functional kernel in the center-surround kernel method, for the asymmetric kernel K in Eqn. (5.7).

5.3.3 The quadratic polynomial projection method

We now present a method in which we eliminate the contribution from rotation by using a projection onto the space of quadratic polynomials. The projection can be computed if an orthonormal basis can be found for the space of quadratic polynomials. We can find such a basis by starting with a set of functions that span this space. One such set is: $\{1, x, y, xy, x^2, y^2\}$. An orthonormal basis can be derived from this set using an *orthogonalizing* method such as the *Gram-Schmidt process.* We need an inner product to be defined over the space in order to accomplish the orthogonalization. Defining the *Hermite inner product* to be

$$< f, g >= \iint f(x,y)g(x,y)e^{(-x^2-y^2)/2}dxdy, \tag{5.13}$$

the orthogonalizing process leads to the following polynomials:

$$\begin{aligned}\phi_1(x,y) &= \frac{1}{\sqrt{2\pi}},\ \phi_2(x,y) = \frac{x}{\sqrt{2\pi}},\ \phi_3(x,y) = \frac{y}{\sqrt{2\pi}},\\ \phi_4(x,y) &= \frac{xy}{\sqrt{2\pi}},\ \phi_5(x,y) = \frac{x^2-1}{2\sqrt{\pi}},\ \phi_6(x,y) = \frac{y^2-1}{2\sqrt{\pi}}.\end{aligned}$$

These are the Hermite polynomials in two variables up to degree two. The Hermite polynomials have unit norm and are mutually orthogonal with respect to the Hermite inner product. The function $U_{(x_0,y_0)}$ will thus be a quadratic polynomial if and only if the following function vanishes identically:

$$\Psi(x,y) = U_{(x_0,y_0)}(x,y) - \sum_{i=1}^{6} < U_{(x_0,y_0)}, \phi_i > \phi_i(x,y). \tag{5.14}$$

The residual function Ψ is the function left after eliminating from $U_{(x_0,y_0)}$ the projection on to the space of quadratic polynomials. For the quadratic polynomial projection method, we will use the square of the norm of this residual function as our error measure $E(x_0,y_0)$:

$$E(x_0,y_0) = \| \Psi(x,y) \|^2 .$$

The (Hermite) norm, which will be zero if and only if $U_{(x_0,y_0)}$ is a quadratic polynomial, may be written as a quadratic functional of $U_{(x_0,y_0)}$, as derived in the following. Note that we make use of the fact that the functions $\phi_i(x,y)$ are orthonormal. The derivation is done for a multivariate $\mathbf{x}$:

$$\begin{aligned}\| \Psi \|^2 &= \| U \|^2 - \sum_{i=1}^{6} \| < U, \phi_i > \phi_i \|^2\\ &= \| U \|^2 - \sum_{i=1}^{6} (< U, \phi_i >)^2\\ &= \left(\int U(x) e^{-\mathbf{x}^2/2} d\mathbf{x} \right)^2 -\end{aligned}$$

$$\begin{aligned}
&\sum_{i=1}^{6}\left[\int U(\mathbf{x})\phi_i(x)e^{-\mathbf{x}^2/2}d\mathbf{x}\int U(\mathbf{x}')\phi_i(\mathbf{x}')e^{\mathbf{x}'^2/2}d\mathbf{x}'\right] \\
= &\iint \delta(\mathbf{x},\mathbf{x}')U(\mathbf{x})U(\mathbf{x}')e^{-\mathbf{x}^2/2}e^{-\mathbf{x}'^2/2}d\mathbf{x}d\mathbf{x}' \\
&-\sum_{i=1}^{6}\iint U(\mathbf{x})U(\mathbf{x}')\phi_i(\mathbf{x})\phi_i(\mathbf{x}')e^{-\mathbf{x}^2/2}e^{-\mathbf{x}'^2/2}d\mathbf{x}d\mathbf{x}' \\
= &\iint \left(\delta(\mathbf{x},\mathbf{x}')-\sum_{i=1}^{6}\phi_i(\mathbf{x})\phi_i(\mathbf{x}')\right)U(\mathbf{x})U(\mathbf{x}')e^{-\frac{\mathbf{x}^2+\mathbf{x}'^2}{2}}d\mathbf{x}d\mathbf{x}'.
\end{aligned}$$

In this, $\delta(\mathbf{x},\mathbf{x}')$ is a delta distribution whose mass lies entirely on the slice $\mathbf{x}=\mathbf{x}'$. Thus, the norm may be written as a quadratic functional of $U_{(x_0,y_0)}$:

$$\begin{aligned}
&E(x_0,y_0) \;= \\
&\iiiint R(x,y,x',y')\cdot U_{(x_0,y_0)}(x,y)U_{(x_0,y_0)}(x',y')e^{-\frac{x^2+y^2+x'^2+y'^2}{2}}dxdydx'dy'. \qquad (5.15)
\end{aligned}$$

In this case, R is the "quadratic polynomial projection kernel," and will equal

$$R(x,y,x',y')=\delta(x,y,x',y')-\sum_{i=1}^{6}\phi_i(x,y)\phi_i(x',y').$$

We thus see that the test for (x_0,y_0) depends on a quadratic functional applied to $U_{(x_0,y_0)}$, and that the kernel of the quadratic functional in (5.15) has the form

$$\begin{aligned}
R(x,y,x',y') \;=\; &\delta(x,y,x',y')-\frac{1}{2\pi}\left[1+xx'+yy'+xyx'y'\right] \\
&-\frac{1}{4\pi}\left[(x^2-1)(x'^2-1)+(y^2-1)(y'^2-1)\right].
\end{aligned}$$

Note that the quadratic polynomial projection kernel is independent of (x_0,y_0).

The quadratic polynomial projection method has the property that if $U_{(x_0,y_0)}$ is a quadratic polynomial, then (x_0,y_0) will be defined as the focus of expansion. Alas, if $U^v_{(x_0,y_0)}$ happens to be a quadratic polynomial, then $U_{(x_0,y_0)}$ is simply the sum of two quadratic polynomials, and so (x_0,y_0) will be erroneously identified. We might assert that it is highly unlikely that the scene will be such that $U^v_{(x_0,y_0)}$ gives precisely a quadratic polynomial, but unfortunately, if $\rho(x,y)$ happens to be a linear function, then $U^v_{(x_0,y_0)}$ is indeed a quadratic polynomial. It so happens that $\rho(x,y)$ is linear (and thus also harmonic, so the center-surround kernel method will also fail) when the surface $Z(x,y)$ is planar (although the entire scene will have to consist of the single planar surface). We will demonstrate this degenerate case in the next chapter where we analyze the various methods presented in this chapter.

5.3.4 The subspace projection method

We note that $U^{\omega}_{(x_0,y_0)}$ is a quadratic polynomial in (5.3) of a special form; $U^{\omega}_{(x_0,y_0)}$ must lie in a three dimensional subspace of quadratic polynomials spanned by the (redefined) basis functions

$$\begin{aligned}
\phi_1(x,y) &= -\frac{x_0}{f}y^2 + \frac{y_0}{f}xy + fx - fx_0, \\
\phi_2(x,y) &= -\frac{y_0}{f}x^2 + \frac{x_0}{f}xy + fy - fy_0, \\
\phi_3(x,y) &= -y^2 - x^2 + x_0x + y_0y.
\end{aligned}$$

Note that unlike the Hermite basis functions from the previous sections, these basis functions depend on (x_0, y_0).

For the subspace projection method, we define the error amount $E(x_0, y_0)$ to be

$$E(x_0,y_0) = \min_{a_1,a_2,a_3} \parallel U_{(x_0,y_0)} - \sum_{i=1}^{3} a_i\phi_i \parallel^2 .$$

The norm is based on an inner product, and we use the Hermite inner product defined in Eqn. (5.13). This is a standard least squares minimization problem, and the a_i may be found as the solution to the "normal equations"

$$Q \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} < U_{(x_0,y_0)}, \phi_1 > \\ < U_{(x_0,y_0)}, \phi_2 > \\ < U_{(x_0,y_0)}, \phi_3 > \end{bmatrix}, \tag{5.16}$$

where Q is the three by three matrix of inner products, i.e., the $(i,j)^{\text{th}}$ component of Q is $< \phi_i, \phi_j >$. For a solution set a_1, a_2, and a_3, the minimum distance $E(x_0, y_0)$ is given by the quadratic functional formula (5.15). In the following derivation for this quadratic functional formula, for brevity, we make use of the abbreviations

$$\alpha = [\phi_1, \phi_2, \phi_3]\,, \text{ and } \beta = [< U, \phi_1 >, < U, \phi_2 >, < U, \phi_3 >]\,.$$

Also, observe that

$$\alpha\alpha^T = Q, \text{ and } Q^T = Q.$$

We have

$$\begin{aligned}
E(x_0,y_0) &= \min_{a_1,a_2,a_3} \|U_{(x_0,y_0)} - \sum_{i=1}^{3} a_i\phi_i\|^2 \\
&= \parallel U \parallel^2 - \parallel \sum_{i=1}^{3} a_i\phi_i \parallel^2 \\
&= \parallel U \parallel^2 - \left\| [\phi_1,\phi_2,\phi_3]\, Q^{-1} \begin{bmatrix} < U, \phi_1 > \\ < U, \phi_2 > \\ < U, \phi_3 > \end{bmatrix} \right\|^2
\end{aligned}$$

$$
\begin{aligned}
&= \| U \|^2 - (\alpha^T Q^{-1}\beta)^T (\alpha^T Q^{-1}\beta) \\
&= \| U \|^2 - \beta^T (Q^{-1})^T \alpha\alpha^T Q^{-1}\beta \\
&= \| U \|^2 - \beta^T Q^{-1}\beta \\
&= \left(\int U(x) e^{-\mathbf{x}^2/2} d\mathbf{x} \right)^2 - \\
&\quad [< U,\phi_1 >, < U,\phi_2 >, < U,\phi_3 >]\, Q^{-1} \begin{bmatrix} < U,\phi_1 > \\ < U,\phi_2 > \\ < U,\phi_3 > \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
&= \int\!\!\int \delta(\mathbf{x},\mathbf{x}') U(\mathbf{x}) U(\mathbf{x}') e^{-\mathbf{x}^2/2} e^{-\mathbf{x}'^2/2} d\mathbf{x} d\mathbf{x}' - \\
&\int\!\!\int U(\mathbf{x}) \left[\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \phi_3(\mathbf{x})\right] Q^{-1} U(\mathbf{x}') \begin{bmatrix} \phi_1(\mathbf{x}') \\ \phi_2(\mathbf{x}') \\ \phi_3(\mathbf{x}') \end{bmatrix} e^{-\mathbf{x}^2/2} e^{-\mathbf{x}'^2/2} d\mathbf{x} d\mathbf{x}'.
\end{aligned}
$$

Thus, R, the "subspace projection kernel" of formula (5.15) is now redefined as

$$
\begin{aligned}
R_{(x_0,y_0)}(x,y,x',y') &= \delta(x,y,x',y') - \\
&\quad [\phi_1(x,y), \phi_2(x,y), \phi_3(x,y)]\, Q^{-1}_{(x_0,y_0)} \begin{bmatrix} \phi_1(x',y') \\ \phi_2(x',y') \\ \phi_3(x',y') \end{bmatrix},
\end{aligned}
$$

where we have written $Q_{(x_0,y_0)}$ in order to emphasize that the ϕ_i depend on (x_0, y_0). The triple product (the second term on the right hand side) means the same thing as

$$
\sum_{i=1}^{3} \sum_{j=1}^{3} q_{i,j} \phi_i(x,y) \phi_j(x',y'),
$$

where $q_{i,j}$ is the $(i,j)^{\text{th}}$ component of the inverse matrix $Q^{-1}_{(x_0,y_0)}$. Note that Q and thus Q^{-1} can be precomputed as functions of (x_0, y_0): they are matrices of constants that depend on (x_0, y_0). Using the resulting kernel R provides the exact test, and $E(x_0, y_0)$ will be zero if and only if $U_{(x_0,y_0)}$ is a quadratic polynomial of exactly the correct form.

With some calculation, we can show that the matrix of Hermite inner products is given by

$$
Q = \begin{bmatrix} 1 + 6x_0^2 + y_0^2 & 5x_0y_0 & 7x_0 \\ 5x_0y_0 & 1 + x_0^2 + 6y_0 & 7y_0 \\ 7x_0 & 7y_0 & 8 + x_0^2 + y_0^2 \end{bmatrix}.
$$

We discover that an inverse always exists, and that the coefficients are quotients of sixth order polynomials (i.e., rational polynomials) in the two position values. These functions do not have to be solved! They are merely evaluated for each (x_0, y_0) that is considered. The nine values $q_{i,j}$ of the inverse matrix Q^{-1} can be computed for each (x_0, y_0) using the following formulas:

$$\begin{aligned}
q_{1,1} &= (8 + 9x_0^2 + 7x_0^2y_0^2 + x_0^4 + 6y_0^4)/P, \\
q_{2,2} &= (8 + 9y_0^2 + 7x_0^2y_0^2 + 6x_0^4 + y_0^4)/P, \\
q_{3,3} &= (1 + 7(x_0^2 + y_0^2) + 6(x_0^4 + y_0^4) + 12x_0^2y_0^2)/P, \\
q_{1,2} &= (-5x_0^3y_0 - 5x_0y_0^3 + 9x_0y_0)/P, \\
q_{1,3} &= -7x_0(1 + x_0^2 + y_0^2)/P, \\
q_{2,3} &= -7y_0(1 + y_0^2 + x_0^2)/P,
\end{aligned} \tag{5.17}$$

where

$$P = \det(Q) = 8 + 8(x_0^2 + y_0^2) + 12x_0^2y_0^2 + 6(x_0^4 + y_0^4) + 18(x_0^2y_0^4 + x_0^4y_0^2) + 6(x_0^6 + y_0^6),$$

and

$$q_{3,2} = q_{2,3}, q_{1,2} = q_{2,1}, q_{1,3} = q_{3,1}. \tag{5.18}$$

To summarize, the task for any given point (x_0, y_0) is to compute the function $U_{(x_0,y_0)}$ using (5.2) and the values of $q_{i,j}$ using (5.17) and (5.18), and to use these values to compose the quadratic functional kernel, and then to compute the value of the quadratic functional applied to the function $U_{(x_0,y_0)}$ using (5.11). The result will be zero if (x_0, y_0) is at the focus of expansion.

In the methods presented so far, there are two ways to compute the error surface value $E(x_0, y_0)$ from $U_{(x_0,y_0)}(x, y)$. One way is to convolve with a kernel (center-surround kernel method) or, to compute an explicit projection on a subspace (quadratic polynomial projection method, subspace projection method) and then to eliminate this projection from $U_{(x_0,y_0)}(x, y)$; the norm of the function left after this operation is the error surface value. The second way is to calculate the quadratic functional kernel $R(x, y, x', y')$ and then to directly compute the error value by doing a quadratic functional computation in the four dimensional space of (x, y, x', y'), as shown in Eqn. (5.15). If there are n points in the discrete domain where the flow values are known, the latter method involving a quadratic functional kernel is of complexity $O(n^2)$ and the former method is of complexity $O(n)$.

5.4 The quadratic error surface result

It will be shown that the error function $E(x_0, y_0)$ in the case of the center-surround kernel method and the quadratic polynomial projection method is itself a quadratic polynomial. The implication of such a result is that one does not have to search over all possible candidate FOEs in order to determine the minimum of the error function. Indeed, there is no need for a minimization procedure, either. Given the values of the function $E(x_0, y_0)$ at six points, the function $E(x_0, y_0)$ is completely determined because of its quadratic nature. The coefficients of the quadratic polynomial can be found from the values at the six points and the minimum of the function can be expressed in closed form in terms of these coefficients.

5.4.1 The center-surround kernel method

For the center-surround kernel method, the function $E(x_0, y_0)$ is obtained by a convolution process. Note first that the function $U_{(x_0,y_0)}(x, y)$ is linear in x_0 and y_0:

$$U_{(x_0,y_0)}(x, y) = V(x, y) \cdot (-y + y_0, x - x_0).$$

Let us make explicit the linearity in (x_0, y_0):

$$U_{(x_0,y_0)}(x, y) = A(x, y)x_0 + B(x, y)y_0 + C(x, y), \tag{5.19}$$

where the functions $A(x, y)$, $B(x, y)$ and $C(x, y)$ will depend on the flow field $V(x, y)$. The convolution kernel K, which is also a function of x and y, is independent of (x_0, y_0). So, when $U_{(x_0,y_0)}$ is convolved with K, the resulting function $\Phi_{(x_0,y_0)}$ is still linear in x_0 and y_0:

$$\begin{aligned}\Phi_{(x_0,y_0)}(x, y) &= U_{(x_0,y_0)} * K(x, y)\\ &= (A * K(x, y))x_0 + (B * K(x, y))y_0 + (C * K(x, y))\\ &= A_1(x, y)x_0 + B_1(x, y)y_0 + C_1(x, y),\end{aligned}$$

where $A_1(x, y)$, $B_1(x, y)$ and $C_1(x, y)$ are respectively, the results of the convolution of $A(x, y)$, $B(x, y)$ and $C(x, y)$ with the kernel $K(x, y)$. The form of $\Phi_{(x_0,y_0)}$ is seen to be linear in x_0 and y_0. Note that this observation does not depend on the choice of K, as long as K is independent of (x_0, y_0). The error function $E(x_0, y_0)$ is defined to be the norm square of $\Phi_{(x_0,y_0)}$ (see Eqn. (5.10)):

$$E(x_0, y_0) = \| \Phi_{(x_0,y_0)}(x, y) \|^2 .$$

Since we know that $\Phi_{(x_0,y_0)}$ is linear in x_0 and y_0, we can show that $E(x_0, y_0)$ is quadratic in x_0 and y_0. In the following, for simplicity, we do not show the dependence on x and y.

$$\begin{aligned}E(x_0, y_0) &= \| \Phi_{(x_0,y_0)} \|^2\\ &= \iint [A_1 x_0 + B_1 y_0 + C_1]^2 \, dxdy\\ &= \| A_1 \|^2 x_0^2 + \| B_1 \|^2 y_0^2 + 2 \| A_1 B_1 \| x_0 y_0 + 2 \| A_1 C_1 \| x_0\\ &\qquad +2 \| B_1 C_1 \| y_0 + \| C_1 \|^2 .\end{aligned} \tag{5.20}$$

For the variance case,

$$E(x_0, y_0) = \iint (\Phi_{(x_0,y_0)} - \Phi^a_{(x_0,y_0)})^2 dxdy \tag{5.21}$$

Here $\Phi^a_{(x_0,y_0)}$ is the mean value of the function $\Phi_{(x_0,y_0)}$, and is also linear in x_0 and y_0. Thus the expression inside the integral in Eqn. 5.21 is linear in x_0 and y_0. By a development similar to the one in Eqn. (5.20), the error function is quadratic in this case also.

Thus, $E(x_0, y_0)$ is a quadratic polynomial in x_0 and y_0, for the center-surround kernel method.

5.4.2 The quadratic polynomial projection method

For the quadratic polynomial projection method, the error function $E(x_0, y_0)$ is obtained as the norm of the orthogonal complement function which is the residual left after subtracting off the projection on to the space of quadratic polynomials. That is, $E(x_0, y_0)$ is the norm of the following function reproduced from Eqn. (5.14):

$$\Phi(x,y) = U_{(x_0,y_0)}(x,y) - \sum_{i=1}^{6} < U_{(x_0,y_0)}, \phi_i > \phi_i(x,y).$$

The function $U_{(x_0,y_0)}(x,y)$ is linear in x_0 and y_0 (see Eqn. (5.19). We have noted that the basis functions ϕ_i that span the space of quadratic polynomials are independent of x_0 and y_0. This means that the projections

$$\begin{aligned} < U_{(x_0,y_0)}, \phi_i > \phi_i(x,y) &= < A, \phi_i > x_0\phi_i(x,y) + < B, \phi_i > y_0\phi_i(x,y) \\ &\quad + < C, \phi_i > \phi_i(x,y) \end{aligned}$$

are linear functions of x_0 and y_0. Here, we have used the linearity of the inner product. Continuing, we see that the summation

$$\sum_{i=1}^{6} < U_{(x_0,y_0)}, \phi_i > \phi_i(x,y)$$

and so the orthogonal complement

$$\Phi(x,y) = U_{(x_0,y_0)}(x,y) - \sum_{i=1}^{6} < U_{(x_0,y_0)}, \phi_i > \phi_i(x,y)$$

are also linear functions of x_0 and y_0. As a result the error, which is the norm square of the function $\Phi(x,y)$, is quadratic in x_0 and y_0, as shown for the center-surround kernel method in Eqn. (5.20).

5.4.3 Invariance under noise

It can be easily seen that additive noise will contribute another linear term to the functions; thus, the error function would still be quadratic. Let $N(x,y)$ represent the additive noise in the optical flow computation. We replace $V(x,y)$ by $V(x,y) + N(x,y)$ in the equations involving $V(x,y)$. The circular-component function for this noisy flow field is

$$U_{(x_0,y_0)}(x,y) = (V(x,y) + N(x,y)) \cdot (-y + y_0, x - x_0).$$

The function $U_{(x_0,y_0)}(x,y)$ still remains linear in x_0 and y_0. Thus, the reasoning for the quadratic nature of the error function $E(x_0, y_0)$ would still carry through as in the noiseless case. In general, the vector flow field $V(x,y)$ in the definition of the circular-component function can be replaced by any arbitrary vector field (in particular, such a flow field can be produced by an arbitrary transformation of the optical flow field); but the error function $E(x_0, y_0)$ would still be quadratic in x_0 and y_0 by means of the same arguments we presented so far. However, the minimum of the quadratic surface need no longer be at the correct FOE. Indeed, in general, the minimum would occur at a location different from the correct FOE because of the transformation brought upon $V(x, y)$. Even though the nature of the surface remains quadratic, the minimum value need no longer be zero. The departure of the minimum value from zero is a measure of the noise in the flow field.

5.4.4 Using the quadratic error surface result

So, the algorithm in both the center-surround kernel method and the quadratic polynomial projection method is as follows: Choose six locations (x_{0i}, y_{0i}). The choice of the locations is immaterial, and could be done at random, as long as they are in a non-degenerate configuration. For each (x_{0i}, y_{0i}), compute the error function $E(x_{0i}, y_{0i})$ by convolving or projecting the circular component $U_{(x_0,y_0)}(x, y)$ (depending on which method is used) and then taking the norm of the resulting function. Fit a quadratic surface to these six values of E. The quadratic surface is of the form

$$E(x_{0i}, y_{0i}) = a_1 x_{0i}^2 + a_2 y_{0i}^2 + a_3 x_{0i} y_{0i} + a_4 x_{0i} + a_5 y_{0i} + a_6.$$

Since we know the value of E for six (x_{0i}, y_{0i}), we get six linear equations in the six unknowns a_j, $1 \le j \le 6$. This linear system can be solved using standard techniques. Of course, one could choose a specific set of six points (x_{0i}, y_{0i}) in advance and precompute the inverse matrix required in the linear system solution.

The minimum of this quadratic surface E, which is the FOE, can be easily determined as a closed-form function of the parameters of the quadratic surface. The minimum of the surface is located at

$$\left(\frac{2a_1a_5 - a_3a_4}{a_3^2 - 4a_1a_2}, \frac{2a_2a_4 - a_3a_5}{a_3^2 - 4a_1a_2} \right).$$

as can be seen by solving for that (x_{0i}, y_{0i}) for which

$$\frac{\partial E}{\partial x_{0i}} = 0, \text{ and } \frac{\partial E}{\partial y_{0i}} = 0.$$

Note that the coefficients a_j are determined from the values of E at six points by solving a linear system of equations. In this sense, no search is involved in the center-surround kernel and the quadratic polynomial projection methods. However, the error surface in the third method (the subspace projection method) does not have the quadratic nature because the projection itself depends on the values of x_{0i} and y_{0i}. So, in this case, a good strategy for locating the minimum would be of advantage.

Note that obtaining the values of E at more than six points is redundant. This is so because the quadratic error surface is perfectly quadratic, irrespective of the input flow field. So, no gain is achieved by computing the value of E at more than six points.

5.5 A fast method: the NCC algorithm

In this section, we present a variation of the FOE search algorithm that offers certain computational advantages over the previous algorithms. In certain circumstances, the method performs better. This method is an approximate method in that it requires that the rotational parameters be small. We want to ignore the part of the flow field resulting from the rotational motion. In other words, we require that the translational motion dominate. Then we can avoid doing one of the steps (that of eliminating the contribution to the flow field by the rotational velocity) in the previous methods. The result is a faster algorithm (compared to the ones presented so far) that does very well under pure or nearly-pure translational motion. In most real-life motion there is hardly any rotation; the fast method we present here will be useful under such conditions.

The circular component function is composed of two parts, one ($U^v_{(x_0,y_0)}$) from the translational velocity, and the other ($U^\omega_{(x_0,y_0)}$) due to the rotational velocity:

$$U_{(x_0,y_0)}(x,y) = U^v_{(x_0,y_0)}(x,y) + U^\omega_{(x_0,y_0)}(x,y).$$

Recall that all the FOE search methods described so far try to eliminate $U^\omega_{(x_0,y_0)}$ by either convolving with a kernel (center-surround kernel method) or by projecting on to a subspace (quadratic polynomial projection method, subspace projection method). Here we consider the option of ignoring the step of eliminating $U^\omega_{(x_0,y_0)}$. To this end, we define an error function $E(x_0,y_0)$ as the norm of $U_{(x_0,y_0)}(x,y)$:

$$E(x_0,y_0) = \|U_{(x_0,y_0)}\|^2. \tag{5.22}$$

The important observation, as noted before, is that $U_{(x_0,y_0)}(x,y)$ is *linear* in the parameters x_0 and y_0:

$$U_{(x_0,y_0)}(x,y) = V(x,y)\cdot(-y+y_0, x-x_0),$$

or, more explicitly,

$$U_{(x_0,y_0)}(x,y) = A(x,y)x_0 + B(x,y)y_0 + C(x,y),$$

where the functions $A(x,y)$, $B(x,y)$ and $C(x,y)$ will depend on the flow field $V(x,y)$. The norm of $U_{(x_0,y_0)}(x,y)$ as shown in Eqn. (5.22) gives an error function

$$\begin{aligned}
E(x_0,y_0) &= \| \, U_{(x_0,y_0)}(x_0,y_0) \, \|^2 \\
&= \iint [Ax_0 + By_0 + C]^2 \, dx dy \\
&= \| \, A \, \|^2 \, x_0^2 + \| \, B \, \|^2 \, y_0^2 + 2 \, \| \, AB \, \| \, x_0 y_0 \\
&\qquad +2 \, \| \, AC \, \| \, x_0 + 2 \, \| \, BC \, \| \, y_0 + \| \, C \, \|^2 .
\end{aligned}$$

Thus the error function defined in Eqn. (5.22) is *quadratic* in x_0 and y_0. The minimum of this quadratic surface is purported to occur at the Focus of Expansion (FOE). We will justify this claim shortly. But first, if the claim is correct, we have a simple procedure [67] – the Norm of the Circular Component (NCC) algorithm – that we describe now.

The first step is to choose six candidate coordinate positions (x_{0i}, y_{0i}) in a non-degenerate configuration (i.e., six points on the image plane that are not on a straight line). Next, for each of these candidates, compute the circular component function and define $E(x_{0i}, y_{0i})$ to be the norm of that function. In a discrete setting, the error value is simply the sum of the squares of the circular component values. Note that this can be done even in the case of a sparse flow field. The error function values at these six points completely define the error surface because of its quadratic nature. If we write the error surface to be

$$E(x_{0i}, y_{0i}) = a_1 x_{0i}^2 + a_2 y_{0i}^2 + a_3 x_{0i} y_{0i} + a_4 x_{0i} + a_5 y_{0i} + a_6,$$

we can solve for the unknowns $a_j, 1 \leq j \leq 6$, given the error values for the six candidates $(x_{0i}, y_{0i}), 1 \leq i \leq 6$, by solving a system of six linear equations in six unknowns. The location of the minimum can be found using the closed-form expression

$$\left(\frac{2a_1a_5 - a_3a_4}{a_3^2 - 4a_1a_2}, \frac{2a_2a_4 - a_3a_5}{a_3^2 - 4a_1a_2} \right),$$

as shown in the previous section. This location is the computed FOE. Thus the algorithm involves computing the norm of the circular component function for six candidates, calculating the parameters of the quadratic error surface by solving a linear system, and then computing the FOE using a closed form expression.

Let us now examine the claim about the minimum being at the the FOE. Note that the function $U_{(x_0,y_0)}(x, y)$ is made up of two parts; one is the translational part shown in Eqn. (5.4), and the other is the rotational, in (Eqn. (5.3)):

$$U^v_{(x_0,y_0)}(x, y) = v_3 \rho(x, y) \cdot \left[(y_0 - \eta)x + (-x_0 + \tau)y + \eta x_0 - \tau y_0 \right],$$

and,

$$\begin{aligned} U^\omega_{(x_0,y_0)} \quad = \quad & \omega_1 \left[-\frac{x_0}{f} y^2 + \frac{y_0}{f} xy + fx - fx_0 \right] \\ & + \omega_2 \left[-\frac{y_0}{f} x^2 + \frac{x_0}{f} xy + fy - fy_0 \right] \\ & + \omega_3 \left[-y^2 - x^2 + x_0 x + y_0 y \right]. \end{aligned}$$

We can rewrite these as

$$U^v_{(x_0,y_0)}(x, y) = f_1 x_0 + f_2 y_0 + f_3,$$

where

$$\begin{aligned} f_1 &= v_3 \rho(x, y)(-y + \eta), \\ f_2 &= v_3 \rho(x, y)(x - \tau), \text{ and} \\ f_3 &= v_3 \rho(x, y)(-x\eta + y\tau). \end{aligned}$$

Similarly,

$$U^{\omega}_{(x_0,y_0)}(x,y) = g_1 x_0 + g_2 y_0 + g_3,$$

where

$$\begin{aligned} g_1 &= -\frac{\omega_1}{f}y^2 + \frac{\omega_2}{f}xy + \omega_3 x - \omega_1 f, \\ g_2 &= \frac{\omega_1}{f}xy - \frac{\omega_2}{f}x^2 + \omega_3 y - \omega_2 f, \text{ and} \\ g_3 &= \omega_1 f x + \omega_2 f y + \omega_3(-y^2 - x^2). \end{aligned}$$

The translational part $U^v_{(x_0,y_0)}(x,y)$ vanishes at the FOE, as shown in Eqn. (5.5) repeated here:

$$U^v_{(\tau,\eta)}(x,y) = v_3 \rho(x,y) \cdot \begin{bmatrix} x-\tau \\ y-\eta \end{bmatrix} \cdot (-y+\eta, x-\tau) = 0,$$

and it is non-zero elsewhere. Thus, the norm $\|U^v_{(x_0,y_0)}(x,y)\|^2$ is positive quadratic with minimum (equal to zero) at the FOE:

$$\begin{aligned} \| U^v_{(x_0,y_0)}(x,y) \|^2 &= \| f_1 \|^2 x_0^2 + \| f_2 \|^2 y_0^2 + 2 \| f_1 f_2 \| x_0 y_0 \\ &\quad + \| f_1 f_3 \| x_0 + \| f_2 f_3 \| y_0 + \| f_3 \|^2 . \end{aligned} \tag{5.23}$$

The minimum is no longer at the FOE once we add the rotational part. However, as long as the contribution from the rotational part is small compared to that from the translational part, we can *approximate* the behavior of $\|U^v_{(x_0,y_0)}(x,y)\|^2$ by $\|U_{(x_0,y_0)}(x,y)\|^2$. We have

$$\begin{aligned} \| U_{(x_0,y_0)}(x,y) \|^2 &= \| h_1 \|^2 x_0^2 + \| h_2 \|^2 y_0^2 + 2 \| h_1 h_2 \| x_0 y_0 \\ &\quad + \| h_1 h_3 \| x_0 + \| h_2 h_3 \| y_0 + \| h_3 \|^2, \end{aligned} \tag{5.24}$$

where

$$h_1 = f_1 + g_1,\ h_2 = f_2 + g_2, \text{ and } h_3 = f_3 + g_3.$$

The norm in Eqn. (5.24) would closely approximate the norm in Eqn. (5.23), if the following conditions are satisfied:

$$\| g_i \| \ll \| f_i \|,\ i = 1,2,3.$$

It is hard to give an exact interpretation of these conditions. However, the conditions are satisfied when the rotational flow magnitudes are small compared to the translational flow magnitudes; in other words, translation should be the dominant cause of the flow field.

The method is exact for pure translation and is approximate when the rotation is small compared to the translation or when the depth of objects is small (i.e., high $\rho(x,y)$) as would be the case in indoor situations. Also, there is no apparent reason for this method to fail in the case where a planar surface occupies the whole field of view. We will show in the next chapter why the planar surface poses problems for the other algorithms and not to the NCC algorithm.

Good results are obtained for this approximate method when tested on real image sequences. Experimental results using this method and the other methods described in this chapter are presented in chapter 8.

Chapter 6

Applicability of the algorithms

6.1 Introduction

In the previous chapters, we presented methods to determine the motion parameters of a moving sensor. In this chapter, we analyze the effect of violating the assumptions, and point out conditions where the algorithms are likely to perform poorly or fail completely.

There are certain intrinsic difficulties in motion analysis which we would like to point out here. When presented with a single planar surface occupying the whole field of view, all the algorithms for motion analysis fail in some way. Mathematically, such a situation produces either unconstrained or underconstrained systems, resulting in a family of solutions. Further information is required to constrain the solution. In general, all algorithms benefit from a rich depth structure. Indeed, Longuet-Higgins has shown that the planar situation is inherently ambiguous [42]. The velocity vector and the plane normal can be interchanged, and reversed in direction, to produce the same flow field.

The assumption of rigidity is quite critical for most motion algorithms. For optical flow-based algorithms, the use of the optical flow equations (Eqn. (3.2)) automatically relies on the rigidity assumption made in deriving the optical flow equations. Thus, if the objects in the scene are not rigid, the methods presented in this thesis, as well as many other optical flow-based methods, will fail, unless the nonrigidity is not apparent due to fine temporal sampling. In that case, it would amount to using a rigid *approximation* to nonrigid motion.

The rigidity constraint is implicit in the *planarity constraint* [41,13] and its various manifestations used in the correspondence based schemes. Violation of rigidity implies that the constraint is no longer valid and the algorithms can be expected to fail.

Here, we analyze the methods presented so far, to identify situations that cause the methods to fail. We start by summarizing the algorithms and then comment on their applicability.

6.2 Flow circulation algorithm

The *Flow circulation algorithm* is easily understood. We choose several closed contours (or *cycles*) on the image plane. Circulation values are obtained for these cycles. A circulation value is simply a contour integral of the vector flow field around the cycle, and measures the "swirling" of the flow field around the cycle. Mathematically, a circulation value is proportional to the average of the curl of the vector field in the region enclosed by the cycle. We have shown in a Chapter 4 that these values, taken as data points at the centroids of the regions, will approximately lie on a linear surface: $g(x, y) = ax + by + c$. Thus, given such data points for several contours, by fitting a linear surface to the given data, the parameters a, b, and c are determined. These are proportional to the rotational parameters of motion ω_1, ω_2, and ω_3 respectively. Clearly, fitting a linear surface to the data is a global process, which we believe will lead to a more stable determination of the rotational parameters than by using local methods based on local deformation parameters (i.e., higher order derivatives of the vector flow field). Further, fitting a linear surface is a particularly simple global process; the coefficients can be determined by a regression analysis that will essentially use weighted sums of the data points.

The fact that the circulation values only *approximately* lie on a linear surface makes the flow circulation algorithm an approximate method. Shortly, we will consider the magnitude of errors that can result pointwise. However, the real issue is the extent of the errors in an average sense, and there are ways that the errors can be managed so as to improve the accuracy of the globally-obtained rotational parameters. We mention three techniques here: (1) discarding of outliers, (2) depth filtering, and (3) surface normal balancing. For discarding outliers, an iterative approach may be used, which first fits a linear surface, and then improves upon the parameters defining the surface by discarding data that lies far from the norm. For depth filtering, we might intentionally discard cycles that enclose points that lie on nearby surfaces. Such surfaces are frequently located near the periphery of the visual field; other nearby surfaces (that do not lie normal to the line of sight) might be identified and discarded from the flow circulation algorithm by independent depth sensing mechanisms.

Recalling the equation for the curl,

$$\nabla \times V(x_0, y_0) = \\ - \left[x_0 \left(\omega_1 + \frac{n_2 v_3}{R} \right) + y_0 \left(\omega_2 + \frac{n_1 v_3}{R} \right) + \left(2f\omega_3 + \frac{n_1 v_2 - n_2 v_1}{R} \right) \right]. \tag{6.1}$$

For surface normal balancing, we note from Eqn. (6.1) that perturbations to the estimates of ω_1, ω_2, and ω_3 are mediated by the values of the surface component normals n_2, n_1, and $n_1 v_2 - n_2 v_1$ respectively. Here, n_1 and n_2 are the horizontal and vertical components of the surface normal. The vector (v_1, v_2) is the (orthographic) projection of the velocity vector onto the image; so, $n_1 v_2 - n_2 v_1$ is the cross product of the projections on the image plane of the velocity vector (which will be in the same direction as the FOE) and the surface normal. Thus if the circulation values obtained for the flow circulation algorithm encompass elements such that the tilts (n_1, n_2) are balanced, then errors will cancel. When surfaces with large tilts are present, we should choose circulation values such that the union of the enclosed regions contains surfaces that have an even balance of tilts with respect to the horizontal axis, the vertical axis, and the axis defined by the direction to the focus of expansion. Again, independent surface structure methods, such as stereo and shading clues, may be used to make the selection.

Now, we investigate the validity of the linearity assumption for the flow circulation values. We first consider the situation with pointwise evaluations of the curl. Looking at Eqn. (6.1), our ability to estimate the parameters ω_1, ω_2, and ω_3 will depend on the size of the remaining terms in the corresponding coefficients. That is, we need that the terms

$$\frac{n_2 v_3}{R}, \quad \frac{n_1 v_3}{R}, \quad \frac{n_1 v_2 - n_2 v_1}{2R}$$

be small in magnitude, respectively, relative to the expected sizes of ω_1, ω_2, and ω_3. In the worst case, the magnitudes of any of these three terms can attain the respective values of

$$\frac{|v_3|}{R}, \quad \frac{|v_3|}{R}, \quad \frac{\sqrt{v_1^2 + v_2^2}}{2R}.$$

If this is all the information that is given, then we will need large values of R. Specifically, suppose that a reasonable value for a rotational parameter is on the order of 0.1 or 0.2 radians per second. Then as long as R is greater than $|v_3|$ by a factor of 50 or so, the rotational parameters ω_1 and ω_2 should be deducible to within an accuracy of 0.02 radians per second. Likewise, R should be larger than the magnitude of the lateral velocity in the (X, Y)-plane by a factor of 25 or so. Recalling that R is the distance from the nearest approach of the tangent plane to the focal point, we observe that tangent planes to surfaces visible in the scene should, for the most part, stay outside of a sphere whose radius is the distance to be traversed at the current velocity in the next 50 seconds. This is quite a large bound, and hardly ever true in practice, unless the translational velocity is zero. However, if the forward velocity component is zero, then accurate estimation of ω_1 and ω_2 is assured by this method, whereas if the lateral translational velocity is zero, then ω_3 will be precise.

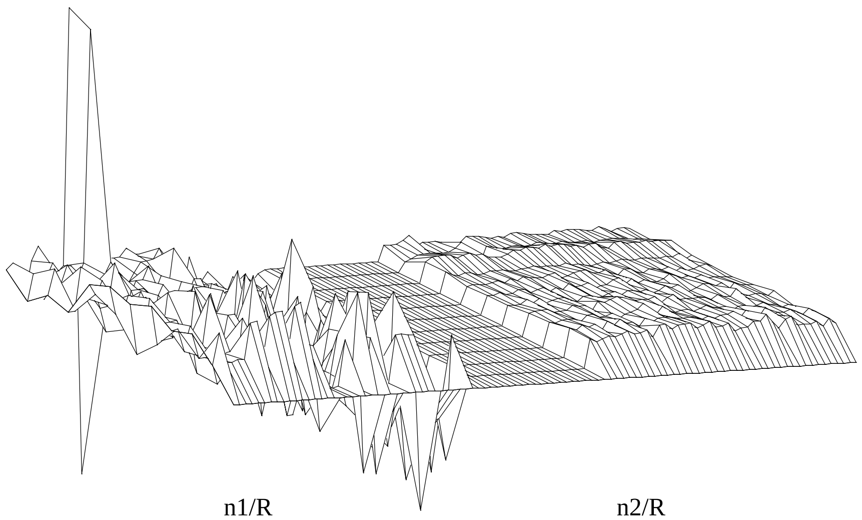


Figure 6.1: A plot of n_1/R is seen in the left, and a plot of n_2/R of the same image domain is on the right, for a typical scene consisting of a road, imaged by a depth sensor. The depth data is used to compute the surface normals. The values of n_2/R are mostly positive, and average to 0.0103, whereas the values of n_1/R are better distributed around zero, averaging to 0.0040.

More realistically, it is the *average* values of n_1 and n_2 throughout the image that influences the accuracy of the linear regression that is used to estimate the components of ω. For example, if n_1 and n_2 average to 0.1, then the previous bounds may be reduced by a factor of 10. Although typical values of the average surface normal tilts must be determined empirically, many scenes are composed of a variety of tilt directions. Such variety makes it possible to apply the *surface normal balancing* technique. Fig. (6.1) shows plots of n_1/R and n_2/R for a typical scene of a road, computed using depth data. The values of n_1/R are well-distributed about zero, and average to 0.0040. The values of n_2/R, on the other hand, are predominantly positive, and average to 0.0103. The result is that using this scene, the value of the horizontal rotational component ω_1 is likely to be estimated with greater error when using the flow circulation algorithm.

We observe in this example that the rotation component about the horizontal axis, ω_1, is confounded by forward velocity and horizontal surfaces with large n_2/R. Likewise, rotation about the vertical axis, ω_2, which would arise as a sensor rotates from left to right, is confounded by a forward velocity and a surface patch whose tilt lies horizontally.

Consider for instance a person walking next to a vertical wall (see Fig. (6.2)). Instantaneously, the motion field induced by the wall can indicate either forward velocity or rotation of the head away from the wall. As shown in Fig. (6.2), the two instantaneous flow fields are similar, although not equal. The curl of these flow fields, along the wall, are equal (equal curls do not imply equal vector fields). In any given vertical slice along the wall, the flow fields are qualitatively extremely similar. The sense in which they differ is that the magnitude of the vector field due to forward translation decreases with the range, so that the velocity vectors are quite small at the far end of the wall. Since the curls are identical, the flow circulation algorithm will not distinguish between the two cases. The ambiguity may easily be resolved by higher-level processing, such as analyzing over time the scene at the gaze point. A similar analysis shows that rotation about the line of sight (optical axis) is readily confused with lateral motion with respect to a surface such that $-n_2v_1 + n_1v_2$ is large.

Our conclusion is that for some scenes, the flow circulation algorithm should be able to estimate the rotational parameters correctly; for other scenes, accurate estimation of rotational parameters from the velocity field using the flow circulation algorithm is an unstable process, and that other processing, such as scene analysis of the gaze point or global analysis of some feature of the flow field in addition to the curl will be necessary.

The confounding of parameters is more general than what has been presented here. It has been observed [35,2] that by reducing the size of the field of view, *optical flows* from a horizontal rotation and that from a leftward FOE can be made to look very similar. We discuss this problem, with an example, in the next section.

One interesting observation about this algorithm is that the most favorable case for the algorithm in terms of depth structure, is the frontal planar situation. This should not be surprising because in this algorithm, the quantities being computed are the rotational parameters whose

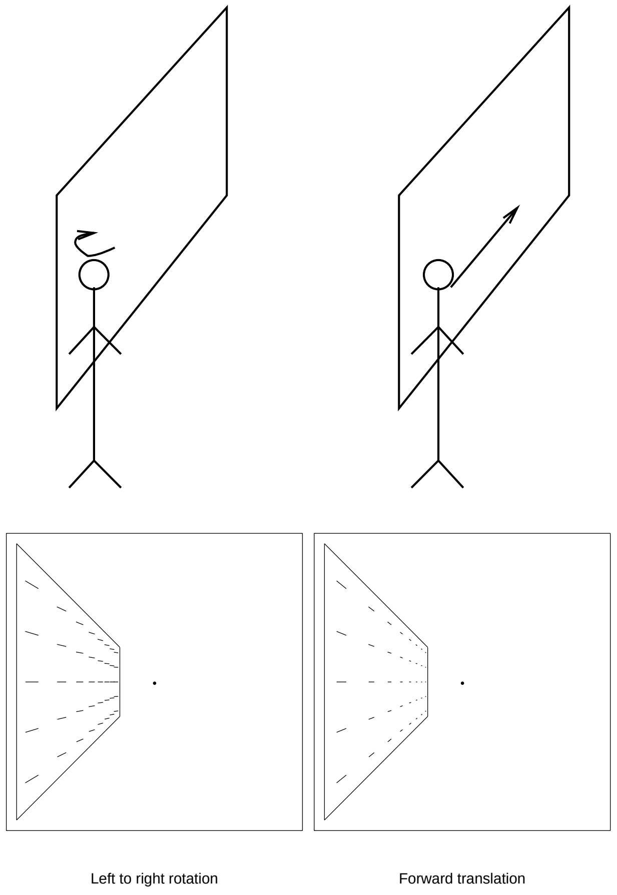


Figure 6.2: Shown on top are two situations where the curl values will be identical for specific choices of the parameters. The flow fields, shown in the bottom figure, are not identical, but are very similar.

influence on the overall optical flow does not depend on the depth structure. It is simply that frontal planarity is the most favorable assumption for this algorithm to get rid of the translational parameters.

6.3 The FOE search algorithms

The *FOE search algorithm* is an exact algorithm, in the sense that barring noise, the focus of expansion will be found. For every candidate point, the circular-component flow velocity function is computed according to Eqn. (5.2). The resulting function is used in a quadratic functional computation, which is used to determine if the function is a quadratic polynomial of the appropriate form. The functional is zero at the focus of expansion.

Three methods were suggested: the center-surround kernel method, the quadratic polynomial projection method, and the subspace projection method. The methods determine, respectively, whether the circular-component flow function has constant Laplacian, is a quadratic polynomial, or is a quadratic polynomial in the proper three-dimensional subspace. Since the error surface for the first two methods is itself a quadratic function, that surface is either identically zero, or there will be at most one zero of the error surface, which, in the noise-free case, must lie at the focus of expansion. Of the three methods suggested, the center-surround kernel method is the simplest for determining whether the circular-component flow function is a quadratic polynomial.

Since $U^{\omega}_{(x_0,y_0)}$ (Eqn. (5.3)) projects to zero (under any of the methods), errors arise solely due to noise in the flow field. The noise is balanced against the error surface E that results from the projection of $U^{v}_{(x_0,y_0)}$ (Eqn. (5.4)). If the scene consists solely of a plane, tilted in any direction, then $\rho(x,y)$ will be linear. This is because such a planar surface can be written as

$$n_1X + n_2Y + n_3Z = R,$$

where R is constant, and this yields

$$\rho(x,y) = \frac{1}{Z} = \frac{n_1x + n_2y + n_3}{R}. \tag{6.2}$$

Eqn. (5.4) is reproduced here:

$$U^{v}_{(x_0,y_0)}(x,y) = v_3\rho(x,y)\cdot\left[(y_0-\eta)x + (-x_0+\tau)y + \eta x_0 - \tau y_0\right]. \tag{6.3}$$

Substituting the expression for $\rho(x,y)$ from Eqn. (6.2) into Eqn. (6.3), we get

$$U^{v}_{(x_0,y_0)}(x,y) = \frac{v_3(n_1x + n_2y + n_3)}{R}\left[(y_0-\eta)x + (-x_0+\tau)y + \eta x_0 - \tau y_0\right]. \tag{6.4}$$

Even without going into the tedium of expanding this, one can easily see that $U^{v}_{(x_0,y_0)}(x,y)$ is a quadratic polynomial in x and y. Thus, in this case, an attempt to cancel the quadratic polynomial $U^{\omega}_{(x_0,y_0)}(x,y)$ using the center-surround kernel methods or the quadratic polynomial projection method will not only eliminate $U^{\omega}_{(x_0,y_0)}(x,y)$, but $U^{v}_{(x_0,y_0)}(x,y)$ as well! As a result, the function $E(x,y)$ will be zero everywhere for these methods. However, the subspace projection method would still succeed because the form of the quadratic polynomial in Eqn. (6.4) is substantially different from that of $U^{\omega}_{(x_0,y_0)}(x,y)$ in Eqn. (5.3). However, if there exist two or more planar surfaces, then E will generally be nonzero (or non-constant for the third kernel) except at the focus of expansion, and any of the methods can be used.

If the variation in depth is insignificant compared to the average depth to the objects in the scene, a *near-planarity* situation arises and in practice, this causes the same problems as the planarity situation discussed above.

Except for these degenerate cases, the center-surround kernel method works well in our experiments. If the convolution kernel used is the radial derivative of the Laplacian of a Gaussian (Eqn. (5.6)), then $E(x_0,y_0)$ is the norm of the function

$$\frac{\partial}{\partial r}\Delta G_\sigma * \left[v_3\rho(x,y)\cdot((y_0-\eta)x+(-x_0+\tau)y+\eta x_0-\tau y_0)\right],$$

which we rewrite as

$$\frac{\partial}{\partial r}\Delta G_\sigma * \left[v_3\rho(x,y)\cdot h(x,y)\right]. \tag{6.5}$$

where $h(x,y)$ represents the linear expression in parantheses. This function is zero when $(x_0,y_0)=(\tau,\eta)$, since $h(x,y)\equiv 0$ there. For other locations of (x_0,y_0), we want this function to be significantly nonzero.

From elementary calculus, the Laplacian of a product of two functions $u(x,y)$ and $v(x,y)$ is given by

$$\Delta(u\cdot v)=u\Delta v+2\nabla u\nabla v+v\Delta u.$$

Using this formula in Eqn. (6.5), and noting that

$$\frac{\partial}{\partial r}\Delta G_\sigma * f(x,y)=G_\sigma*\frac{\partial}{\partial r}\Delta f(x,y),$$

we get

$$G_\sigma*\frac{1}{\sqrt{x^2+y^2}}\left(x\frac{\partial}{\partial x}+y\frac{\partial}{\partial y}\right)\left[h(x,y)\Delta\rho+c\cdot D_{\mathbf{w}}\rho\right],$$

where $D_{\mathbf{w}}$ is a directional derivative in a fixed direction (which depends on (x_0,y_0)). Clearly, discontinuities and sudden jumps in ρ will lead to large Laplacian and gradient values which will contribute to the residual, and thus to $E(x_0,y_0)$. We thus see that the center-surround kernel method benefits from scenes with depth variations.

The quadratic polynomial and subspace projection methods have the advantage that they can be used with sparse data, if necessary, by fitting the appropriate quadratic polynomials to the distributed data.

As noted before, the effects due to the translational parameters and the rotational parameters are not completely *independent*. For instance, rotation with respect to the vertical axis produces

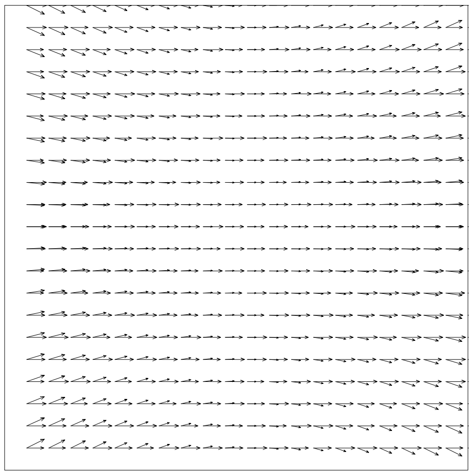


Figure 6.3: Two flow fields, shown superimposed, one due to horizontal translation and the other, due to rotation with respect to the vertical axis. The field of view is about 90 degrees. Note that near the center, the flow fields are very similar

a flow field that is similar to that produced due to a horizontal translation (leftward FOE). The two flow fields produced for these two cases are shown superimposed in Fig. (6.3). Notice that if the field of view is small, it is nearly impossible to distinguish between the two flow fields. In this particular case, the depth structure was chosen to be random, the variation in depth being small compared to the average depth. This is a *nearly-planar* case, and was chosen to emphasize the similarity of the flow fields produced due to the two different kinds of motion. In a real situation, even if the similarity is not as striking, qualitatively, the flow fields would still remain similar, and, with sufficient perturbation due to noise or other ill-effects of the flow field computation, can result in a complete misinterpretation the flow field.

All the FOE search methods described here attempt to cancel the rotational parameters. So, when there is a large leftward motion (high τ), for small field-of-view, the flow field which will look like that due to a substantial value of ω_2, can be interpreted so, thus missing the leftward translation. A similar argument holds for large η. In essence, for noisy flow fields and a small field-of-view, the methods described here tend to *push* the FOE as close to the origin as possible. This is because a portion of the contribution from the translation is erroneously removed from consideration, having been mistaken as arising due to rotation.

6.4 The fast method

The fast method to locate the FOE is an approximate version of the FOE search algorithm. In this method, the effect due to rotation is assumed to be small compared to that due to translation. Eliminating the convolution/projection step of the previous methods yields this fast method. The method involves no search because of the known form of the error surface. In this method, the FOE is considered to be the minimum of the norm $E(x_0, y_0)$ of the circular-component function

$$U_{(x_0,y_0)}(x,y) = U^v_{(x_0,y_0)}(x,y) + U^\omega_{(x_0,y_0)}(x,y).$$

The norm is computed for six candidate points (x_0, y_0) and the minimum of $E(x_0, y_0)$ is found using a closed form expression.

As before, the confounding of parameters affects this method, though it manifests in an indirect way. Certain combination of the motion parameter values results in incorrect points being identified as the FOE. Recall that

$$U^v_{(x_0,y_0)}(x,y) = v_3\rho(x,y) \cdot [(y_0 - \eta)x + (-x_0 + \tau)y + \eta x_0 - \tau y_0],$$

and

$$\begin{aligned} U^\omega_{(x_0,y_0)}(x,y) &= \omega_1\left[-\frac{x_0}{f}y^2 + \frac{y_0}{f}xy + fx - fx_0\right] \\ &+\omega_2\left[-\frac{y_0}{f}x^2 + \frac{x_0}{f}xy + fy - fy_0\right] + \omega_3\left[-y^2 - x^2 + x_0x + y_0y\right]. \end{aligned}$$

It is always possible to choose values for the parameters (v_1, v_2, v_3), $(\omega_1, \omega_2, \omega_3)$ and $\rho(x,y)$ in such a way that $U_{(x_0,y_0)}(x,y)$ is reduced to a function of the form

$$h_1(x,y)x_0 + h_2(x,y)y_0$$

where $h_1(x,y)$ and $h_2(x,y)$ are functions that don't involve x_0 or y_0, while choosing an FOE that is not $(0,0)$. In this case, $E(x_0, y_0) = \| U_{(x_0,y_0)} \|^2$ will be zero at $(0,0)$. One example is a choice of the following values:

$$v_2 = 0, v_3 = 1, \omega_1 = 0, \omega_2 = -v_1, \omega_3 = 0, \rho(x,y) = 1.$$

In this case,

$$U_{(x_0,y_0)}(x,y) = (-y - \tau xy)x_0 + (x - \tau x^2)y_0.$$

The FOE is not (0,0) for any non-zero v_1 (and hence a non-zero τ), but the minimum of the $U_{(x_0,y_0)}(x,y)$ is at $(x_0 = 0, y_0 = 0)$ which is incorrect. Notice that the interaction is between ω_2 and v_1, as seen in the qualitative descriptions of the previous section. A similar situation will be observed when ω_1 and v_2 are related in an analogous fashion.

The performance of this algorithm is explored empirically in Chapter 8; it is found to work satisfactorily for real sequences where translation is predominant.

6.5 Relationship to Heeger and Jepson method

Heeger and Jepson [24] observe that for fixed (τ, η), the optical flow equations 5.1 are linear in the remaining collection of unknowns ω, $(v_3 \cdot \rho(x_i, y_i))_{i=1}^N$. We thus have $2N$ linear equations in $N+3$ unknowns, in the case when (τ, η) is fixed. As long as $N \geq 4$, these are easily solved to give the least mean square error, and that error is also easily determined. The residual error in the least mean square solution can be used as a measure of the quality of the estimate (τ, η) for the focus of expansion. In noise-free circumstances, there should be zero residual error if τ and η are the correct values. Equivalently, we see that the data $(u(x_i, y_i), v(x_i, y_i))_{i=1}^N$ regarded as a vector in $2N$-space, should lie on an $N+3$-dimensional hyperplane defined by the fixed (τ, η); the extent that the vector lies off this hyperplane measures the noise in the data and the inexactness of the (τ, η) estimate. Heeger and Jepson then simply check a large array of possible (τ, η) values, computing the error at each such position. When the error is minimized, they declare the correct (τ, η) to be found.

Our FOE search algorithm is the same algorithm, but derived under the assumption that the data is given on a continuum of points, rather than at N distinct points. In their exposition of the algorithm, Heeger and Jepson make use of the projection on to vectors in a space orthogonal to the range of the transformation that takes an $N+3$ vector of depth values and the rotation values

$$[\rho_1, \rho_2, \ldots \rho_n, \omega_1, \omega_2, \omega_3]$$

to the vector of the $2N$ flow components. The vectors in this *orthogonal* space are of the form [19,32]:

$$\Psi_j(T) = \sum_{i=1}^{N} c_i (s_i \times T)$$

The cross product $s_i \times T$ is nothing but the *circular vector field* at the point i, because s_i is the vector from the focal point to the ith point and T is the scaled translation vector (recall that the focal point is at distance f behind the image plane on which the points lie). Thus projecting on to the vector $\Psi_j(T)$ is a discrete formulation of the inner product of the optical flow field and the circular flow field described in our FOE search algorithm.

In a certain sense, their assumption, of a discrete collection of data, is more realistic in image processing applications. However, the projection method is then dependent on the locations of the distinct points, and would have to be computed for each new collection of points. The FOE search algorithm presented here makes clearer the analytical structure of the problem, and provides a method that is independent of the sampling locations, assuming a sufficient density of values are obtained. When there are only a few discrete sampling locations, the FOE search algorithm may still be used, in a modified form, and will essentially be equivalent to the Heeger and Jepson algorithm. Specifically, the circular-component functions may still be computed, but they will be defined only at the collection of sample points. The test to determine the focus of expansion then checks whether the discrete collection of data located at the sample points corresponds to point evaluates of a quadratic polynomial.

In the next two chapters, we will see the results of applying our algorithms to synthetic and real data.

Chapter 7

Experimental Results: Synthetic Data

7.1 Introduction

In the previous chapters, several methods to compute the motion parameters of a moving sensor were presented and analyzed. In this chapter, we present the implementation details of the algorithms and show experimental results from synthetic data. In the next chapter, we present results from actual image sequences.

7.2 Rotation estimation

Recall that the rotation estimation method involves the computation of curl values at several locations on the image plane. Three kinds of experiments are possible. The first is to determine circulation values, which are equivalent to the curl values of the flow field at the centroids of the circulation contour areas. The second way is to compute the curl values directly by taking discrete differences; this is possible if the flow field is synthetic or not noisy. Alternatively, one can compute the curl values analytically if the structure of the environment is known. Here we shall present experiments of all these three types.

The optical flow field can be generated synthetically using the equations in (3.2), providing the depth data ($Z(x,y)$) and the motion parameters ($\mathbf{T}$ and ω) are known. If depth information is available analytically or as the output of a sensor, we can synthesize flow fields by choosing appropriate values for the motion parameters. We begin by taking an object of known shape to be in the environment and compute the curl values analytically.

Consider the image of an ellipsoid defined by

$$\left(\frac{X - X_0}{a}\right)^2 + \left(\frac{Y - Y_0}{b}\right)^2 + \left(\frac{Z - Z_0}{c}\right)^2 = 1. \tag{7.1}$$

We assume that the ellipsoid center is 150 focal units away (along the Z-axis), and the semiradii are 50, 30 and 70 focal units respectively. We assume the ellipsoid is in front of a planar background. Choosing an observer moving with a velocity of (0.3, 0, 0.2) focal units per second, and rotating with an angular velocity of (0.2, 0.1, 0.5) radians per second, we can analytically compute the curl of the flow field. This is possible since we can compute $\frac{\partial \rho}{\partial x}$ and $\frac{\partial \rho}{\partial y}$, given the equation of the ellipsoid in Eqn. (7.1). Recalling that $\rho = 1/Z$, $x = fX/Z$ and $y = fY/Z$, multiplying both sides of Eqn. (7.1) by $f\rho$ and taking derivatives with respect to x, we get

$$\frac{\partial \rho}{\partial x}(x,y) = \frac{1}{d(x,y)}\left(\frac{x - f\rho X_0}{f^2 a^2}\right),$$

where $d(x,y)$ is given by

$$d(x,y) = \left(\rho + \frac{X_0(x - f\rho X_0)}{fa^2} + \frac{Y_0(y - f\rho Y_0)}{fb^2} + \frac{Z_0(1 - \rho Z_0)}{c^2}\right).$$

Note that the inverse depth is a function of x and y. For simplicity, we have written ρ instead of $\rho(x,y)$. Similarly, taking derivatives with respect to y, we get

$$\frac{\partial \rho}{\partial y}(x,y) = \frac{1}{d(x,y)}\left(\frac{y - f\rho X_0}{f^2 b^2}\right).$$

We can compute the curl values for any (x,y) which corresponds to the projection of a surface point on the ellipsoid, using the computed gradient of ρ in the equation

$$\nabla \times V(x,y) = \left[\frac{\partial \rho}{\partial x}\cdot(v_3 y - f v_2) - \frac{\partial \rho}{\partial y}\cdot(v_3 x - f v_1)\right] - \frac{1}{f}\left[x\omega_1 + y\omega_2 + 2f\omega_3\right].$$

Knowing the geometry of the scene, for a given point (x,y), it is straightforward to determine if the point is on the image of the ellipsoid or the background plane. Recall that for all the other points (corresponding to a planar background), the translational part of the curl vanishes and only the linear, rotational part remains.

The curl will have values as graphed in Fig. (7.1). The true linear surface is seen in the background region (which is a frontal planar surface and hence gives the exact results); the distortion in the other regions is caused by the translational velocity in conjunction with the surface tilts.

If there is an error in the determination of the curl of the vector field, then the resulting surface shown in Fig. (7.1) will be perturbed. Estimates of the linear surface will most likely be based on averaged values over regions, and not on local gradients of the displayed surface; the least squares error estimate of the linear surface finds the best-fitting plane. Clearly, the best-fit linear surface to the surface shown in Fig. (7.1) will give an accurate estimate of the parameters of the unperturbed linear surface. For the surface shown, the estimated rotation parameters are accurate to the fourth decimal place. This example shows that accurate results can be obtained using the method, if there is a *good* distribution of the surface normals about the normal to the frontal planar surface.

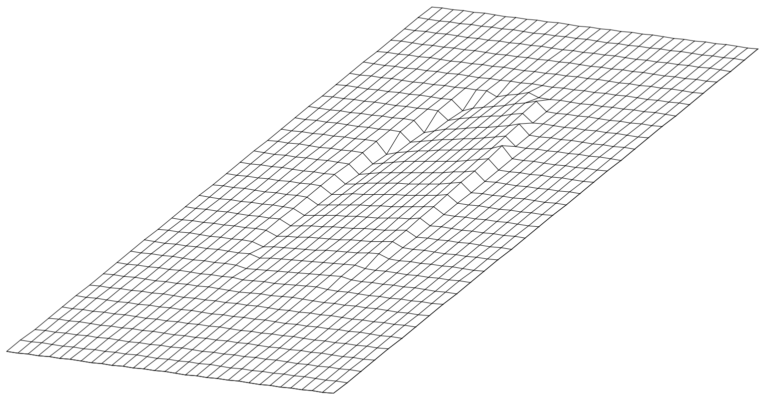


Figure 7.1: The curl of the flow field due to an imaged ellipsoid, with a sensor velocity of $\mathbf{T} = (0.3, 0, 2)$ and rotational velocity $\omega = (0.2, 0.1, 0.5)$.

We next apply the algorithm to a synthetically-generated vector flow field using a scene obtained from actual depth data. The depth data was made available by researchers at Michigan State University, obtained by scanning several objects using a "White Scanner." Fig. (7.2) shows a gray-scale interpretation of the raw range data of the scene. Depth data is not available throughout the image. Black pixels correspond to points of unknown depth.

The scene contains a sphere and a cylinder; the sphere, for example, has radius of about 4 focal units and its center is located 50 focal units from the sensor. The scene, it should be noted, is a "foveal region," occupying an image size of 0.15 by 0.25 focal units. A simulated flow field is computed using the depth values, as well as the projected image coordinate values, in conjunction with a camera motion of $\mathbf{T} = (5, 2, 20)$ focal units per second, and $\omega = (0.2, 0.1, 0.5)$ radians per second. An indication of the flow field is shown in Fig. (7.3).

An approximation to the curl of the flow field is computed using the flow field data (u and v) at five points in the neighborhood of each discrete sample, using the following formula:

$$\text{curl}(i, j) = \frac{v(i, j-1) - v(i, j+1)}{\delta x} - \frac{u(i-1, j) - u(i-1, j)}{\delta y},$$

where δx and δy are the sample distances in the x and y directions. The resulting curl of the flow field is displayed in Fig. (7.4). Data is plotted only at (i, j) positions where meaningful depth data is derived from Fig. (7.2).

Again, if there were noise in the computation of the flow field, this surface would be perturbed.

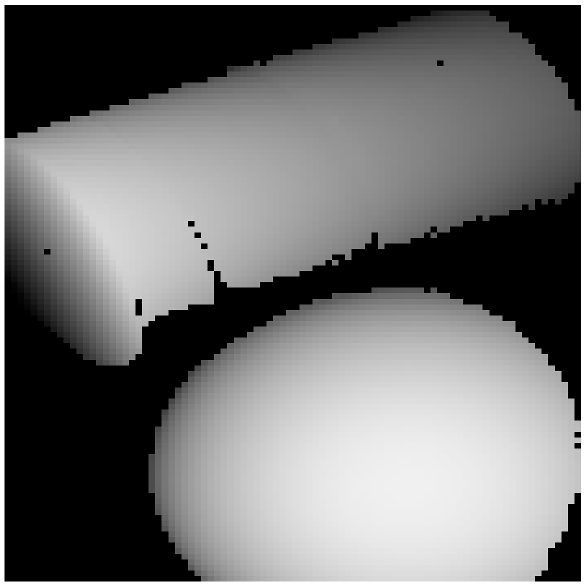


Figure 7.2: A depth image, where lighter values represent points with smaller depth. The scene consists of a cylinder and a sphere sitting in front of a flat plane. Only depth values are depicted, (the sampling rate with respect to x and y vary somewhat throughout the image).

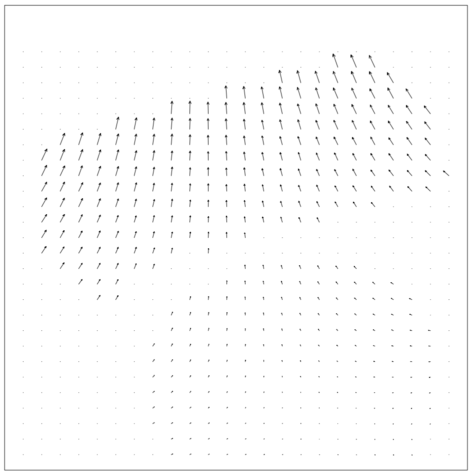


Figure 7.3: The flow field computed using depth values of Fig. (7.2), computed analytically using a camera motion of $\mathbf{T} = (5, 2, 20)$ focal units per second, and $\omega = (0.2, 0.1, 0.5)$ radians per second.

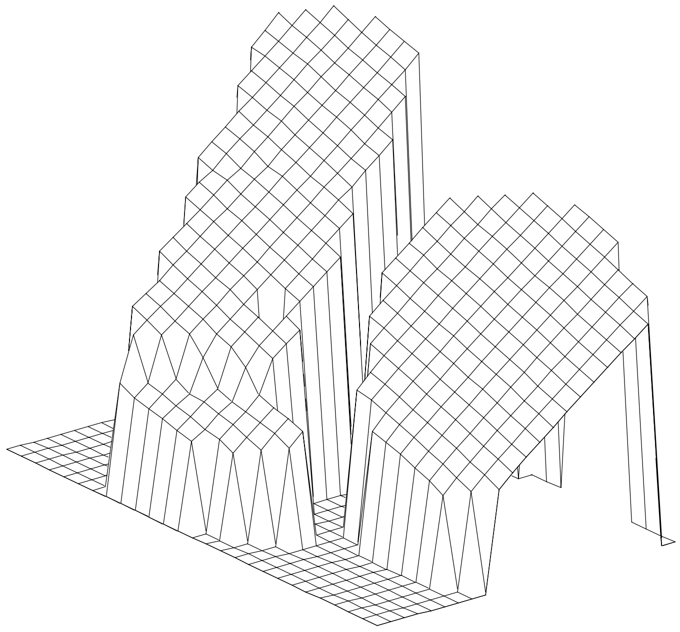


Figure 7.4: The approximate curl values of the vector flow field that result for the scene whose depth values are displayed in Fig. (7.2) imaged by a sensor whose translational velocity is $\mathbf{T} = (5, 2, 20)$, and rotational velocity is $\omega = (0.2, 0.1, 0.5)$. Data is only shown in regions where depth values are defined. Partial derivatives of the flow components are computed using a local regression, and the location of the curl values are moved to approximately the correct image location. The resulting best-fit planar surface predicts rotational parameters of $(0.2126, 0.1023, 0.4982)$.

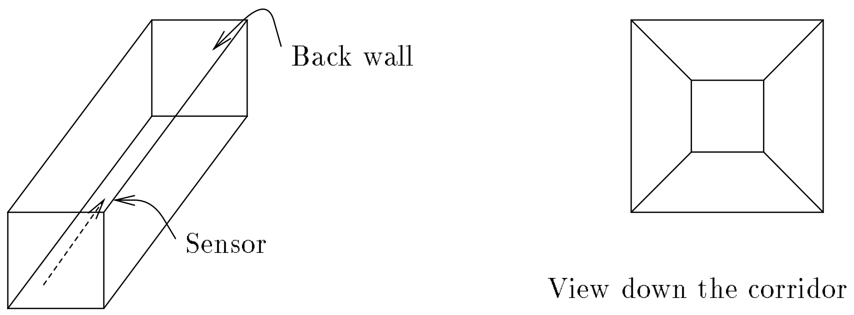


Figure 7.5: The corridor, and the view from the sensor

However, circulation values will give averages over regions, and thus will dampen the noise, providing it is uniformly distributed. It can be seen that the curl is approximately linear, as expected. The best-fit plane (without using any benefit of discarding outliers) is computed to the approximated curl data, over the regions where curl data has been computed, and is used to estimate the rotational parameters. The procedure yields the estimate $\omega \approx (0.2126, 0.1023, 0.4982)$, which quite accurately reflects the true rotational velocity.

Next, we consider a typical indoor scene. This is a synthetic image of a corridor, shown in Fig. (7.5).

The sensor moves with fixed velocity along the central line of the corridor, with a slight sideways drift. The view down the corridor is shown in Fig. (7.5). The translational velocity satisfies $v_1 = 1, v_2 = 0$, and $v_3 = 5$ focal units per second, and $\omega = (0.2, 0.1, 0.5)$ radians per second. The actual curl of the vector flow field is shown in Fig. (7.6). The curl values were computed analytically, using the piecewise planar depth functions. On the back wall of the corridor, the curl is precisely linear and is entirely due to the rotational parameters; on the sides, there is a deviation due to the translational parameters and the tilt of the surfaces. In some sense, a corridor is a worst-case scenario for the algorithm, due to the abundance of surface area with small R (see Eqn. (4.5)). However, clearly, the use of large-scale averages, or the best fit linear surface, is likely to lead to the correct parametric estimates. In this case, the regression fit of a surface to the observed data leads to the estimate $\omega = (0.2, 0.1379, 0.5)$ radians per second. The inaccuracy in the estimation of ω_2 is an illustration of the confounding parameters (in this case, between v_1 and ω_2) discussed in Section (6.2). Noise in sensing the curl values will perturb the surface shown in Fig. (7.6). However, if circulation values are calculated about circuits, then the surface will be locally averaged, accordingly. The accuracy in determining the rotational parameters will depend on the accuracy with which the parameters of the plane defined by the surface in the central region (where the back face of the corridor is imaged) are computed.

Next, we proceed to estimate the rotation values using circulation values, as opposed to using local curl values. A flow field of size 512×512 is generated for the corridor scene, for a translation of $\mathbf{T} = (1, 0, 5)$ focal units per second and a rotation of $\omega = (0.2, 0.1, 0.5)$ radians per second. It

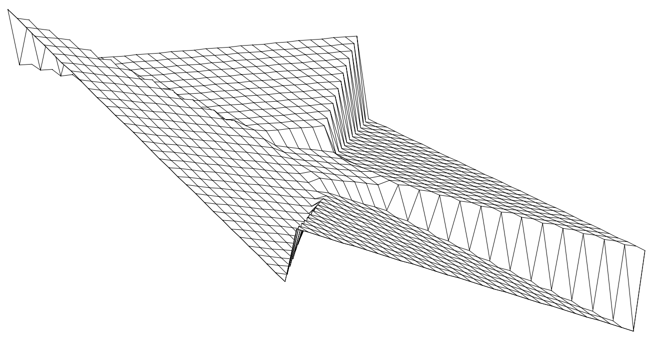


Figure 7.6: The curl of the flow field for the corridor scene

should be noted that high sampling rates favor a good approximation to the contour integrals by summation. We use square contours, resulting in a simple computation. The contour integral is simply the sum of the vertical components (with appropriate sign) along the two vertical sides of the square contour and the sum of the horizontal components along the two horizontal sides of the square contour. In the discrete implementation, one needs to multiply the appropriate component value at a pixel by the interpixel distance along the direction of integration, the approximation being that the component value is constant between pixels, akin to the rectangle rule for integration. About 960 overlapping square contours of side 200 were used for the corridor scene; the contours were chosen randomly within the 512×512 scene. The computed rotation, $(0.2007, 0.07263, 0.5023)$, is quite accurate. Notice again the effect of the horizontal translation on the estimate of ω_2. The circulation values have been plotted in Fig. (7.7). It can be seen that the surface approximates a linear surface.

The performance of the algorithm in the presence of synthetic noise is presented in Section (7.4). Results of applying the flow circulation algorithm to real data are presented in chapter 8. We will next consider the FOE search algorithms applied to synthetic data.

7.3 Translation estimation

In the previous chapter, we presented four methods to compute the translational parameters of a moving sensor. All the methods involve computing the circular component function whose

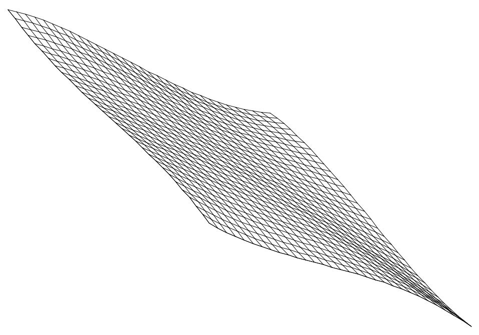


Figure 7.7: The circulation values computed from a synthetic flow field for the corridor scene. It is approximately a linear surface.

definition is repeated here:

$$U_{(x_0,y_0)} = V(x,y) \cdot (-y + y_0, x - x_0). \tag{7.2}$$

Note that (x_0, y_0) is a candidate for the FOE. The steps are as follows, assuming that the optical flow field has been computed:

1. For each choice of (x_0, y_0),
   - Compute the circular component function $U_{(x_0,y_0)}(x,y)$.
   - Cancel the rotational component by convolution or projection, if the method requires it.
   - Compute the error value to be the norm of the residual function.
2. Compute the FOE as the location of the minimum of the error surface.

For the center-surround kernel method, the quadratic polynomial projection method, and the NCC method, we need to choose only six different values for (x_0, y_0), to completely determine the minimum of the error surface. This is due to the quadratic nature of the error surface for these methods. For the Subspace Projection method, we need to do a search on the image domain for the FOE. In this chapter, we will compute the error surface for each of the methods. Thus, the emphasis will be on displaying the error surface and pointing out that the FOE is indeed located at the minimum of the error surface. In the next chapter, we will concentrate on obtaining the FOE by means of procedures that exploit the known shape of the error surface.

Certain implementation details are relevant here. For the experiments, the flow field is computed on a discrete grid. We choose the focal length to be unity, and the image to be of size 2 by 2 focal units. This corresponds to a field of view of ninety degrees. This is a rather wide field of view.

To demonstrate the methods, we will make use of the synthetic flow field generated for the corridor scene in Fig. (7.5).

7.3.1 Center-surround kernel method

Recall that the Center-surround kernel method is based on the observation that derivatives of the Laplacian of the circular component function are zero at the correct FOE. This is because at the FOE, there is no contribution to the circular component function from the translational part of the flow field, and the contribution from the rotational part is a quadratic polynomial that will be eliminated by any derivative of a Laplacian. The use of a convolution kernel was suggested earlier (see Section (5.3.1)). For use with discrete images, we use the difference (along the horizontal direction) of discrete Laplacians as the kernel K, which is represented up to a constant of proportionality by the matrix:

$$\begin{bmatrix} 1 & 3 & -3 & -1 \\ 4 & -24 & 24 & -4 \\ 1 & 3 & -3 & -1 \end{bmatrix}. \tag{7.3}$$

If this kernel is applied to point evaluates on a discrete grid of the circular-component function $U_{(x_0,y_0)}(x,y)$ with (x_0,y_0) at the focus of expansion, then, despite the fact that the discrete Laplacian is an approximation to the continuous Laplacian, the result should be a zero function. For every candidate (x_0,y_0), we compute $U_{(x_0,y_0)}(i,j)$ and then convolve it with the filter K, yielding $\Phi_{(x_0,y_0)}$. We know that for the correct FOE, this function $\Phi_{(x_0,y_0)}$ is identically zero. To test this, we compute

$$E(x_0,y_0) = \sum_i \sum_j \left(\Phi_{(x_0,y_0)}(i,j)\right)^2.$$

We can generate the flow field for the case where the camera is moving along the corridor (see Fig. (7.5)) with a translational velocity $\mathbf{T} = (-4,-2,16)$ and with rotational parameters $\omega = (0.1, 0.2, 0.035)$. The flow field is shown in Fig. (7.8). Each grid point is treated as the candidate (x_0,y_0) and the error function is computed. A plot of the function $E(x_0,y_0)$ is shown in Fig. (7.9). The minimum of the function, which is precisely zero, occurs at the true focus of expansion $(-0.25,-0.125)$. As can be seen, the error function is nonzero elsewhere, and has a quadratic behavior.

As a second example, we use the depth scene of Fig. (7.2) to compute a vector flow field, using translational and rotational parameters of (5, 2, 20) and (0.2, 0.1, 0.5) respectively. Then we apply the same center-surround kernel method to obtain the $\Phi_{(x_0,y_0)}$ functions. The error function is presented as a contour plot together with a field denoting the gradient of E, in order to suggest

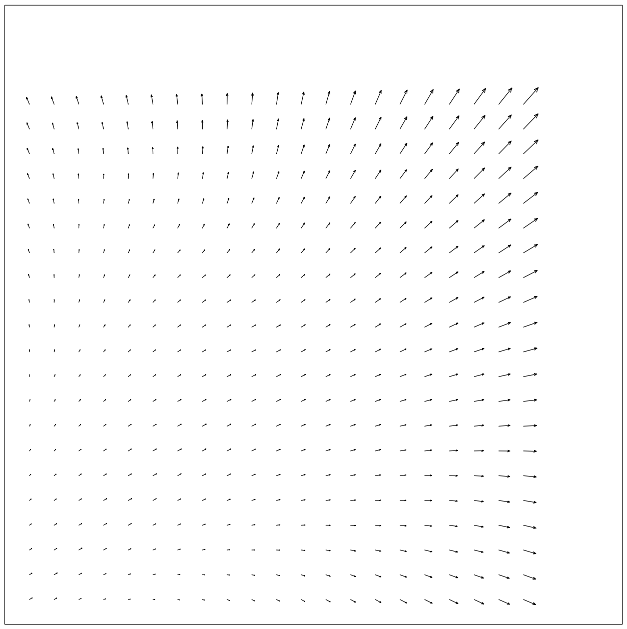


Figure 7.8: The flow field for the corridor scene

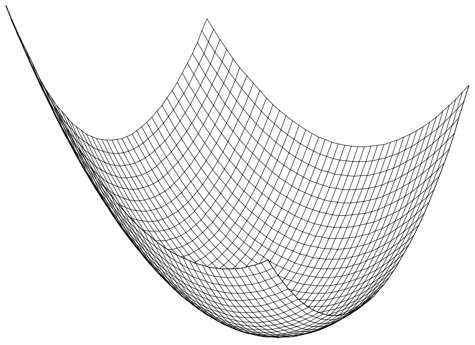


Figure 7.9: The error surface for the corridor scene

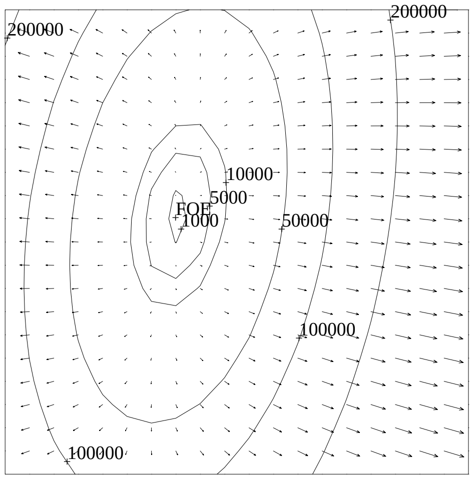


Figure 7.10: The contour and gradient of the error function $E(x_0, y_0)$ using the center-surround kernel method, computed for the flow field induced when the depth image of Fig. (7.2) is imaged by a sensor with translational velocity of (5, 2, 20) and rotational velocity of (0.2, 0.1, 0.5). The function has the minimum at the focus of expansion at (0.25, 0.1).

the possibility of a gradient descent search for the focus of expansion. In this case, we assume that there is a fixed-depth background behind the objects in Fig. (7.2), so that the circular-component functions can be defined over the entire image. Once again, the error function correctly zeros out at the focus of expansion, at (0.25, 0.1).

For the corridor scene, we used the Laplacian kernel (see Eqn. (5.8) convolution followed by a variance computation. The variance is simply the mean square deviation from the mean. The discrete Laplacian kernel used is:

$$\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}. \tag{7.4}$$

The results are shown in the contour plot in Fig. (7.11). The FOE is located at the correct position.

7.3.2 Quadratic polynomial projection method

The center-surround kernel method will fail to work under certain pathological situations; these are situations where the convolution eliminates not only the contribution from rotation, but also that from translation. This happens when the whole field of view is occupied by a planar surface or, in general, surfaces with depth functions that are harmonic. However, since the method works adequately for the examples given in the previous section, we would expect the other two methods

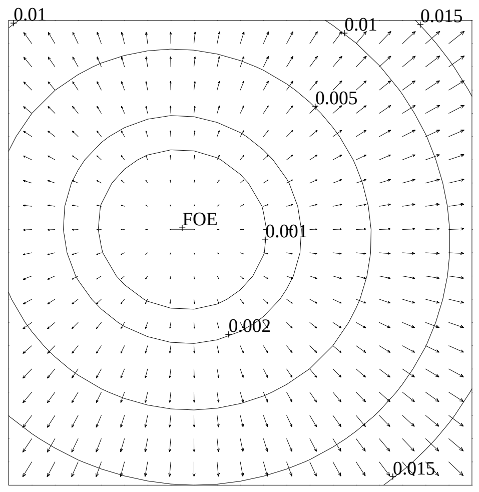


Figure 7.11: The contour and gradient of the error function $E(x_0, y_0)$ using the center-surround kernel method, by using a Laplacian kernel followed by a variance computation, for the flow field of the corridor scene imaged by a sensor with translational velocity of (5, 2, 20) and rotational velocity of (0.2, 0.1, 0.5). The function has the minimum at the focus of expansion at (0.25, 0.1).

also to work. The main issue is whether the accuracy of the localization of the focus of expansion suffers due to the enhanced specificity of the determination of the quadratic polynomial nature of the circular component functions.

For the quadratic polynomial projection method, we use orthogonal polynomials in two variables up to degree two on a discrete domain of finite size. Rather than using a discrete approximation to the continuous theory presented in Section (5.3.3), we use basis functions that are defined and are orthonormal on a discrete grid of size $2n + 1$ by $2n + 1$, with interpixel sampling distance d. We assume that the grid is indexed by (i, j), where $-n \leq i, j \leq n$. The discrete basis functions are derived by choosing a known set of functions that span the space (in this case, the functions 1, x, y, xy, x^2 and y^2), and then orthogonalizing them using a standard method like the Gram-Schmidt process [64]. Note that in this finite discrete domain, $x = j \cdot d$, and $y = i \cdot d$, where $-n \leq i, j \leq n$. The resulting basis functions are

$$\phi_1(i,j) = \frac{1}{d(2n+1)},\ \phi_2(i,j) = \frac{x}{\sqrt{D}},\ \phi_3(i,j) = \frac{y}{\sqrt{D}},$$

$$\phi_4(i,j) = \frac{x \cdot y}{D},\ \phi_5(i,j) = \frac{x^2 - C}{\sqrt{K}},\ \phi_6(i,j) = \frac{y^2 - C}{\sqrt{K}},$$

where D, C, and K are constants that depend only on n and d:

$$D = \frac{d^2 n(n+1)(2n+1)^2}{3}, \; C = \frac{n(n+1)}{3},$$

$$K = (2n+1) \cdot d^2 \left[2d^2 \sum_{i=1}^{n} i^4 + (2n+1)C^2 - \frac{2Cn(n+1)(2n+1)}{3} \right].$$

The constants have been chosen such that the ϕ_i functions have unit norm and are orthogonal with respect to the inner product

$$< f, g > = \sum_{i=-n}^{n} \sum_{j=-n}^{n} f(i,j) g(i,j). \tag{7.5}$$

Note that we use a discrete (pointwise) inner product as opposed to a Hermite inner product. This is possible because we are dealing with a finite discrete domain of small size. Large or unbounded domains require the Hermite inner product because the inner product in Eqn. (7.5) could be unstable or unbounded over such domains. We compute

$$U_{(x_0,y_0)}(i,j) - \sum_{k=1}^{6} < U_{(x_0,y_0)}, \phi_k > \phi_k(i,j).$$

The squared norm of this function, based on the inner product in Eqn. (7.5), is the error measure $E(x_0, y_0)$.

We compute the error function $E(x_0, y_0)$ for the flow field created when the camera moves along the corridor (see Fig. (7.5)) with a translational velocity of $(5, 2, 20)$ and with rotational parameters $(0.1, 0.025, -0.05)$. The function $E(x_0, y_0)$ is shown in Fig. (7.12).

We correctly determine the focus of expansion as the zero of the error function shown in Fig. (7.12).

7.3.3 Subspace projection method

The subspace projection method involves projecting the circular-component function on to the space spanned by three quadratic polynomials $\phi_1(x,y)$, $\phi_2(x,y)$ and $\phi_3(x,y)$ of known form:

$$\begin{aligned}
\phi_1(x,y) &= -\frac{x_0}{f} y^2 + \frac{y_0}{f} xy + fx - fx_0, \\
\phi_2(x,y) &= -\frac{y_0}{f} x^2 + \frac{x_0}{f} xy + fy - fy_0, \\
\phi_3(x,y) &= -y^2 - x^2 + x_0 x + y_0 y.
\end{aligned}$$

The coefficients of the projection are a_1, a_2 and a_3. For the discrete case, the basis functions ϕ_k, for $k = 1, 2, 3$ are simply point evaluates (on the discrete grid) of the functions defined by the above equations. We compute the inner product matrix Q explicitly instead of using the analytical

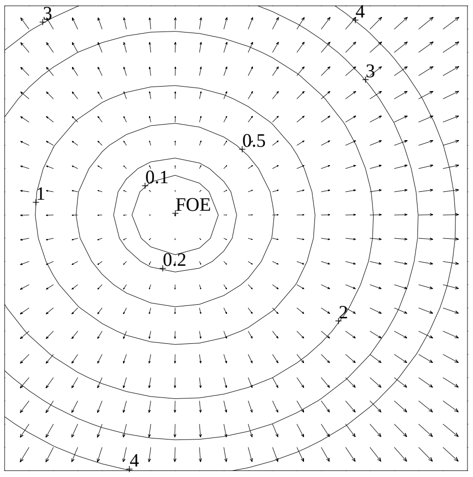


Figure 7.12: Contour and gradient plot of the error function $E(x_0, y_0)$ obtained using the quadratic polynomial projection method for the image flow field induced by a translational velocity of $(5, 2, 20)$ and rotational velocity of $(0.1, 0.025, -0.05)$ for the corridor scene.

expressions given in Section. (5.3.4), again using the discrete inner product in Eqn. (7.5). Recall that the entry $q_{i,j}$ is the inner product $< \phi_i, \phi_j >$. All the entries of this symmetric matrix Q can be computed by multiplying pairwise, point evaluates of the functions ϕ_k and summing up over the discrete grid. Here, we are using the inner product definition given in Eqn. (7.5). We also compute the inner products $< U_{(x_0,y_0)}, \phi_k >$. We solve for a_1, a_2 and a_3 by solving the "normal equations" (Eqn. (5.16)), repeated here:

$$Q \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} < U_{(x_0,y_0)}, \phi_1 > \\ < U_{(x_0,y_0)}, \phi_2 > \\ < U_{(x_0,y_0)}, \phi_3 > \end{bmatrix}. \tag{7.6}$$

Once we have a_1, a_2 and a_3, we can compute the error function

$$E(x_0, y_0) = ||U_{(x_0,y_0)} - \sum_{i=1}^{3} a_i \phi_i||^2.$$

Thus $E(x_0, y_0)$ gives the norm of the residual after $U_{(x_0,y_0)}$ is differenced with its projection on to the subspace spanned by the basis functions (which are quadratic polynomials) defined at (x_0, y_0). Again, the norm is based on the inner product defined in Eqn. (7.5). We determine the correct focus of expansion for the case of a camera whose motion is the same as described in Fig. (7.12). A contour plot of $E(x_0, y_0)$ is shown in Fig. (7.13). Again, the FOE is correctly located.

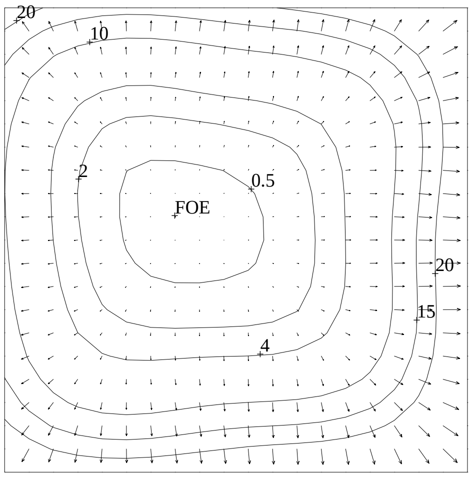


Figure 7.13: Contour and gradient plot of the error function obtained using the subspace projection method, using the scene and motion described in Fig. (7.12).

7.4 Performance with noisy data

In this section, we present results of applying the algorithms to synthetic flow field data corrupted by noise. For this purpose, we adopt two different noise models. We present the noise models first, and then the experimental results.

7.4.1 Noise models

Two noise models are considered. The first one is additive uniform noise. The two (horizontal and vertical) components of each flow vector are each perturbed by the addition of a uniform random noise. The magnitude of this added noise is modulated by the average value of the corresponding vector component (averaged over the entire flow field). The overall procedure is as follows:

1. Synthesize the flow field data for a suitable scene (we do it for two different kinds of scenes: random depth data and the corridor scene). Let the horizontal and vertical components be $h_{i,j}$ and $v_{i,j}$ respectively, at the image location (i, j).

2. Compute the averages h_a and v_a by summing over the entire flow field (of size $n \times n$):

$$h_a = \sum_{i,j} h_{i,j}, \qquad v_a = \sum_{i,j} v_{i,j}.$$

3. For each image location (i, j), perturb each component by a random value from a uniform distribution, modulated by (a fraction of) the average values:

$$h'_{i,j} = h_{i,j} + f \cdot h_a \cdot r^h_{i,j}, \qquad v'_{i,j} = v_{i,j} + f \cdot v_a \cdot r^v_{i,j},$$

   where f is a fraction (a parameter in the experiments), and r^h and r^v are random values distributed uniformly in the interval $[-0.5, 0.5]$.

By varying the fraction f, the flow field is perturbed to varying extents and the algorithms are tested on these flow fields. For each value of f, each experiment is repeated several times (the exact number is reported along with the results) and the error is averaged over the several runs. The additive uniform noise model has been used before [24]. So, this method serves to compare the results with previous ones.

The second model attempts to account for the way optical flow computations work. A majority of the algorithms to compute optical flow use some form of regularization (smoothing), to overcome the aperture problem. Because of this smoothing, the resulting optical flow, even though noisy, tends to have noise components that are not independent identically distributed (i.i.d.), but are correlated. To mimic this non-i.i.d. behavior, we add two different noise components to the synthetic flow field. First, we add Gaussian distributed random values to the magnitude of the flow field vectors. Observation of error in real flow fields suggests that Gaussian distribution is a reasonable approximation [7,14]. Next, we add "smoothed" random values to the phase angle of the flow field vectors, resulting in a systematic distortion of the flow field. Thus, the noise modulation consists of two steps:

1. Let $M_{i,j}$ and $P_{i,j}$ be the magnitude and the phase angle of the (synthetic) optical flow vector at image location (i, j). Add Gaussian random noise to the magnitude:

$$M'_{i,j} = M_{i,j} \cdot (1 + \alpha \cdot r_{i,j}),$$

   where $r_{i,j}$ is Gaussian randomly distributed with zero mean and unit variance ($N(0, 1)$), and α is a fraction determining the extent of the noise modulation. Note that the additive noise is proportional to the magnitude of individual flow vectors (as opposed to the *average* value in the previous noise model).

2. Modulate the phase angles by a smoothed noise field:

$$P' = P + \beta \cdot (r * s),$$

   where r is randomly distributed noise with distribution $N(0, 1)$, β is a fraction determining the extent of the noise modulation, s is a smoothing filter, and "$*$" indicates the convolution operation.

We generate synthetic flow fields for both the corridor scene and random depth data. The corridor scene is as described in Section 7.2. The random depth data is similar to the one used in [24]; the depth values are randomly distributed uniformly between 2 units and 4 units, the unit of measure being the focal length. The flow is computed over a grid of size 21×21. We next present the results of applying the algorithms to the flow fields with noise.

7.4.2 Experimental results with noisy data

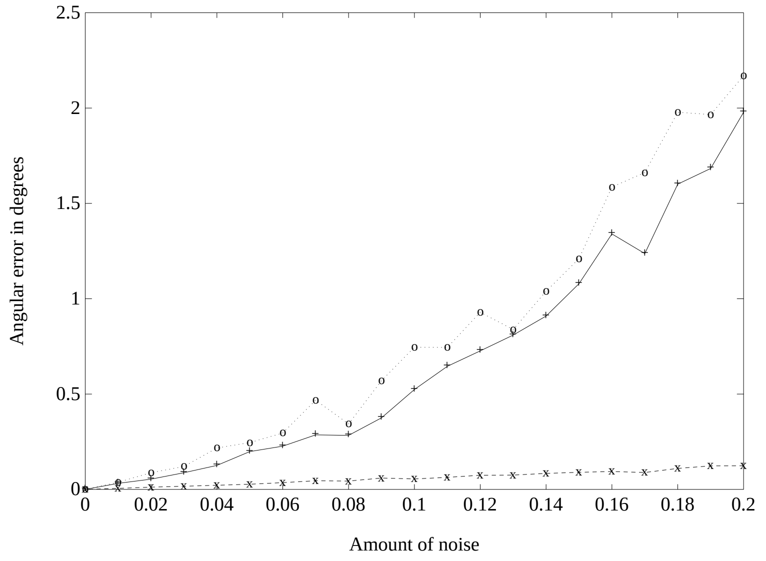
We begin with experiments on the flow computed using random depth data. Depth values between 2 and 4 units were generated by sampling a uniform distribution. Recall (see Section 6.3) that depth variations favor the FOE search algorithms. We synthesized flow fields using a translation of $v_1 = 0.6$, $v_2 = 0$, and $v_3 = 0.8$ units, and zero rotational velocity. The results are shown in Fig. (7.14). The plots show the variation of the angular error (angle between the direction of actual translation and the direction computed) with respect to the extent of noise, measured by the fraction that modulates the noise amplitude.

For the uniform noise model, we note that the error is consistently the highest for the center-surround kernel method, less for the quadratic polynomial projection method and the least for the NCC algorithm. However, even for the value of $f = 0.2$, all the methods have errors below 2.5 degrees; this is quite accurate. For the smoothed Gaussian noise model, the worst performance is by far the quadratic polynomial projection method, followed by the center-surround kernel method and the NCC algorithm. Systematic distortions seem to be unfavorable for the quadratic polynomial projection method. Also, all the methods have larger errors for this noise. Note that in both noise situations, the NCC algorithm performs very well; it is to be expected, because there is no rotation, and this is the best situation for the NCC algorithm.

In the second experiment, the parameters are the same as in the first, except that the rotation is $[0.0081, -0.0116, -0.0168]$ radians/sec (a typical angular velocity taken from an actual motion sequence data). The results are shown in Fig. (7.15). As expected, the NCC algorithm has a high error, even in the noise-free case (corresponding to $f = 0$ in the plots), but hardly degrades with noise. The behavior of the other two methods is similar to that in the first experiment. In both experiments, we see that the degradation with noise (for the noise models considered here) is graceful. The error values are very low for the additive noise case, as in the zero angular velocity situation. The errors are higher for the smoothed Gaussian noise, as before.

Next, we consider the corridor scene (Section 7.2). For the first experiment, we use a translation of $[5, 2, 20]$ units and zero angular velocity. The field of view is again about 30 degrees. The error plots are in Fig. (7.16). The NCC algorithm has the best performance, and the quadratic polynomial projection method has a performance comparable to the random depth data. However, the center-surround kernel method has a much higher error rate, possibly due to the lack of large depth variations.

Finally, we repeat the experiment for the case where the rotation is $[0.0081 - 0.0116 - 0.0168]$.



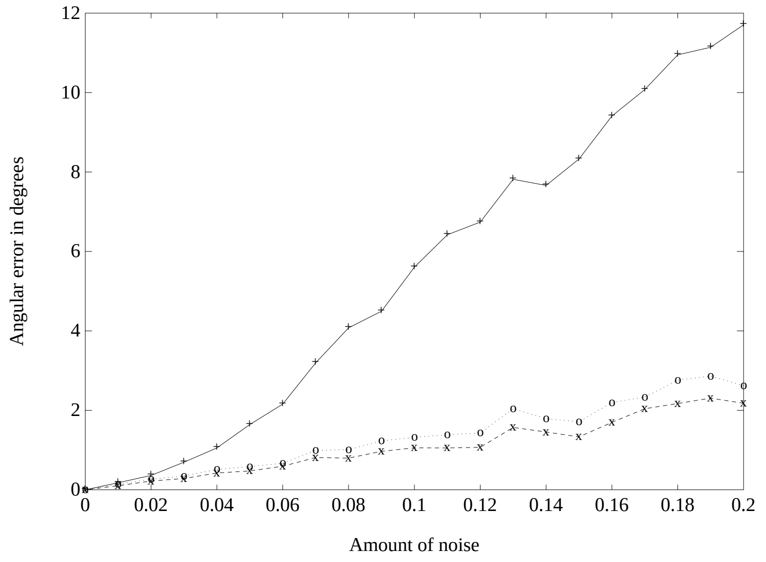
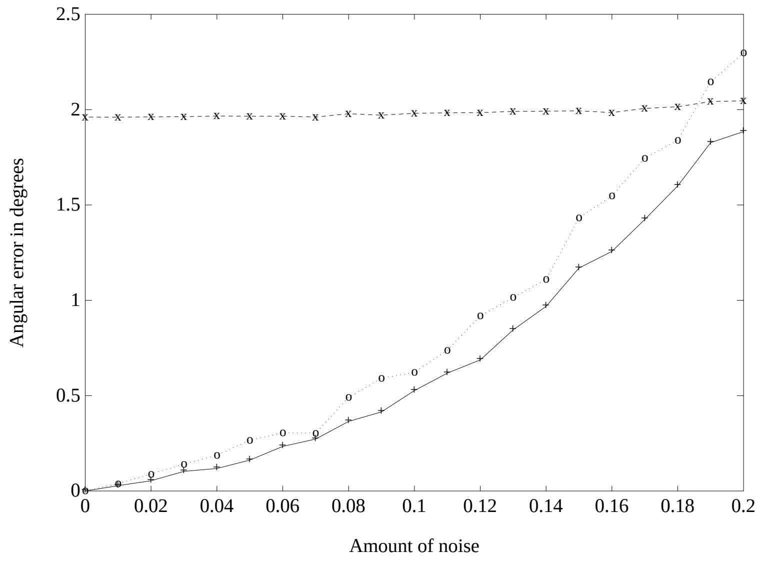


Figure 7.14: Error plots for the flow fields synthesized using random depth data, with translation $(0.6, 0.0, 0.8)$ and zero angular velocity. The depth points were randomly distributed (uniformly) between 2 and 4 units, the unit being the focal length. The field of view is about 30 degrees. The plot on the top corresponds to the additive uniform noise model and the one on the bottom corresponds to the smoothed Gaussian noise model. The dotted line corresponds to the center-surround kernel method, the solid line for the quadratic polynomial projection method, and the dashed line for the NCC algorithm. Each data point is the result of averaging the error over 50 runs.



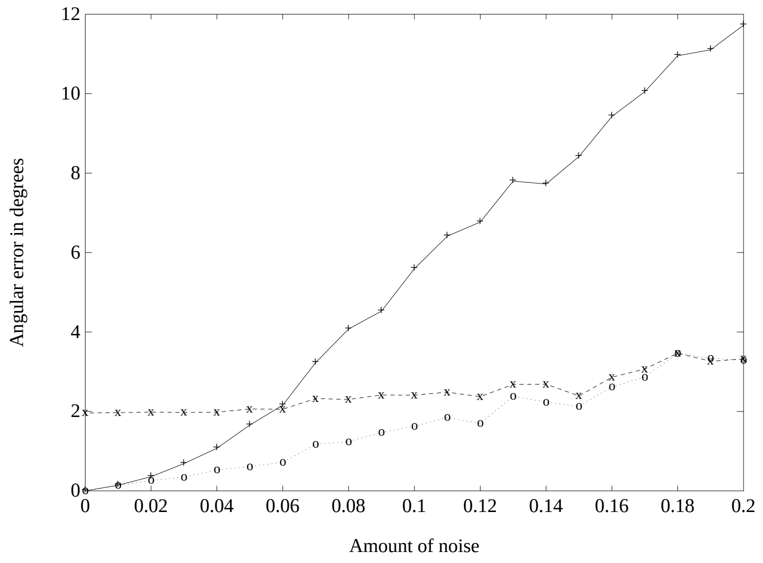
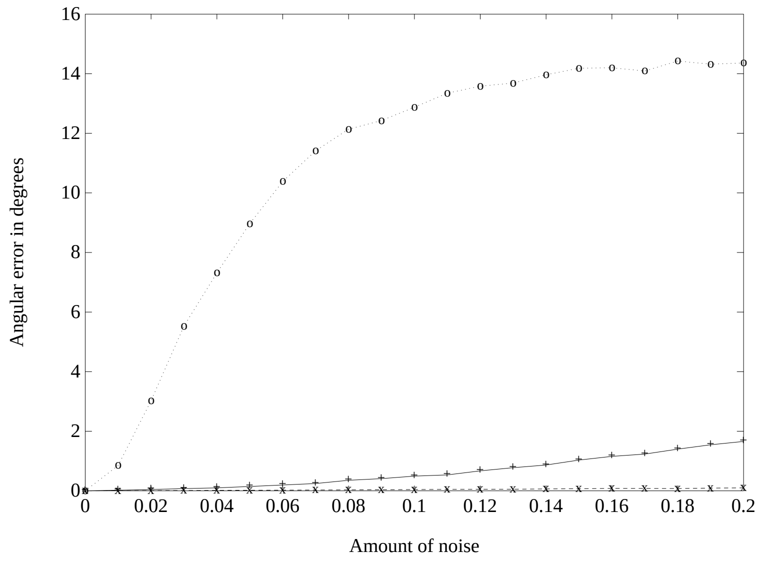


Figure 7.15: Error plots for the flow fields synthesized using random depth data, with translation $(0.6, 0.0, 0.8)$, and an angular velocity $(0.0081, -0.0116, -0.0168)$ radians/sec. The depth points were randomly distributed (uniformly) between 2 and 4 units, the unit being the focal length. The field of view is about 30 degrees. The plot on the top corresponds to the additive uniform noise model and the one on the bottom corresponds to the smoothed Gaussian noise model. The dotted line corresponds to the center-surround kernel method, the solid line for the quadratic polynomial projection method, and the dashed line for the NCC algorithm. Each data point is the result of averaging the error over 100 runs for the top plot and over 50 runs for the bottom plot.



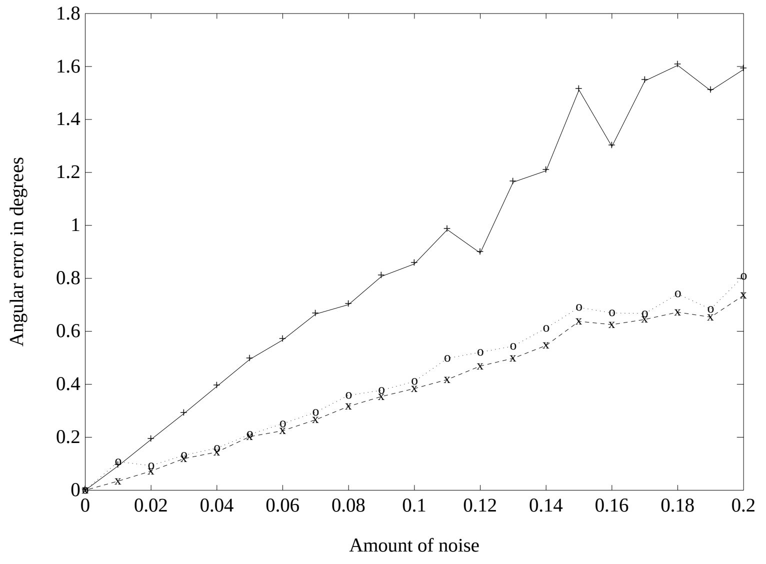


Figure 7.16: Error plots for the flow fields synthesized for the corridor scene, with a translation of $(5, 2, 20)$ and zero angular velocity. The field of view is about 30 degrees. The plot on the top corresponds to the additive uniform noise model and the one on the bottom corresponds to the smoothed Gaussian noise model. The dotted line corresponds to the center-surround kernel method, the solid line for the quadratic polynomial projection method, and the dashed line for the NCC algorithm. Each data point is the result of averaging the error over 100 runs.

The results are shown in Fig. (7.17). As before, the NCC algorithm has an error even for the noise-free case because of the presence of rotation, but there is hardly any degradation with noise. The center-surround kernel and the quadratic polynomial projection methods have errors increasing with noise, but exhibiting a graceful degradation.

We now present results from the *flow circulation algorithm* obtained using noisy synthetic flow fields. Unlike the translational parameters, the rotational parameters can be estimated without a scale factor; i.e., we can compute all the three rotational parameters. Recall that the 3-vector ω representing the rotational parameters is equivalent to an axis-magnitude representation; i.e., a unit vector A_ω in the direction of ω is the axis and the magnitude M_ω of ω is the magnitude of the rotational velocity. We represent error in the computed parameters by graphing the error in the estimate of the direction and the percentage error in the magnitude. In all the cases, the flow is computed over a grid of size 51×51, and square contours of side 20 pixels were used in the flow circulation algorithm.

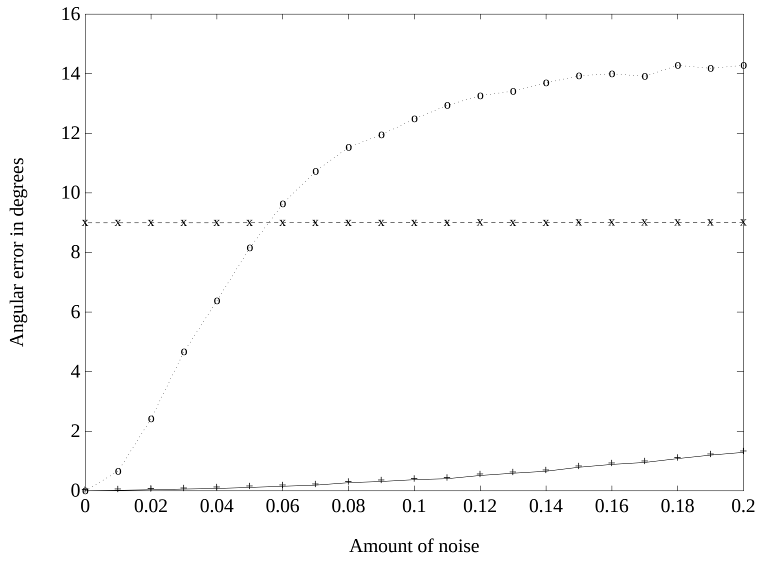
First, we consider the additive uniform noise model. This is a zero translation case (a favorable case for the flow circulation algorithm). The rotation is $(0.2, 0.1, 0.5)$. The results are shown in Fig. (7.18). The estimated parameters are quite accurate for the no-noise situation. The only error is in the magnitude, due to the discrete approximation to the contour integral. The error is found to increase gradually with the increase in added noise. In this case, the angular error is within 6 degrees for up to 20% perturbation, and the error in magnitude is within 15% in the worst case.

The zero translation case with the noise added using the smoothed Gaussian random noise model is shown in Fig. (7.19). The errors in this case are much higher, indicating that systematic noise is unfavorable to the flow circulation algorithm. However, the error increase with increasing noise is still gradual, asserting graceful degradation of performance.

The errors are expected to increase when the translation is non-zero. Indeed, this can be seen in the plots in Fig. (7.20), for the additive uniform noise model. Here the translation is (0.5, 0.0, 2.0). The rotation remains the same at $(0.2, 0.1, 0.5)$. Notice the nonzero error (and higher compared to the non-zero translation case) of both the magnitude and the direction for the no-noise situation. The maximum error in direction is within 8 degrees of angle and the maximum error in magnitude is 16%. The error, however, does not climb significantly with noise, indicating good tolerance of the algorithm to this kind of additive uniform noise.

Fig. (7.21) contains the results for the same motion parameters with the noise from the smoothed Gaussian random noise model. As before, the systematic noise model results in higher error, but the performance still degrades gracefully with noise.

This presentation of the performance of the algorithms under no-noise and noisy situtations indicates that with flow fields of reasonable accuracy (less than 20% noise), one can expect to get quite accurate results.



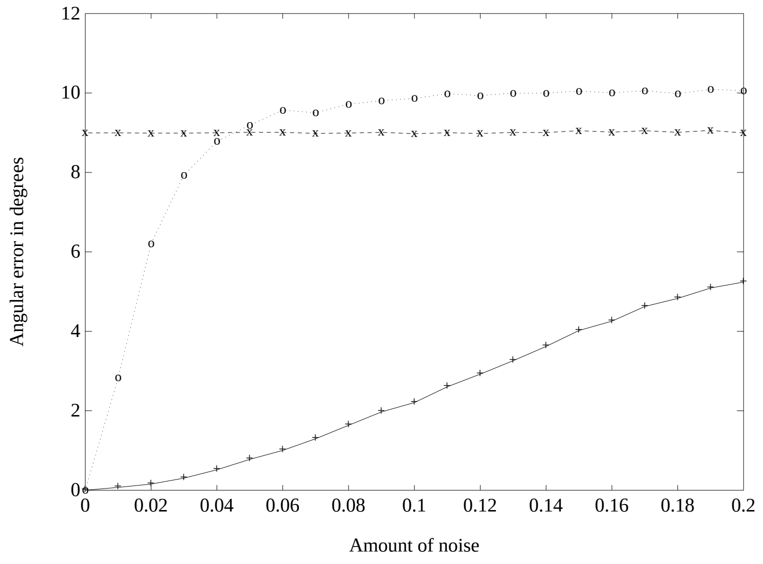
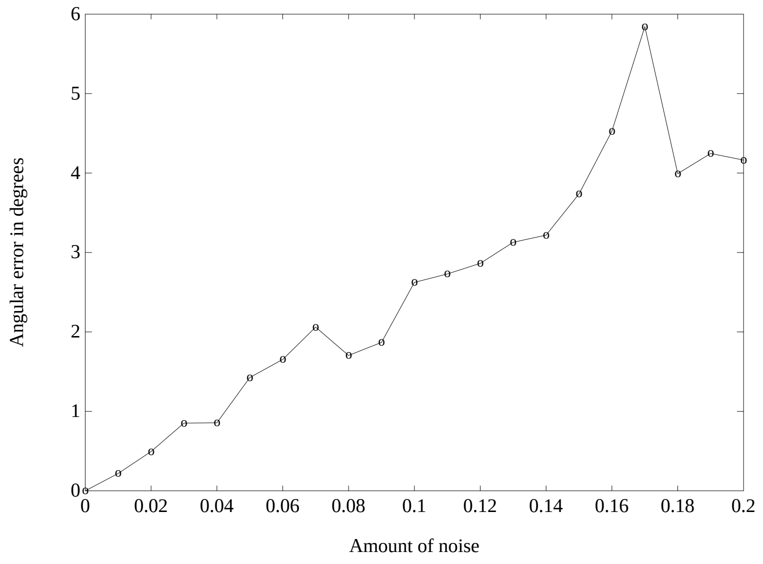


Figure 7.17: Error plots for the flow fields synthesized for the corridor scene, with a translation of $(5, 2, 20)$ and an angular velocity of $(0.0081, -0.0116, -0.0168)$ radians/sec. The field of view is about 30 degrees. The plot on the top corresponds to the additive uniform noise model and the one on the bottom corresponds to the smoothed Gaussian noise model. The dotted line corresponds to the center-surround kernel method, the solid line for the quadratic polynomial projection method, and the dashed line for the NCC algorithm. Each data point is the result of averaging the error over 100 runs.



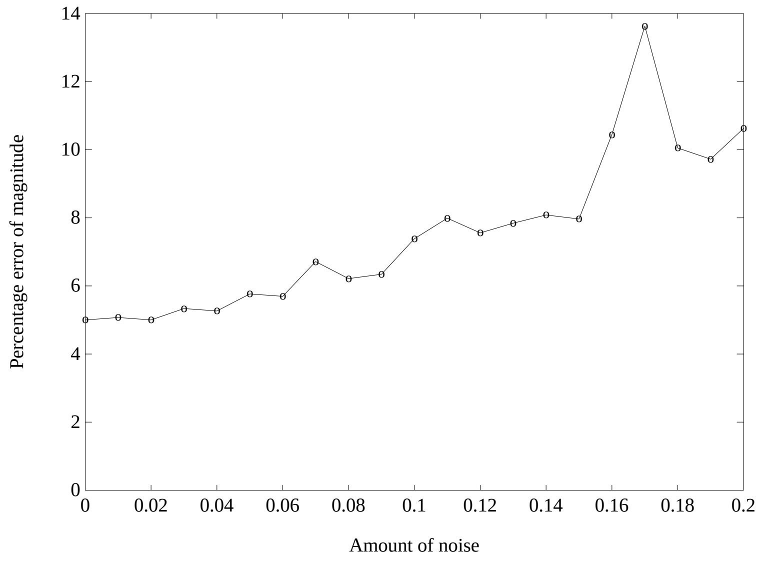
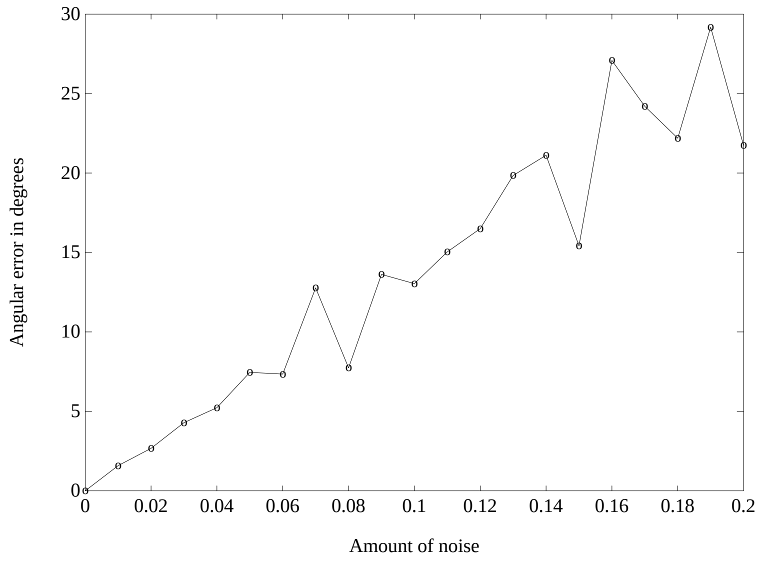


Figure 7.18: Error plots for the flow fields synthesized for the corridor scene, with no translational velocity and an angular velocity of $(0.2,\ 0.1,\ 0.5)$ radians/sec. The field of view is about 30 degrees. The noise is additive uniform. Each data point is the result of averaging the error over 10 runs. The numbers on the abscissa indicate the fraction of the average velocity, used to modulate the random noise. The top plot shows the error in the estimation of the axis of rotation, and the bottom plot shows the percentage error in the estimation of the rotational velocity magnitude.



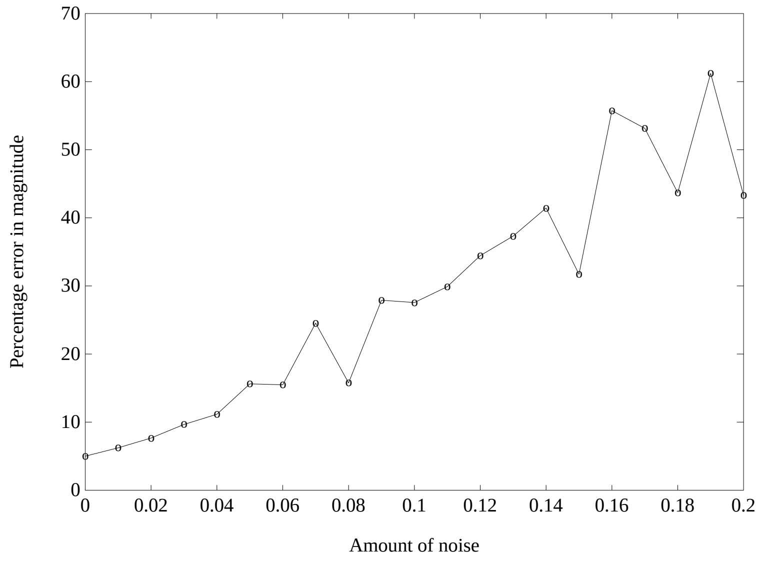
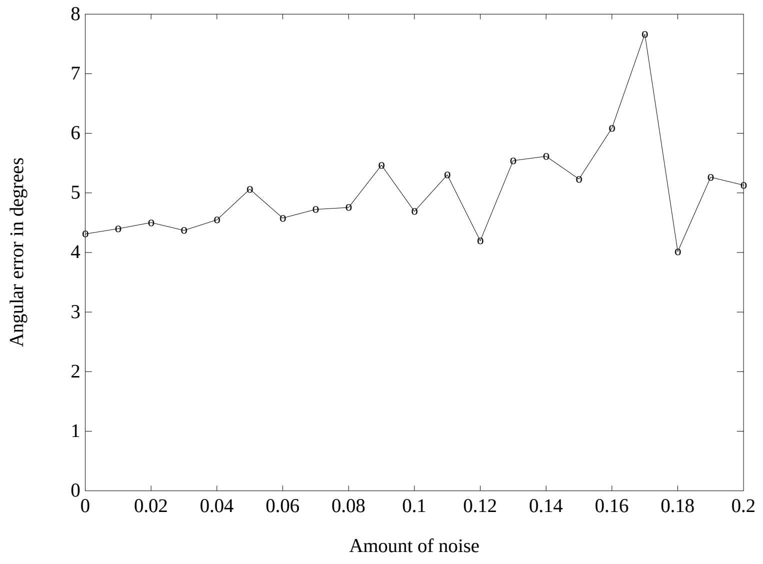


Figure 7.19: Error plots for the flow fields synthesized for the corridor scene, with no translational velocity and an angular velocity of $(0.2,\ 0.1,\ 0.5)$ radians/sec. The field of view is about 30 degrees. The noise is smoothed Gaussian. Each data point is the result of averaging the error over 10 runs. The numbers on the abscissa indicate the fraction of the velocity used to modulate the random noise. The top plot shows the error in the estimation of the axis of rotation, and the bottom plot shows the percentage error in the estimation of the rotational velocity magnitude.



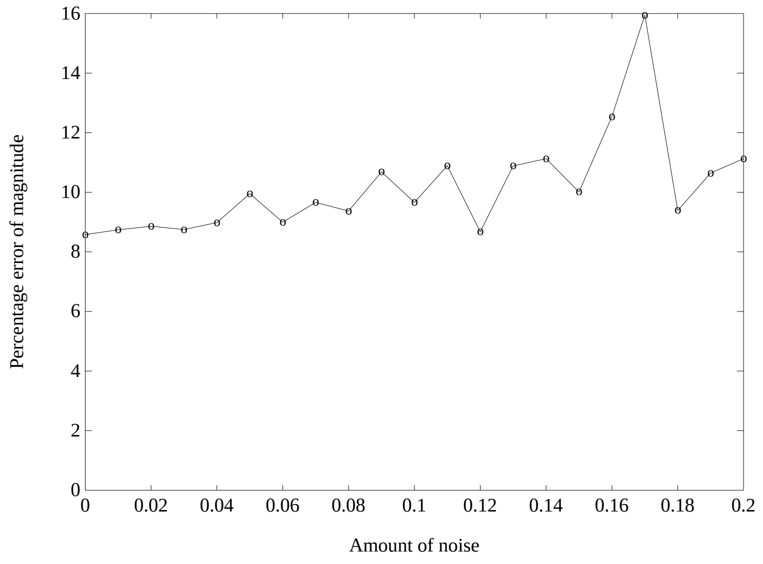
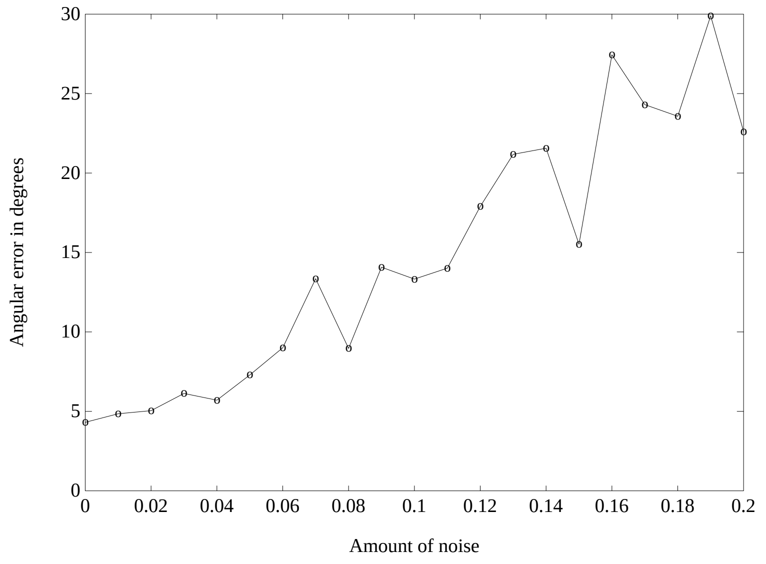


Figure 7.20: Error plots for the flow fields synthesized for the corridor scene, with translation $(0.5,\ 0.0,\ 2.0)$ and an angular velocity of $(0.2,\ 0.1,\ 0.5)$ radians/sec. The field of view is about 30 degrees. The noise is additive uniform. Each data point is the result of averaging the error over 10 runs. The numbers on the abscissa indicate the fraction of the average velocity, used to modulate the random noise. The top plot shows the error in the estimation of the axis of rotation, and the bottom plot shows the percentage error in the estimation of the rotational velocity magnitude.



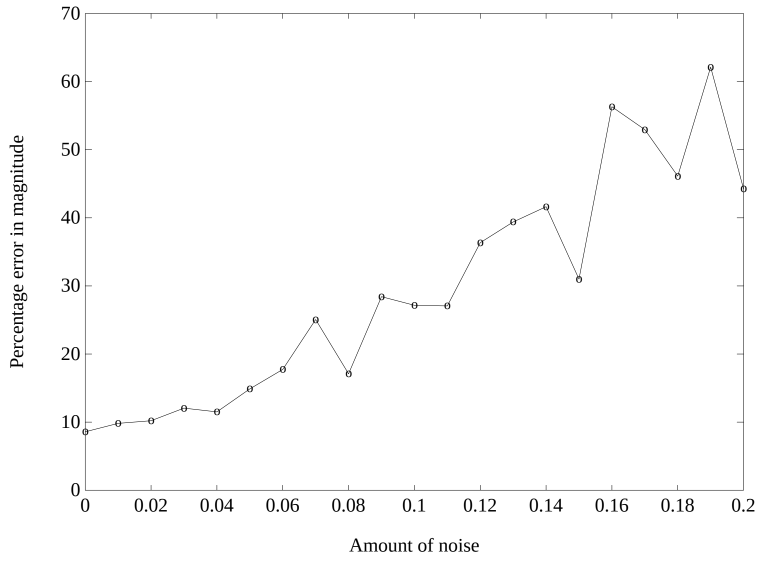


Figure 7.21: Error plots for the flow fields synthesized for the corridor scene, with translation $(0.5,\ 0.0,\ 2.0)$ and an angular velocity of $(0.2,\ 0.1,\ 0.5)$ radians/sec. The field of view is about 30 degrees. The noise is smoothed Gaussian. Each data point is the result of averaging the error over 10 runs. The numbers on the abscissa indicate the fraction of the velocity used to modulate the random noise. The top plot shows the error in the estimation of the axis of rotation, and the bottom plot shows the percentage error in the estimation of the rotational velocity magnitude.

Chapter 8

Experimental Results: Real Data

8.1 Introduction

In this chapter, we present the results from applying the algorithms to real image sequences.

Two different methods were used to compute the flow fields. One is the multiscale matching approach of Anandan [5], and the other is a multiscale MRF procedure [25]. Brief descriptions of the multiscale matching method of Anandan and the multiscale MRF procedure were presented in Section 2.4 of Chapter 2.

In most of the experiments, we compare the computed motion parameters to the parameters available from the calibration information. The structure computation is possible only for relatively noise-free flow fields because relative depth is local information encoded in the optical flow vectors in a pointwise fashion. So, in some of the experiments, we compute the relative depth (structure) information and display the computed depth data much like a range map.

The implementation details are identical to those described in the previous chapter about experiments on synthetic data, unless otherwise noted.

8.2 The Straight flight sequence

This sequence was shot from a helicopter flying in a straight line, with very little rotational velocity. A sample picture from the sequence is shown in Fig. 8.1. The flow field was computed using the multiscale matching method. The original images are of the size 512×512 pixels. The flow fields were computed at a lower resolution (128×128) to save computation time and space.

The results of the Center-surround kernel method (with the horizontal differential of a Laplacian, as in Eqn. (5.7)) are shown in Fig. (8.2). The FOEs computed using the variance of the output of the convolution with a Laplacian kernel (Eqn. (5.8)) are shown in Fig. (8.3). The results of the quadratic polynomial projection method, computed using the same flow fields, are shown in Fig. (8.4). The results of using the subspace projection method and the fast (approximate) method are in Fig. (8.5) and Fig. (8.6) respectively. Angular measurements of errors are shown in

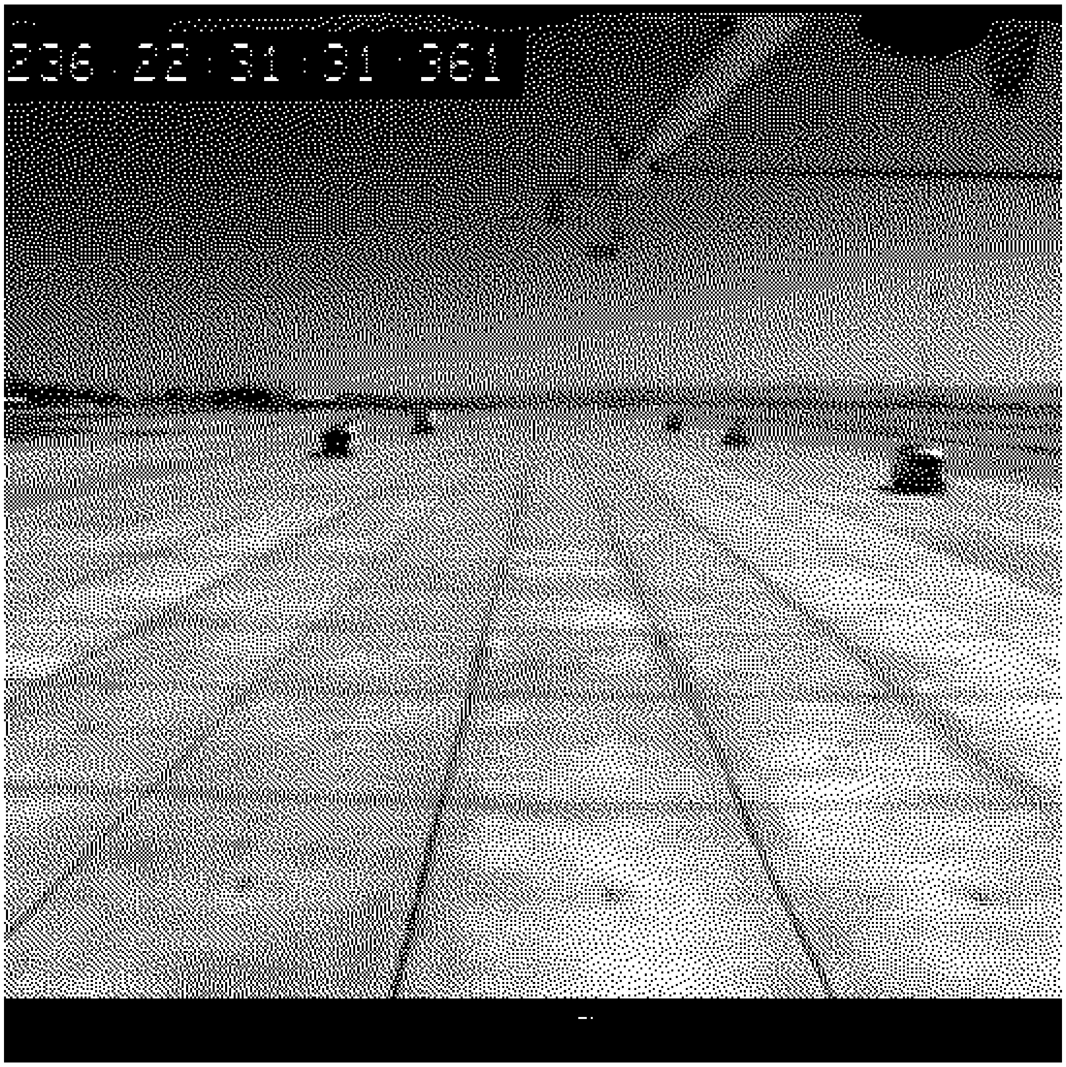


Figure 8.1: Sample frame from the Straight flight sequence.

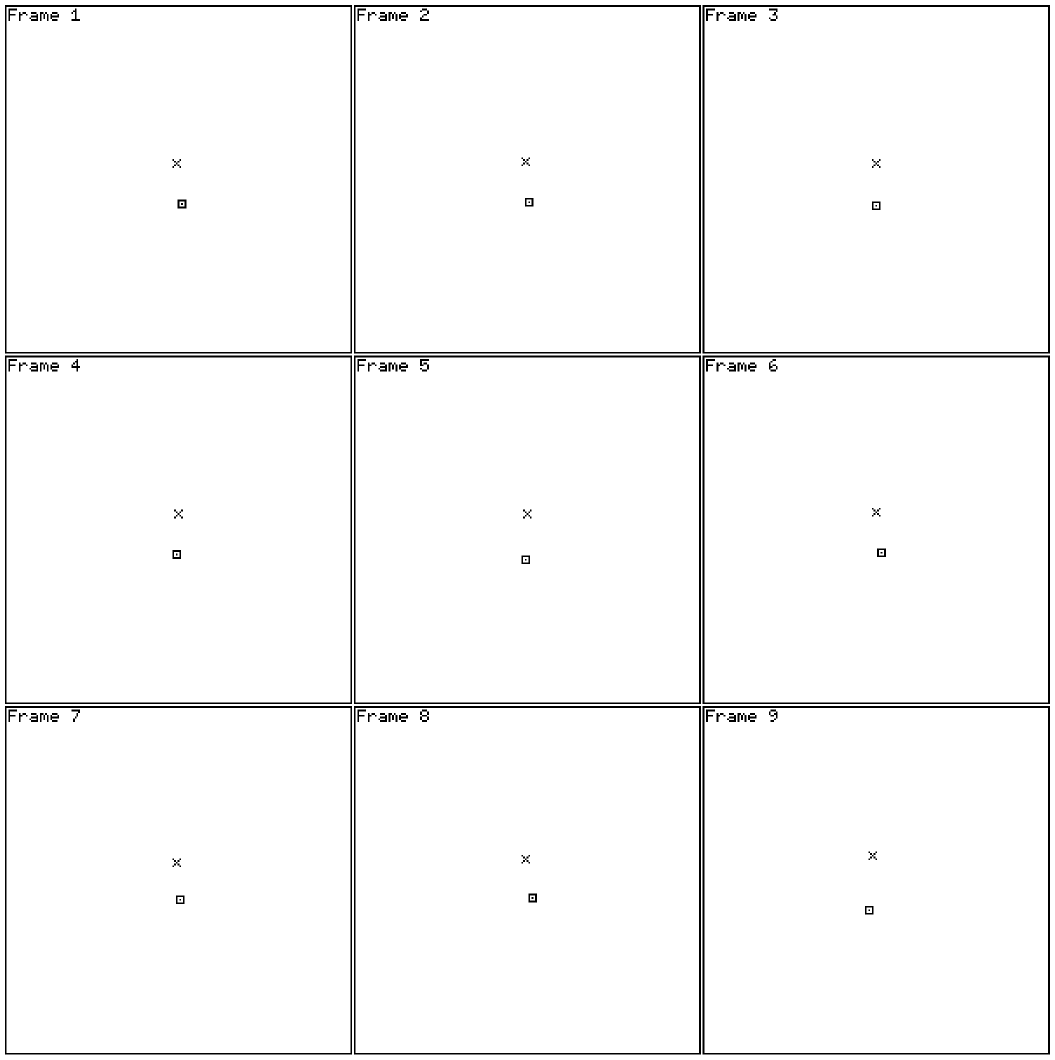


Figure 8.2: The computed and actual FOEs for the Straight Flight sequence, using the convolution method (using the horizontal derivative of the Laplacian). The cross denotes the position of the actual FOE, and the square shows the position of the computed FOE. The instantaneous FOE positions for the different frames are shown in this figure.

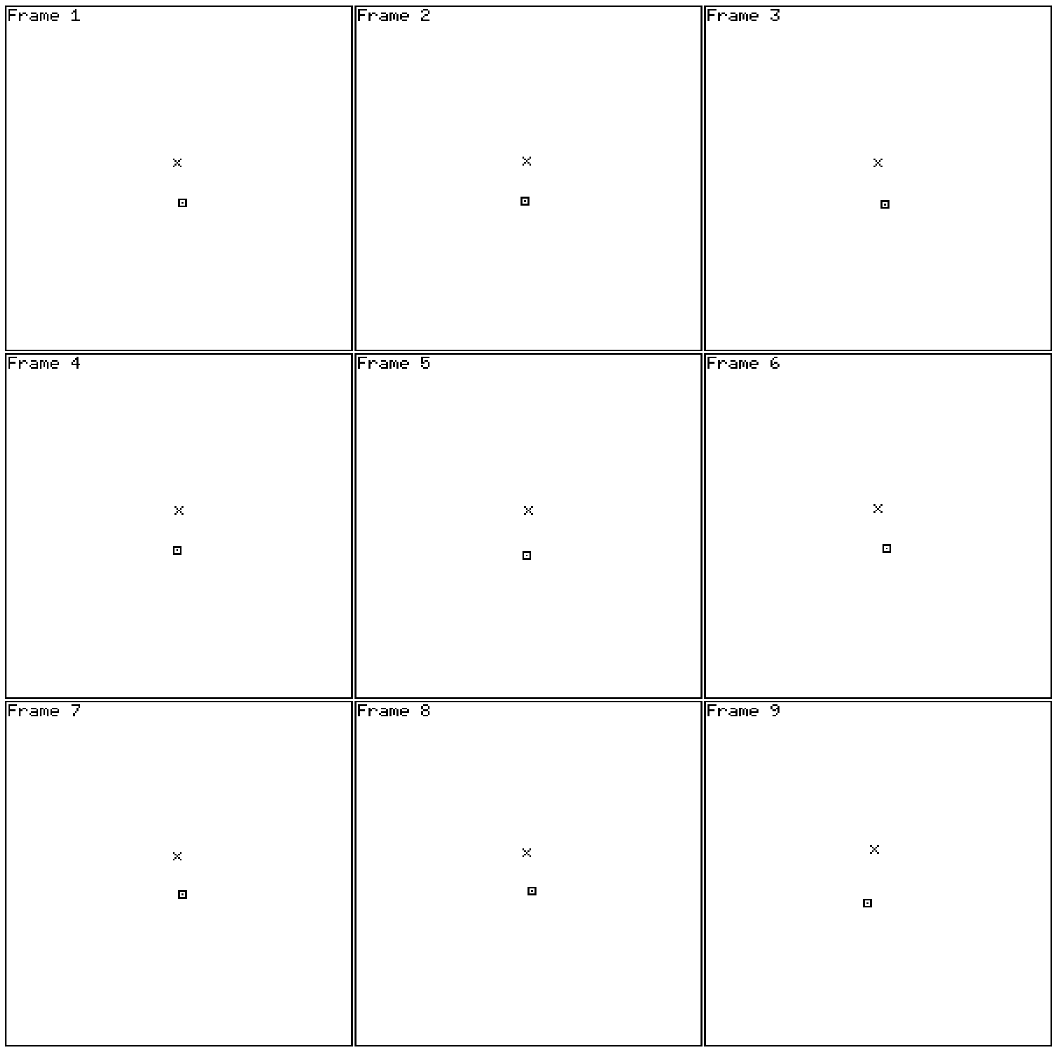


Figure 8.3: The computed and actual FOEs for the Straight Flight sequence, using the variance method (with the Laplacian kernel convolution).

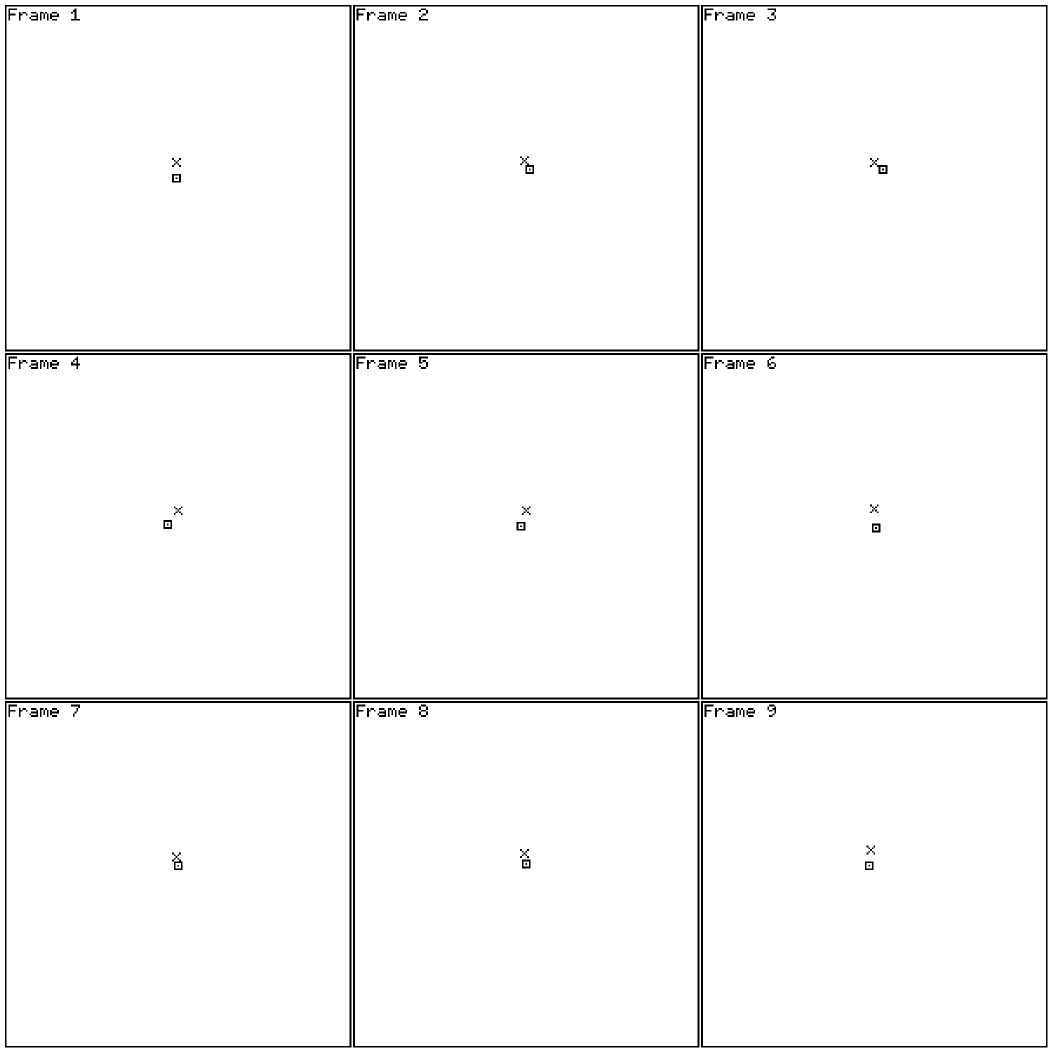


Figure 8.4: The computed and actual FOEs for the Straight Flight sequence, using the quadratic polynomial projection method. As in Fig. 8.2, the cross denotes the position of the actual FOE, and the square shows the position of the computed FOE. The instantaneous FOE positions for the different frames are shown in this figure.

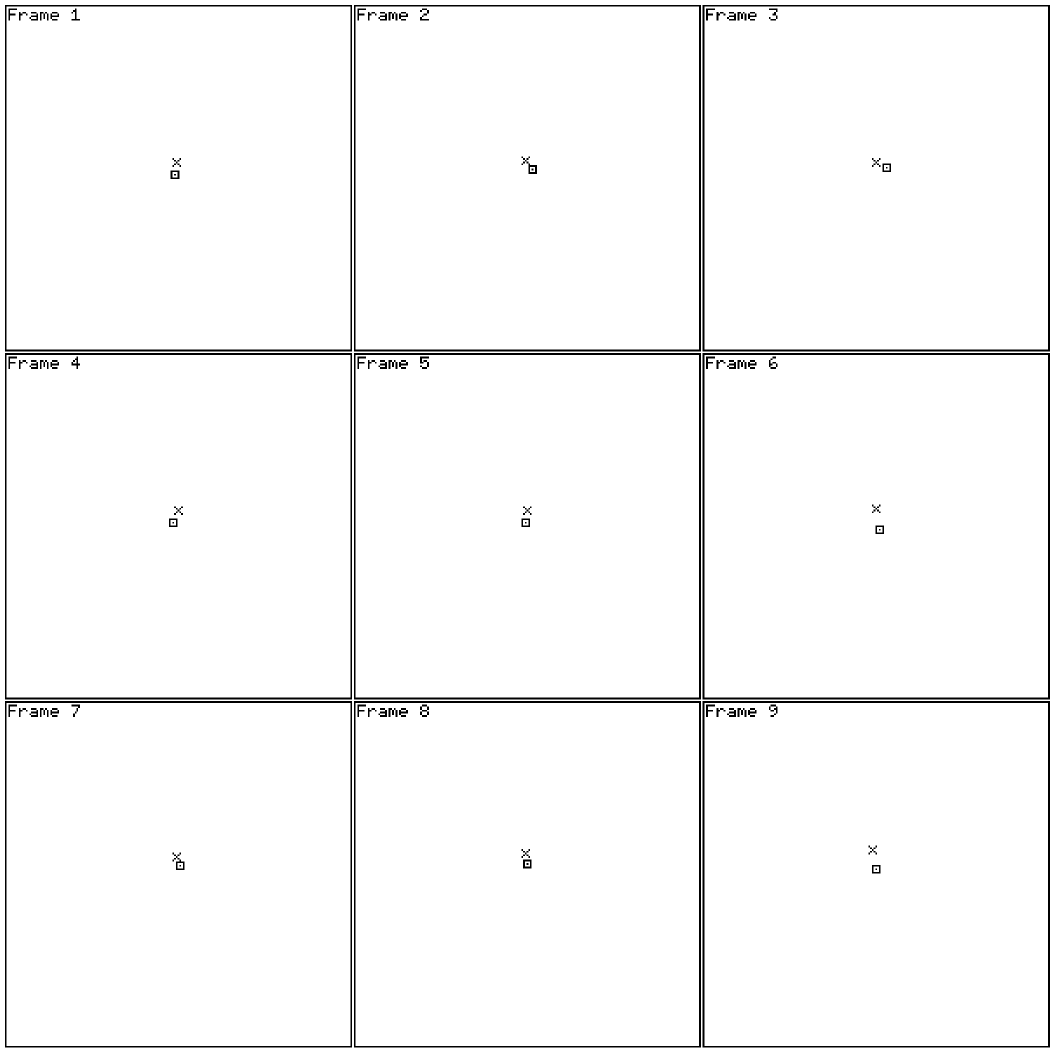


Figure 8.5: The computed and actual FOEs for the Straight Flight sequence, using the subspace projection method. As in Fig. 8.2, the cross denotes the position of the actual FOE, and the square shows the position of the computed FOE. The instantaneous FOE positions for the different frames are shown in this figure.

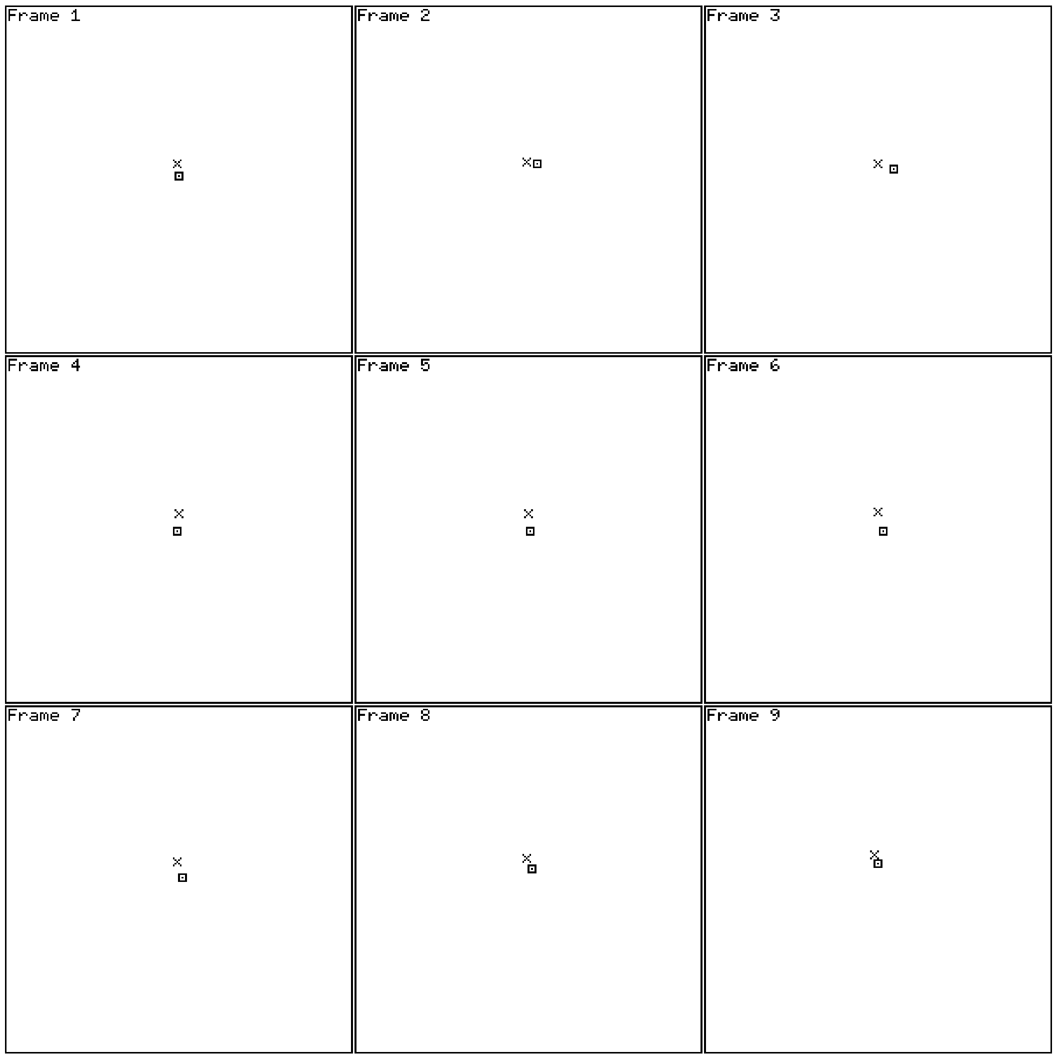


Figure 8.6: The computed and actual FOEs for the Straight Flight sequence, using the fast method (the NCC algorithm).

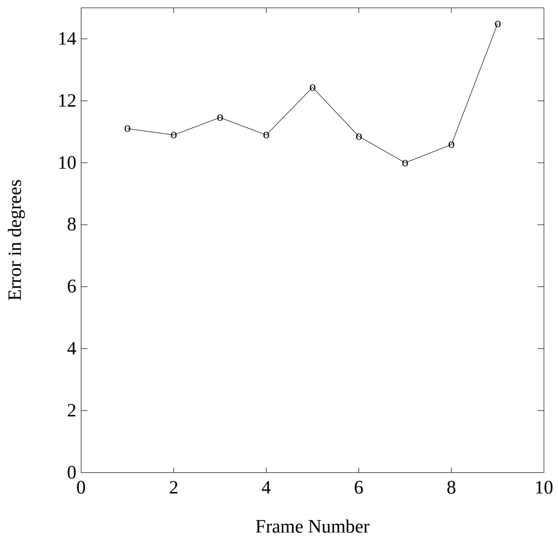


Figure 8.7: The error plot for the Center-surround kernel method.

Figs. (8.7), (8.8), (8.9), (8.10) and (8.11). The error is measured in degrees of angle between the direction of the actual FOE and the direction of the computed FOE. The results vary depending on the accuracy of the method. The subspace projection method is ideally the best because it uses the projection on to the precise three-dimensional subspace of quadratic polynomials. The quadratic polynomial projection method is less precise because it does not make use the specific form of the quadratic polynomials. The center-surround kernel methods are even less precise. The performance of the methods are consistent with these observations, as seen from the results shown in Figs. (8.7), (8.8), (8.9), (8.10) and (8.11). Good results are obtained by the fast method (the NCC algorithm) because the motion is predominantly translational.

8.3 The Turning Flight sequence

This sequence was also shot from a helicopter that has a predominantly forward motion, in addition to a significant rotation about the vertical axis. This proved to be a less favorable case for the algorithms presented here for estimating the translational motion. Indeed, errors averaging to about 9.5 degrees are obtained using the center-surround method, and to about 8 degrees for the subspace projection methods. However, this sequence is a good example to try the rotational parameter estimation because of the high rotation about the vertical axis.

The implementation of the flow circulation algorithm uses several square contours; adding flow components along each contour gives a quantity proportional to the curl at the center of the area enclosed by the square contour. For the turning flight sequence, the flow field was computed

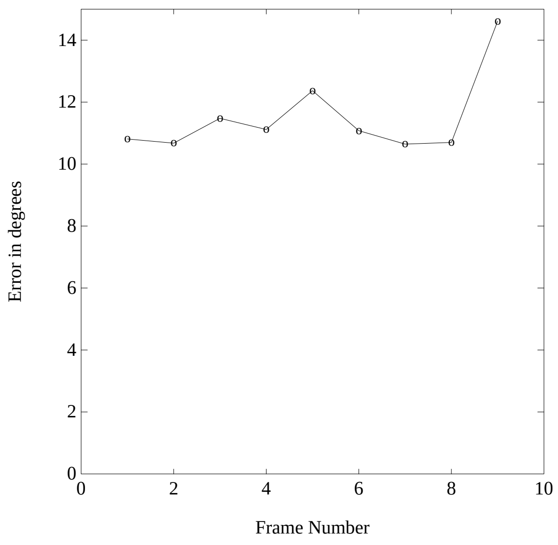


Figure 8.8: The error plot for the Center-surround kernel method using the variance computation.

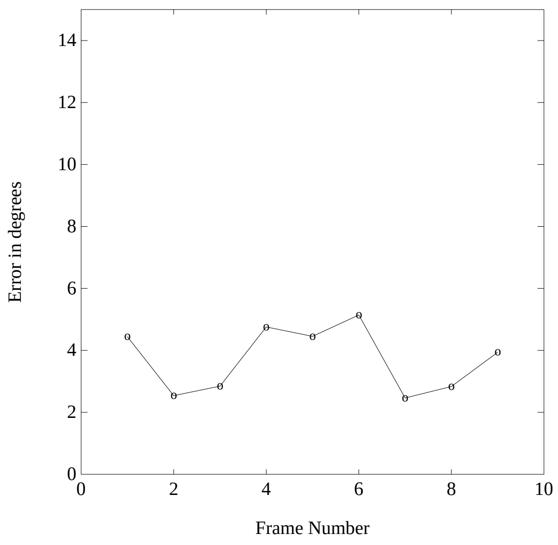


Figure 8.9: The error plot for the Quadratic polynomial projection method.

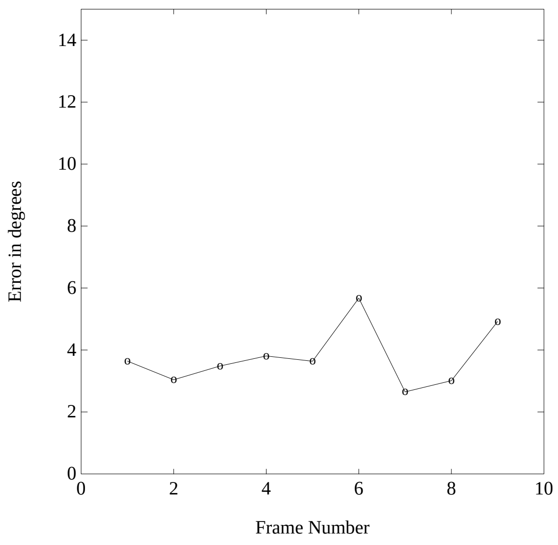


Figure 8.10: The error plot for the Subspace projection method.

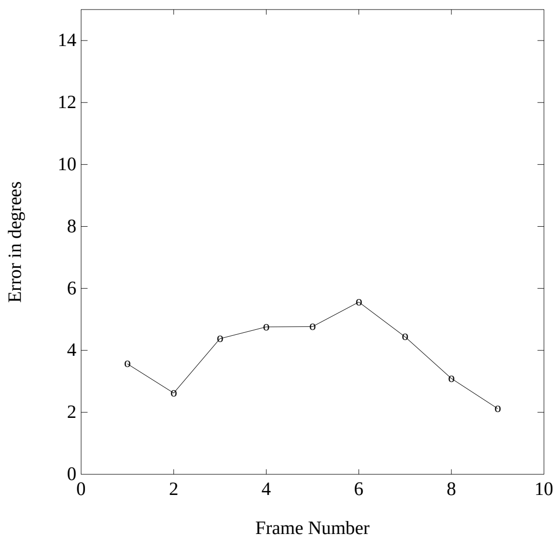


Figure 8.11: The error plot for the NCC algorithm.

frame	actual rotation			computed rotation		
	ω_1	ω_2	ω_3	ω_1	ω_2	ω_3
1	-0.0077	**-0.1200**	-0.0287	0.0063	**-0.1061**	-0.0064
2	-0.0160	**-0.1129**	0.0287	-0.0059	**-0.0990**	-0.0053
3	-0.0110	**-0.1204**	-0.0130	0.0041	**-0.1002**	-0.0059
4	-0.0120	**-0.1211**	-0.0164	0.0079	**-0.0875**	-0.0053
5	-0.0172	**-0.1155**	0.0296	-0.0070	**-0.1109**	-0.0068
6	-0.0135	**-0.1246**	-0.0256	-0.0397	**-0.0889**	-0.0053
7	-0.0185	**-0.1200**	0.0053	-0.0028	**-0.0992**	-0.0057
8	-0.0179	**-0.1201**	0.0110	-0.0034	**-0.0879**	-0.0054

Table 8.1: Table showing results of applying the flow circulation algorithm to the flow field data from the Turning Flight sequence.

over the 512×512 image using the Markov random field approach of Heitz and Bouthemy [25]. This computation yields a dense flow field, i.e., one flow vector per pixel position. Next, 5000 square contours of side 240 pixels were used to perform the linear surface fitting. The parameters of the linear surface are proportional to the rotational parameters of the camera. The results obtained on this sequence are shown in Table (8.3). It is seen that even though the method is an approximate one, the estimation of the largest component of the rotational velocity, namely the y-component, is fairly accurate in all the frames. Let us suppose that the rotation is constant over the eight frames (which is only approximately true). Then the estimated rotation parameters would give $(-0.0051, -0.0975, -0.0058)$ (compared to the average of the actual rotation parameters $(-0.0142, -0.1193, -0.0011)$), based on a simple average. Using this information, it is possible to reuse the optical flow data to compute the FOE directly.

8.4 The Ridge sequence

The camera moves over a ridge in this sequence. An interesting feature of this sequence is that the translation has a considerable "upward" component (as opposed to a predominantly-forward motion). This means that the FOE is not near the center of the image but closer to the upper border. Such sequences are known to be hard for motion parameter estimation due to the confounding of parameters. The flow due to upward translation is similar to the flow due to rotation about the horizontal axis. Thus, methods trying to "eliminate" the rotation would cancel out a large fraction of the flow due to the upward motion, resulting in an estimation of the FOE closer to the center of the image than it is actually the case. This interaction between the parameters is discussed in chapter 6.

The result of such confounding can be clearly seen in the FOE estimations by the convolution method, shown in Figs. (8.12a), (8.12b), (8.12c), and (8.12d). However, a method that does not attempt to eliminate the rotation, such as the NCC algorithm described in this thesis, is less likely to be affected by the confounding parameters problem. This is evident from the (more accurate) FOEs computed by the fast method, displayed in Figs. (8.13a), (8.13b), (8.13c), and (8.13d).

8.5 The Yosemite sequence

This sequence was generated using computer graphic techniques at MIT. The flow field was computed using the markov random field technique in [25]. The FOE was computed using the NCC algorithm. In this case, we demonstrate the feasibility of computing depth information, even though computing structure is not stressed in this thesis. This is because a reasonable computation of structure is possible only if we are able to obtain accurate flow fields. On the other hand, computation of the motion parameters is possible even if the flow field is not very accurate, because of the global nature of the algorithms presented here.

Fig. (8.15) shows the flow field computed for a pair of images from the Yosemite sequence. A sample image from the sequence is shown in Fig. (8.14).

The relative depth values computed using the FOE estimated using the NCC algorithm (and assuming zero rotation) are shown as a range map in Fig. (8.16). The brighter a point, the closer it is to the viewer. A circular portion around the FOE has been masked out because the flow magnitudes are very small in the neighborhood of the FOE and hence very unstable. It can be seen that the depth information seems to be qualitatively correct (see the intensity image in Fig. (8.14) for comparison).

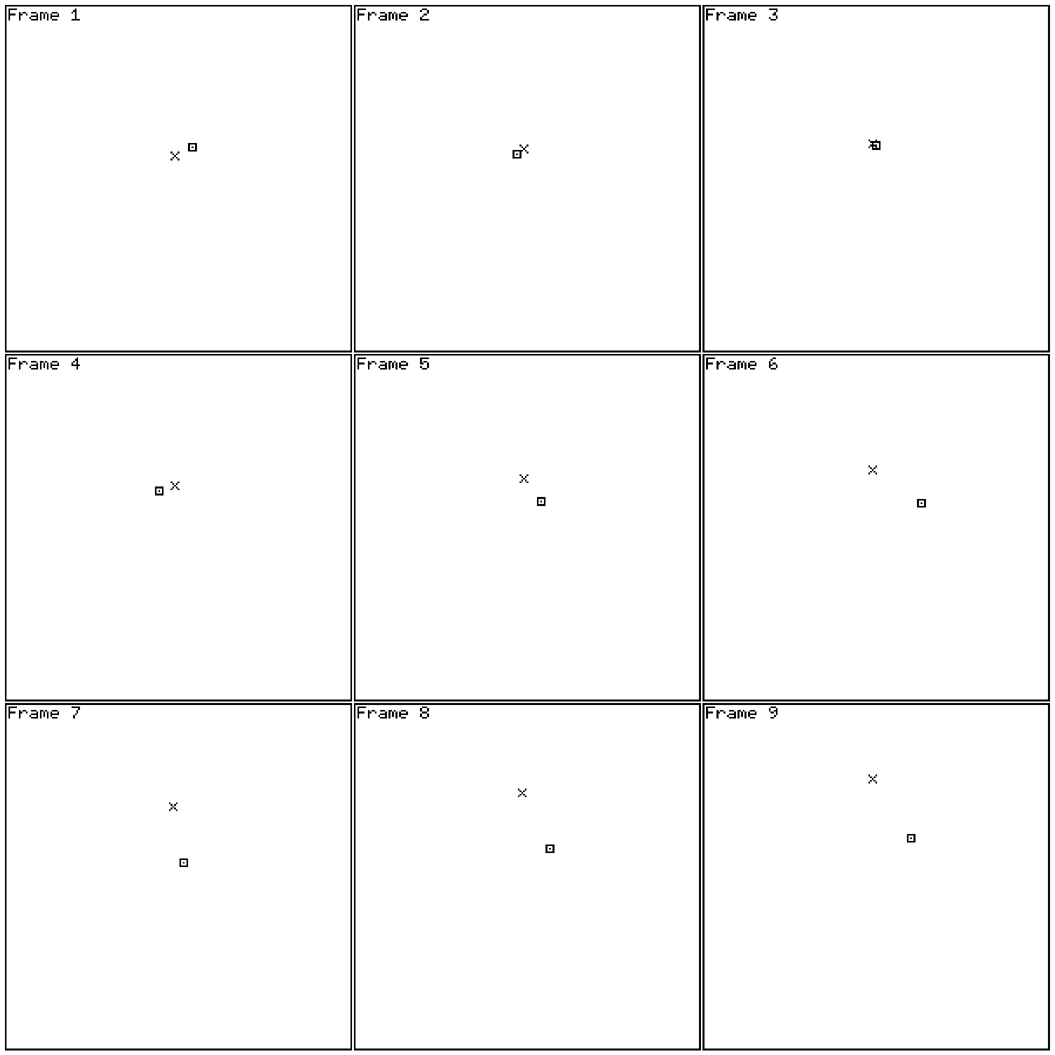


Figure 8.12a: The computed and actual FOEs for the ridge sequence, using the convolution method (using the horizontal derivative of the laplacian). The cross denotes the position of the actual FOE, and the square shows the position of the computed FOE. The instantaneous FOE positions for the different frames are shown in this figure, and continued in Figs. (8.12b), (8.12c), and (8.12d).

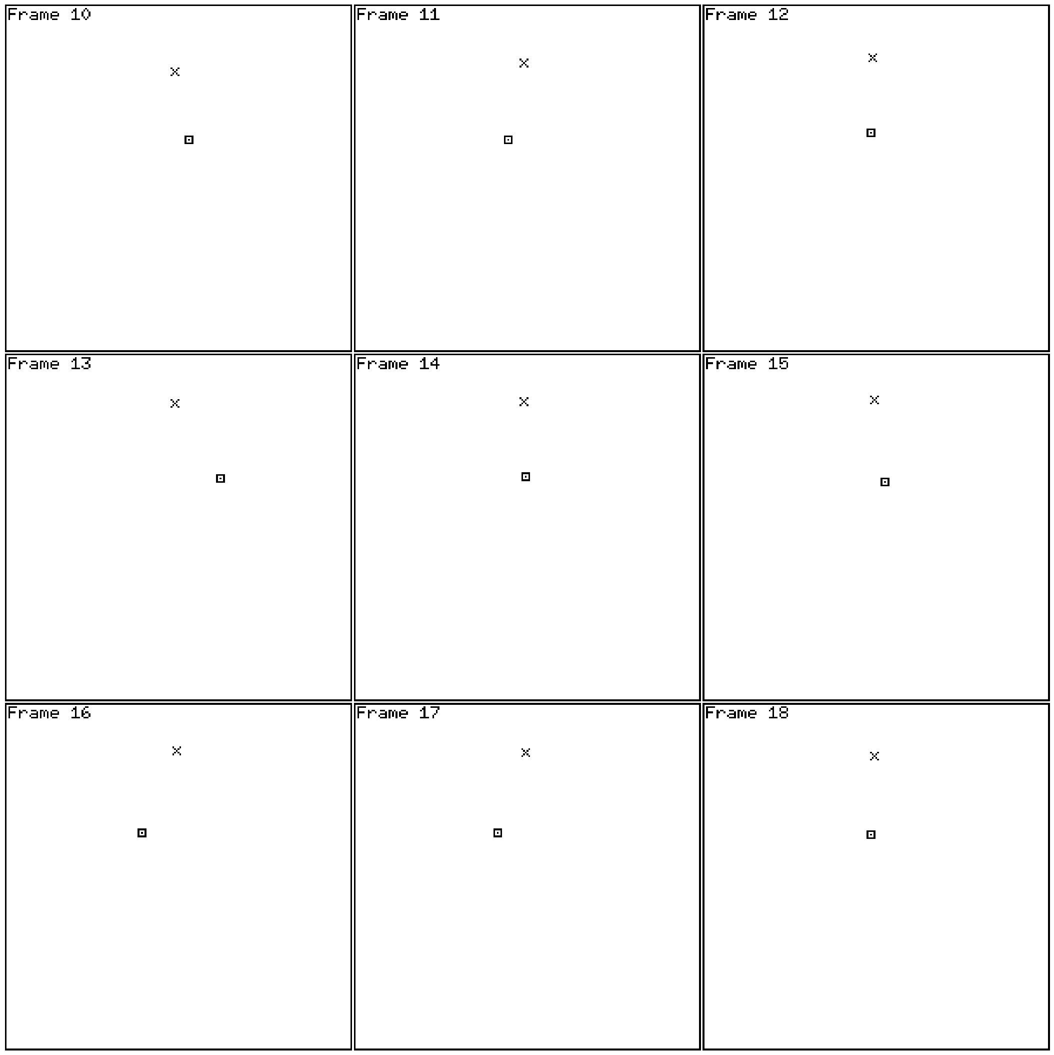


Figure 8.12b: See the note in Fig. (8.12a).

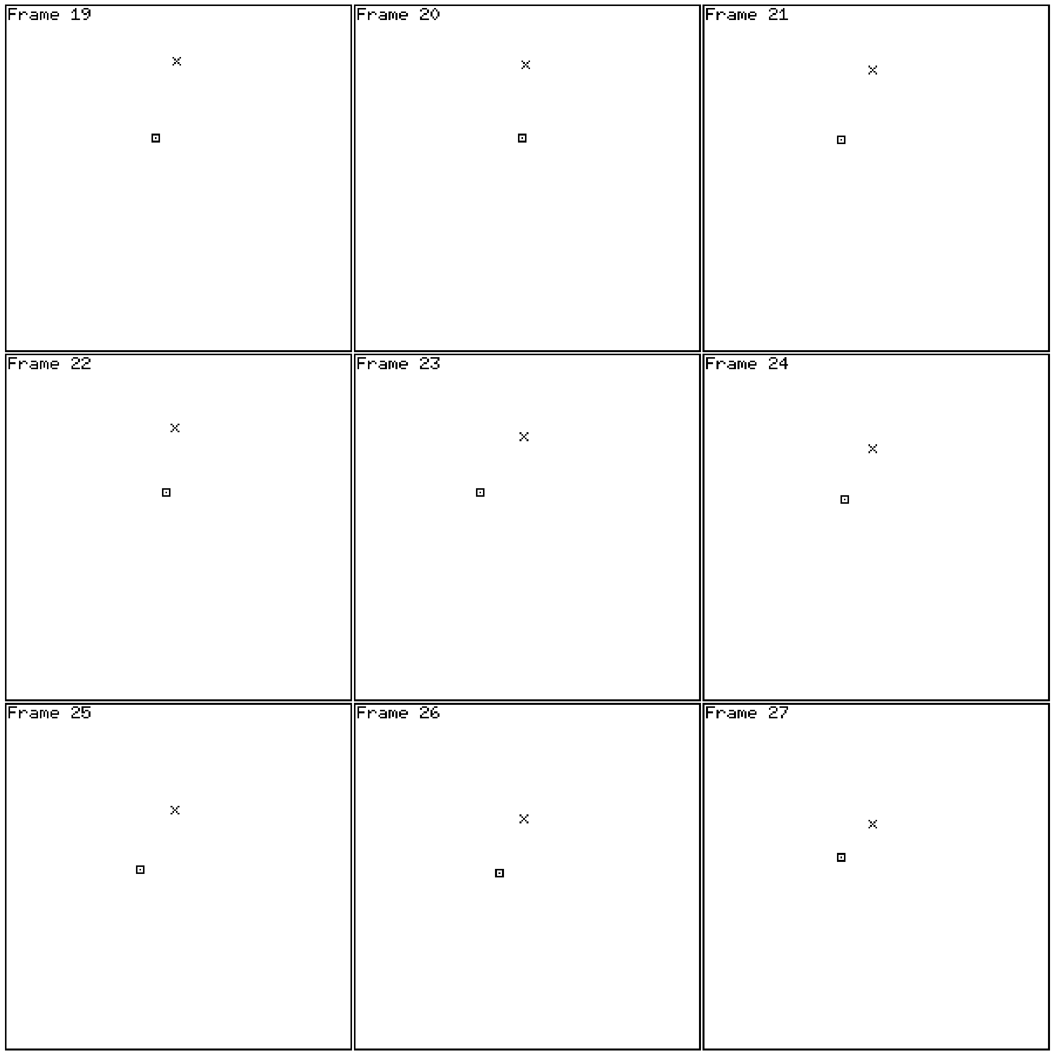


Figure 8.12c: See the note in Fig. (8.12a).

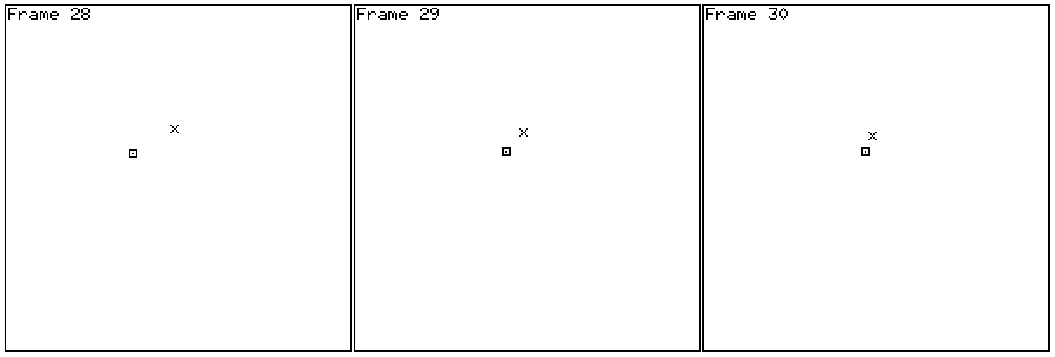


Figure 8.12d: See the note in Fig. (8.12a).

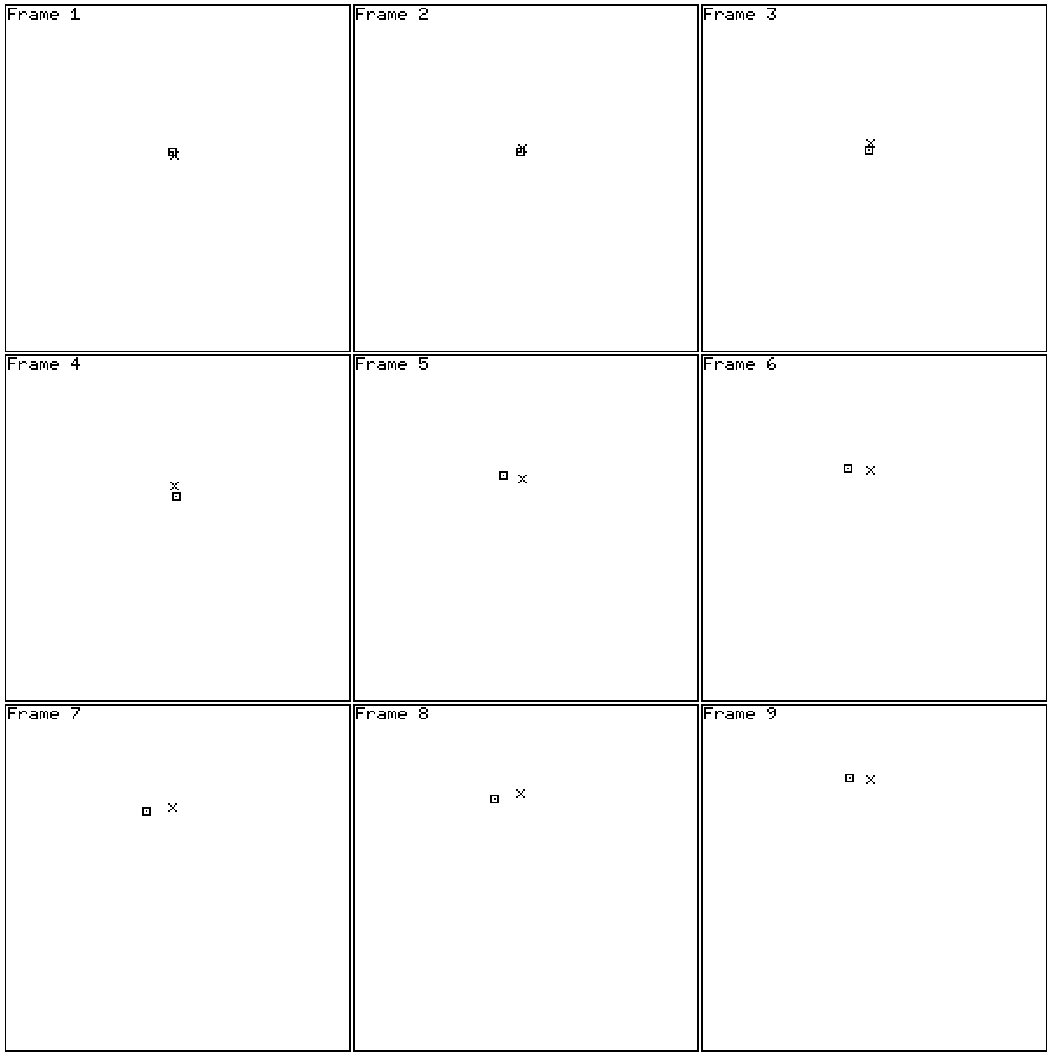


Figure 8.13a: The computed and actual FOEs for the ridge sequence, using the norm of circular components method. The cross denotes the position of the actual FOE, and the square shows the position of the computed FOE. The instantaneous FOE positions for the different frames are shown in this figure, and continued in Figs. (8.13b), (8.13c), and (8.13d).

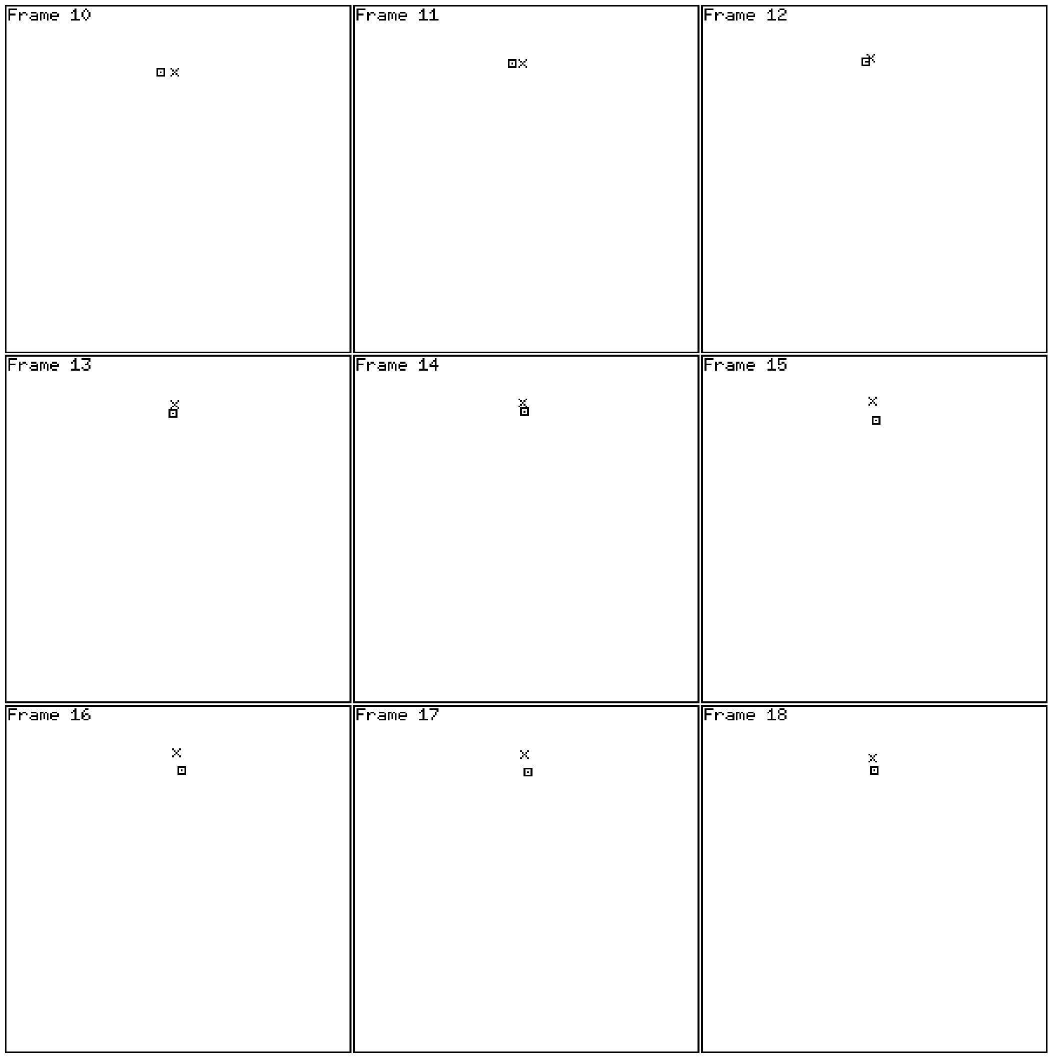


Figure 8.13b: See the note in Fig. (8.13a).

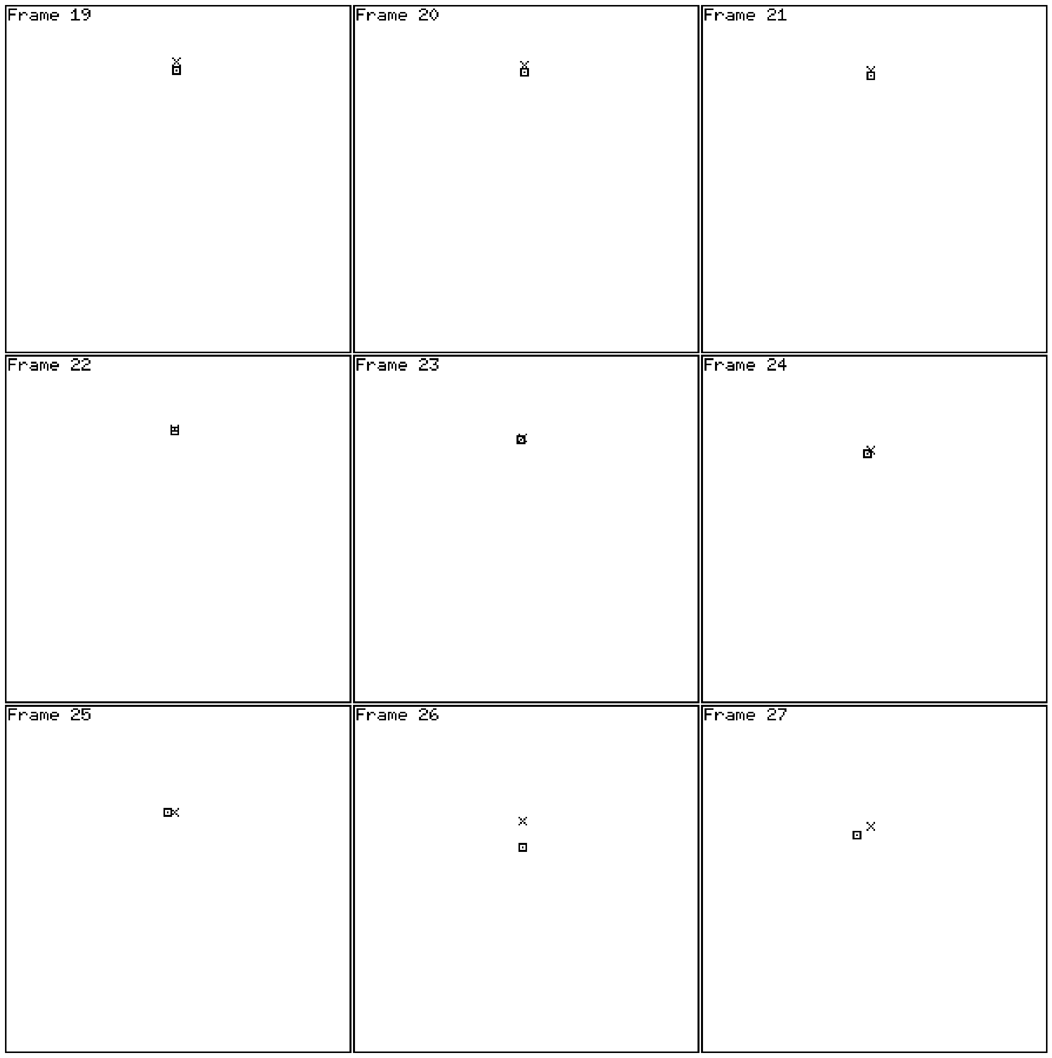


Figure 8.13c: See the note in Fig. (8.13a).

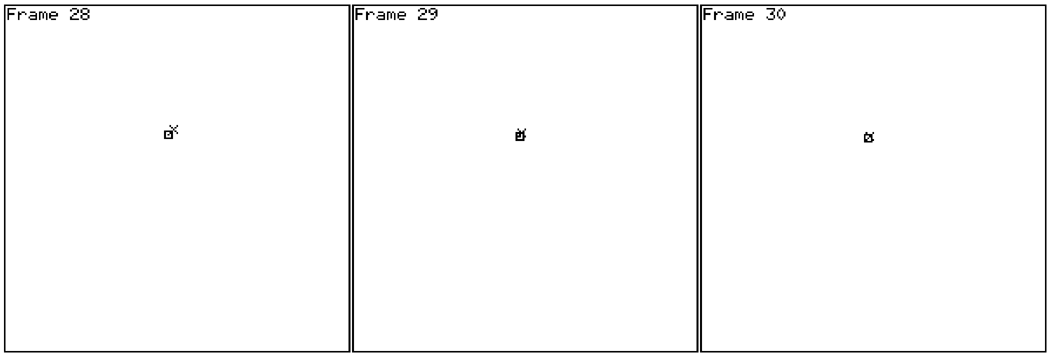


Figure 8.13d: See the note in Fig. (8.13a).

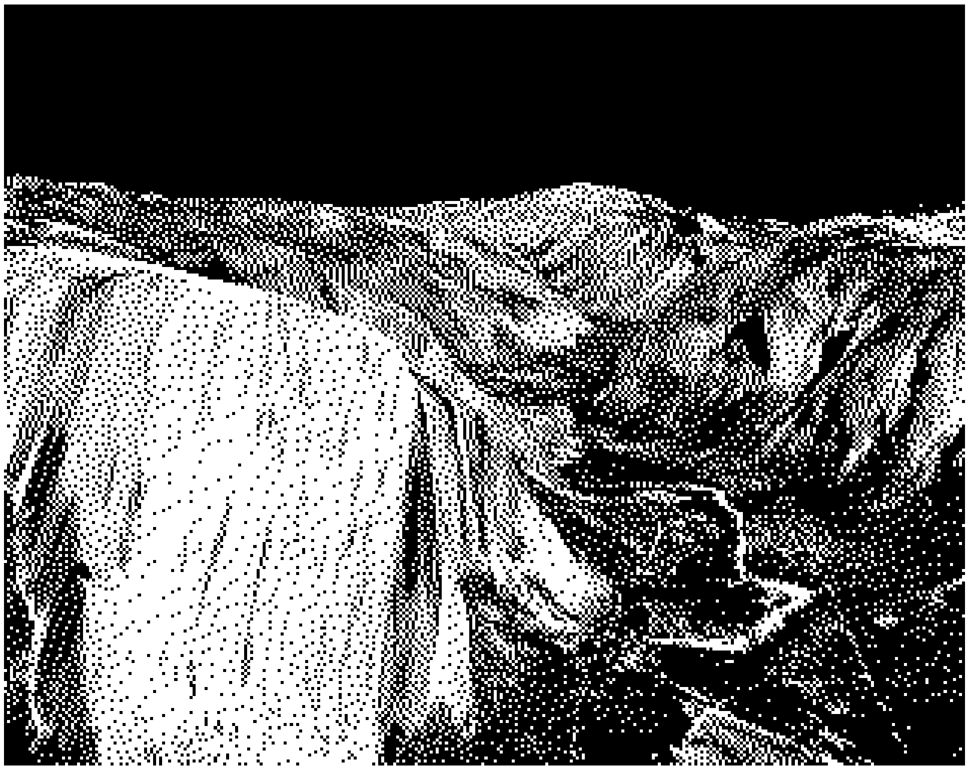


Figure 8.14: A sample frame from the Yosemite sequence.

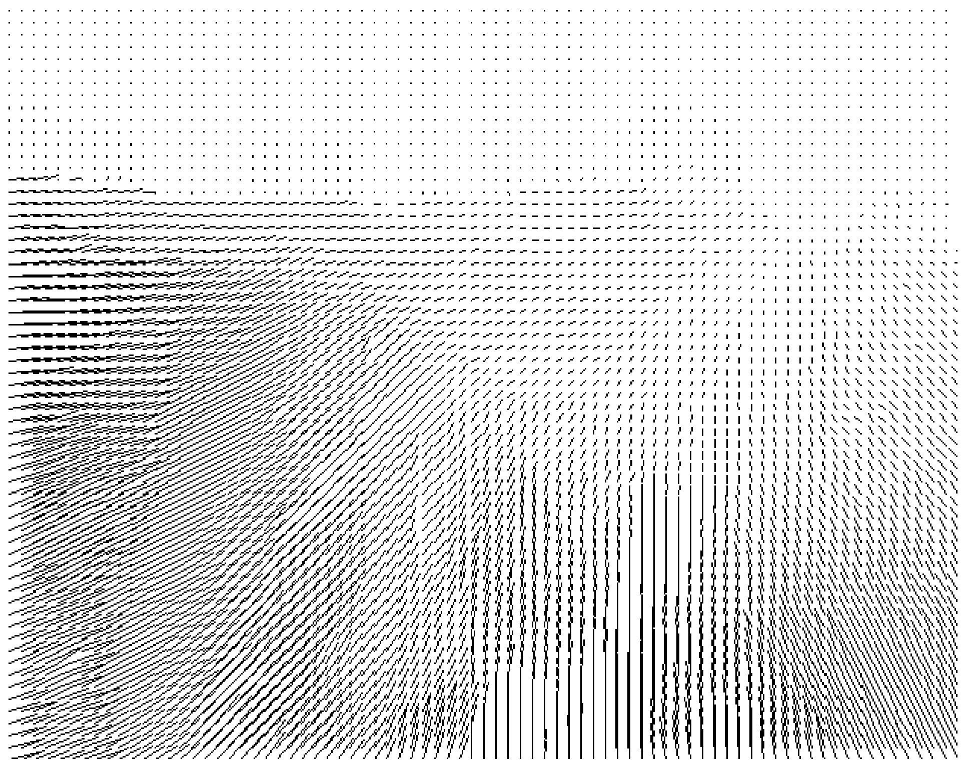


Figure 8.15: The flow field computed using the frame in Fig. (8.14) and the subsequent frame in the sequence.

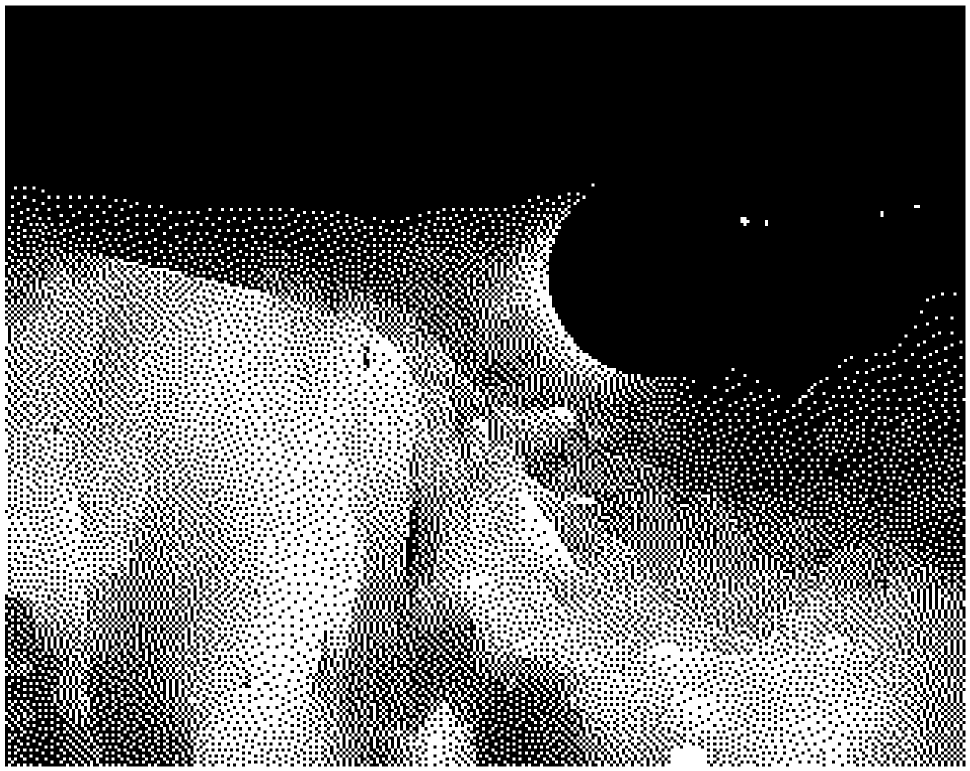


Figure 8.16: The relative depth map computed using the motion parameters obtained from the flow field in Fig. (8.15) by applying the NCC algorithm. A circular area around the FOE has been masked out because flow values close to the FOE are very small and hence unstable. Brighter points correspond to smaller relative depth (nearer features).

Chapter 9

Conclusions

We presented algorithms to compute the motion parameters of a sensor. Unlike existing alternative methods, our methods use all available data, qualifying as *global methods.* These can be expected to be more robust when compared to local methods that operate on data from a small region of the image. This is supported by experimental results that were shown in this thesis.

The *flow circulation algorithm* computes the rotational parameters. It is based on the observation that the curl of the vector optical flow field is linear in the rotational parameters. Instead of using the curl values, equivalently, we make use of *circulation values* which are contour integrals on the image plane. The parameters of the linear fit are declared to be the rotational velocities. The algorithm is approximate, in that the curl values are composed of not just the linear terms from the rotational parameters, but also contain contribution from the translational parameters. The effect of the translational terms is nullified for pure rotation and for frontal planar surfaces; under such conditions, the linearity is exact and we obtain the correct rotational parameters (for no-noise situations). Under other conditions, the linearity is only approximate, and the approximation is expected to hold well in a global sense; i.e., if we use data from all over the image, we expect the deviations from the linearity to cancel out under typical circumstances.

From the experiments, we find that this is indeed the case. We obtained very accurate results for flow fields simulated using range data (of non-planar objects), even in the presence of translation. Using flow fields computed from a real image sequence (taken from an aircraft), we obtained quite accurate estimates for the angular velocity. We observe from our derivations and from the experiments that the algorithm is exact for pure rotation, and is globally approximate for typical situations. The algorithm is very easy to implement because of the simplicity of the computations. Integrating along square contours is particularly simple if a dense flow field is available on the cartesian image plane. Note however, that it is sufficient to obtain a few measurements of the curl value from areas of the image with the same relative motion parameters. This could conceivably be done by feature detectors measuring *circulation.* The results from the noise experiments suggest that the performance of the algorithm degrades gracefully with respect to noise, starting with zero error in the case of pure rotation.

The FOE search algorithms are also easily understood. For candidate FOEs, circular component functions are computed; the projection of the circular component function on suitable spaces leaves behind a residual function whose norm is a measure of the correctness of the choice of the FOE. In particular, for the correct choice of the FOE, this norm is zero, and is non-zero for the other choices. We provide several methods to perform this computation. Of particular interest are the methods that exploit the known (quadratic) shape of the error surface in order to simplify the algorithms. As a result, the center-surround kernel and the quadratic polynomial projection method involve no search; we need to do the residue computation only for six choices of the FOE. The correct FOE is obtained by a simple computation (a linear system solution).

The algorithms perform well on synthetic and real data. For the noise models treated here, the degradation with increasing noise magnitude is smooth. The quadratic polynomial projection method performs better than the center-surround kernel method. The NCC algorithm performs better than either of these if there is little or no rotational velocity. This can be seen from the results from both synthetic and real data. The subspace projection method involves searching and so is not as computationally efficient as the other methods. Also, in the experiments on real images, the quadratic polynomial projection method seems to perform as well as the subspace projection method. The center-surround kernel method, even though fast, does not do well on real image sequences possibly because of its local derivative operation. Thus, the quadratic polynomial projection method seems to be a good compromise since it is fast and quite accurate. For sequences where motion is known to be predominantly translational, the method of choice is the NCC algorithm because it is the fastest one and is the most accurate for such sequences.

For the methods presented in this thesis, we have assumed that the optical flow field is given to us. For the experiments with real image sequences, we used available code to do the computation. Often, the optical flow field was very noisy. Obtaining noise-free optical flow fields is still an elusive issue and is provably impossible to obtain in many situations, due to the aperture problem. Any advances in better computation methods for optical flow fields will directly benefit the algorithms presented in this thesis.

In the methods presented in this thesis, we have ignored certain issues of visual motion analysis. We mention them here. The judgement of egomotion and scene structure does not necessarily have to be obtained exclusively from the optical flow field. Correspondence of features is also a powerful clue to self-motion and structure. Also, the analysis here has been implicitly restricted to two-frame motion. It is possible to consider several frames at once, not merely to obtain a reliable optical flow (as attempted by Heeger [22], for example), but integration of information across time by the use of techniques such as Kalman filtering might be necessary to reduce the noise-sensitivity of the computations. Future work in visual motion analysis should consider integration of the different sources of information, arising from modalities such as optical flow and correspondence and temporal consistency. Other useful techniques that are likely to simplify motion analysis include tracking, active vision, and qualitative vision.

There is also a need to study the confounding parameters problem. As discussed in Chapter 6, there is an interplay between the rotational parameters and the translational parameters. The effect is worsened when the field of view is small or when the entire scene consists of planar or nearly-planar structures. This is an issue that affects not only the algorithms presented here; all the algorithms that exist to do motion parameter estimation or structure from motion are plagued by this issue. It would be useful to find a different set of motion parameters which does not suffer from this problem of interaction between parameters.

Bibliography

[1] E.H. Adelson and J.R Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2:284–299, 1985.

[2] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:384–401, 1985.

[3] J. K. Aggarwal and N. Nandhakumar. On the compuation of motion from sequences of images–a review. *Proceedings of the IEEE*, 76(8):917–935, August 1988.

[4] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1:333–356, 1988.

[5] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[6] S.T. Barnard and W.B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:333–340, 1980.

[7] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. Technical Report RPL-TR-9107, Robotics and Perception Laboratory, Queens University, Kingston, Ontario, Canada, July 1992. Also as TR No. 299, Dept. of Comp. Sci., U of Western Ontario.

[8] P. Bouthemy. A maximum-likelihood framework for determining moving edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):499–511, May 1989.

[9] W. Burger and B. Bhanu. Estimating 3-d egomotion from perspective image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1040–1058, November 1990.

[10] P. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

[11] P.J. Burt. Fast algorithms for estimating local image properties. *Computer Vision, Graphics and Image Processing*, 21:368–382, 1983.

[12] M. Fahle and T. Poggio. Visual hyperacuity: Spatio-temporal interpolation in human vision. *Proceedings of the Royal Society of London B 213*, B 213:451–477, 1981.

[13] F. Faugeras, O.D. Lustman and G. Toscani. Motion and structure from point and line matches. In *Proc. of the First International Conference on Computer Vision*, 1987.

[14] D. Fleet. Personal communication, 1992.

[15] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–104, 1990.

[16] J. J. Gibson. *The Perception of the Visual World.* The Riverside Press, Cambridge, Mass, 1950.

[17] J. J. Gibson, P. Olum, and F. Rosenblatt. Parallax and perspective during aircraft landings. *American Journal of Psychology*, 68:372–385, 1955.

[18] N. M. Grzywacz and A. L. Yuille. A model for the estimate of local image velocity by cells in the visual cortex. *Proc. Royal Soc. Lond. A*, 239:129–161, 1990.

[19] D. Heeger and A. Jepson. Simple method for computing 3d motion and depth. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 96–100, Osaka, Japan, December 1990.

[20] D. Heeger and A. Jepson. Visual perception of 3d motion. *Neural Computation*, 2:127–135, 1990.

[21] D. Heeger and A Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. Research in Biological and Computational Vision Tech Rep RBCV-TR-90-35, University of Toronto, 1991.

[22] D. J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1:279–302, 1988.

[23] D. J. Heeger and A. Jepson. Visual perception of three-dimensional motion. MIT Media laboratory technical report 124, Massachusetts Institute of Technology, December 1989.

[24] D. J. Heeger and A. D. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992.

[25] F. Heitz and P. Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. Technical report, IRISA/INRIA, Campus Universitaire de Beaulieu, 35042-Rennes Cedex, France, 1990.

[26] F. B. Hildebrand. *Advanced Calculus for Applications.* Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1962.

[27] E.C Hildreth. Computations underlying the measurement of visual motion. *Artificial Intelligence*, 23:309–354, 1984.

[28] B. K. P. Horn and E. J. Weldon. Direct methods for recovering motion. *International Journal of Computer Vision*, 2:51–76, 1988.

[29] B.K.P Horn. *Robot Vision.* The MIT Press, 1987.

[30] B.K.P Horn and B.G Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[31] R. Hummel and V. Sundareswaran. Motion parameter estimation from global flow field data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear.

[32] A. Jepson and D. Heeger. Subspace methods for recovering rigid motion ii: Theory. Research in Biological and Computational Vision Tech Rep RBCV-TR-90-36, University of Toronto, 1991.

[33] C. Jerian and R. Jain. Polynomial methods for structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:1150–1166, 1990.

[34] G. Johansson. Visual motion perception. *Scientific American*, 232:76–88, 1975.

[35] J. J. Koenderink. Some theoretical aspects of optic flow. In Rik Warren and Alexander H. Wertheim, editors, *Perception and control of self-motion*, chapter 2, pages 53–68. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1990.

[36] J. J. Koenderink and A. J. Van Doorn. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica Acta*, 22(9):773–791, 1975.

[37] J. J. Koenderink and A. J. Van Doorn. Local structure of movement parallax of the plane. *Journal of the Optical Society of America*, 66(7), 1976.

[38] J. J. Koenderink and A. J. Van Doorn. Exterospecific component of the motion parallax field. *Journal of the Optical Society of America*, 71(8):953–957, 1981.

[39] D. Lawton. Processing translational image sequences. *Computer Vision, Graphics and Image Processing*, 22:116–144, 1983.

[40] J. R. Lishman and D. N. Lee. The autonomy of visual kinaesthesis. *Perception*, 2:287–294, 1973.

[41] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[42] H.C. Longuet-Higgins. The visual ambiguity of a moving plane. *Proc. Royal Soc. Lond. B*, 223:165–175, 1984.

[43] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proc. Royal Soc. Lond. B*, 208:385–397, 1980.

[44] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. 7th Intern. Joint Conference on AI*, pages 674–679, Vancouver, 1981.

[45] S. J. Maybank. Algorithm for analysing optical flow based on the least-squares method. *Image and Vision Computing*, 4:38–42, 1986.

[46] S. J. Maybank. *A Theoretical Study of Optical flow.* PhD thesis, University of London, 1987.

[47] A. Mitiche, R. Grisell, and J.K. Aggarwal. On the smoothness of a vector field–application to optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):943–949, 1988.

[48] H.P. Moravec. Towards automatic visual obstacle avoidance. In *Proc. 5th International Joint Conference on Artificial Intelligence*, page 584, 1977.

[49] H. H. Nagel. On the estimation of optical flow: relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324, 1987.

[50] K. Nakayama. Biological image motion processing: A review. *Vision Research*, 25:625–660, 1985.

[51] K. Nakayama and J. Loomis. Optical velocity patterns, velocity-sensitive neurons, and space perception: a hypothesis. *Perception*, 3:63–80, 1974.

[52] S. Negahdaripour and B. K. P. Horn. A direct method for locating the focus of expansion. *Computer Vision, Graphics and Image Processing*, 46:303–326, 1989.

[53] K. Prazdny. Egomotion and relative depth from optical flow. *Biological Cybernetics*, 36:87–102, 1980.

[54] K. Prazdny. Determining the instantaneous direction of motion from optical flow generated by a curvilinearly moving observer. *Computer Vision, Graphics and Image Processing*, 17:238–248, 1981.

[55] K. Prazdny. On the information in optical flows. *Computer Vision, Graphics and Image Processing*, 22:239–259, 1983.

[56] D. Regan and K. I. Beverley. Visual responses to vorticity and the neural analysis of optic flow. *Journal of the Optical Society of America A*, 2(2):280–283, February 1985.

[57] J. H. Rieger and D. T. Lawton. Processing differential image motion. *Journal of the Optical Society of America A*, 2:354, 1985.

[58] I. Rigoutsos. *Massively parallel Bayesian object recognition.* PhD thesis, New York University, New York, Aug 1992.

[59] A. Rosenfeld and A.C. Kak. *Digital Picture Processing.* Academic Press, New York, 1976.

[60] R. Schalkoff. *Digital Image Processing and Computer Vision.* John Wiley, 1989.

[61] I.K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:56–73, January 1987.

[62] E.P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optical flow. In *Proc. Computer Vision and Patter Recognition*, pages 310–315, Maui, Hawaii, June 1991.

[63] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control.* John Wiley & Sons, 1989.

[64] G. Strang. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, San Diego, third edition, 1988.

[65] M. Subbarao and A. Waxman. On the uniqueness of image flow solutions for planar surfaces in motion. *Computer Vision, Graphics and Image Processing*, 36:208–228, 1986.

[66] V. Sundareswaran. Egomotion from global flow field data. In *IEEE Workshop on Visual Motion*, pages 140–145, Princeton, New Jersey, Oct 1991.

[67] V. Sundareswaran. A fast method to estimate sensor translation. In *Proceedings of the Second European Conference on Computer Vision*, Santa Margherita, Italy, May 1992.

[68] V. Sundareswaran and R. Hummel. Motion parameter estimation using the curl of the flow field. In *Eighth Israeli Conference on AI and Computer Vision*, Tel-Aviv, Dec 1991.

[69] K. Tanaka and Hide-Aki Saito. Analysis of motion of the visual field by direction, expansion/contraction, and rotation cells clustered in the dorsal part of the medial superior temporal area of the macaque monkey. *Journal of Neurophysiology*, 62(3):626, September 1989. Also see page 642.

[70] C. Tomasi and T. Kanade. Factoring image sequences into shape and motion. In *IEEE Workshop on Visual Motion*, pages 21–28, Princeton, New Jersey, Oct 1991.

[71] R. Tsai and T. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid bodies with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:13–27, 1984.

[72] S. Ullman. *The interpretation of visual motion.* M. I. T. Press, Cambridge, 1979.

[73] S. Ullman. Analysis of visual motion by biological and computer systems. *Computer*, 14(8):57–69, 1981.

[74] A. Verri and T. Poggio. Motion field and optical flow: Qualitative properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:490–498, 1989.

[75] H. von Helmholtz. *Handbuck der Physiologischen Optik.* Verlag von Leopold Voss, Hamburg, 1896. Translation published by Dover Press.

[76] H. Wallach and D.N. O'Connell. The kinetic depth effect. *Journal of Experimental Psychology*, 45:205–217, 1953.

[77] A. Waxman, B. Kamgar-Parsi, and M. Subbarao. Closed form solutions to image flow equations. In *Proc. of the First International Conference on Computer Vision*, pages 12–24, London, 1987.

[78] A. Waxman and S. Ullman. Surface structure and 3-d motion from image flow: A kinematic analysis. *International Journal of Robotics Research*, 4(3):72–94, 1985.

[79] A. Waxman and K. Wohn. Contour evolution, neighborhood deformation, and global image flow: Planar surfaces in motion. *International Journal of Robotics Research*, 4:95–108, 1985.

[80] T.S. Weng, J. Huang and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:451–476, 1989.

[81] P. Werkhoven and J. J. Koenderink. Extraction of motion parallax structure in the visual system i. *Biological Cybernetics*, 63:185–191, 1990. Also see pages 193–199.

[82] K. Wohn and A. Waxman. The analytic structure of image flows: deformations and segmentation. *Computer Vision, Graphics and Image Processing*, 49:127–151, 1990.