Geometry and Statistics of Visual Space-Time

Cornelia Fermüller, Patrick Baker and Yiannis Aloimonos*

Abstract

Although the fundamental ideas underlying research efforts in the field of computer vision have not radically changed in the past two decades, there has been a transformation in the way work in this field is conducted. This is primarily due to the emergence of a number of tools, of both a practical and a theoretical nature. One such tool, celebrated throughout the nineties, is the geometry of visual space-time. It is known under a variety of headings, such as multiple view geometry, structure from motion, and model building. It is a mathematical theory relating multiple views *(images) of a scene taken at different viewpoints to* three-dimensional models *of the (possibly dynamic) scene. This mathematical theory gave rise to algorithms that take as input images (or video) and provide as output a model of the scene. Such algorithms are one of the biggest successes of the field and they have many applications in other disciplines, such as graphics. One of the difficulties, however, is that the current tools cannot yet be fully automated, and they do not provide very accurate results. More research is required for automation and high precision. During the past few years we have investigated a number of basic questions underlying the structure from motion problem. Our investigations resulted in a small number of principles that characterize the problem. These principles, which give rise to automatic procedures and point to new avenues for studying the next level of the structure from motion problem, are the subject of this paper.*

1 Introduction: The problem

We are given a number of images of a scene taken at different viewpoints and the goal is to create 3D models of the scene in view. What is a geometric model of image formation? To make an image, we first pick a point in space and consider all the light rays passing through this point. Then we cut these rays with a surface. For the standard pinhole camera, this surface is a plane and images are formed by central projection on a plane (Fig. 1a). The focal length is f and the coordinate system $OXYZ$ is attached to the camera, with Z being the optical axis, perpendicular to the image plane. Image points are represented as vectors $\mathbf{r} = [x, y, f]^{\mathrm{T}}$, where x and y are the image coordinates of the point in the coordinate system oxy, with $ox \| OX$, $oy \| OY$ and O the intersection of the axis OZ with the image plane, and f is the focal length in pixels. A scene point $\mathbf{R}$ is projected onto the image point

$$\mathbf{r} = f \frac{\mathbf{R}}{\mathbf{R} \cdot \hat{\mathbf{z}}} \tag{1}$$

where $\hat{\mathbf{z}}$ is the unit vector in the direction of the Z axis.

If we cut the rays with a sphere, we obtain a spherical eye with a full field of view (Fig. 1b). In the case of video, the camera is moved to different locations while acquiring new images. Thus, video acquired by a moving camera amounts to a collection of images of a scene, i.e., projections onto an imaging surface, acquired from different viewpoints. Figuring out a model for the scene and the movement in the scene becomes a problem of relating the different projections (images) to each other.

In general, when a scene is viewed from two positions, there are two concepts of interest:

(a) The 3D transformation relating the two viewpoints. This is a rigid motion transformation, consisting of a translation and a rotation (six degrees of freedom). When the viewpoints are close together, this transformation is modeled by the 3D motion of the eye (or camera).

(b) The 2D transformation relating the pixels in the two images, i.e., a transformation that given a point in the first image maps it onto its corresponding one in the second image (that is, these two points are the projections of the same scene point). When the viewpoints are close together, this transformation amounts to a vector field denoting the velocity of each pixel, called an image motion field.

Perfect knowledge of both transformations described above leads to perfect knowledge of models of space and action. Regarding models of space, this is easy to understand. Knowing exactly how the two viewpoints and the

*The authors are with the Computer Vision Laboratory and Center for Automation Research at the University of Maryland, College Park, MD 20742-3275, USA. E-mail: fer@cfar.umd.edu

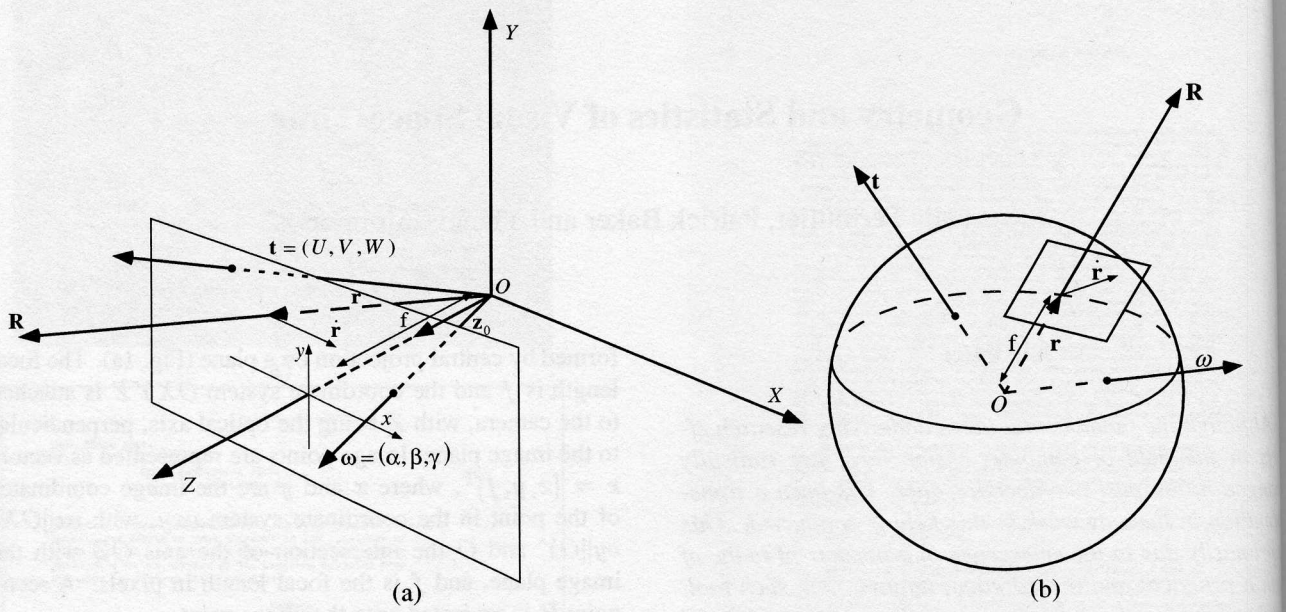


Figure 1. Image formation on the plane (a) and on the sphere (b). The system moves with a rigid motion with translational velocity $\mathbf{t}$ and rotational velocity ω. Scene points $\mathbf{R}$ project onto image points $\mathbf{r}$ and the 3D velocity $\dot{\mathbf{R}}$ of a scene point is observed in the image as image velocity $\dot{\mathbf{r}}$.

images are related provides the exact position of each scene point in space. Regarding models of action, knowing the exact velocity of each image point, by projecting it back onto the scene, for which a model is available by the previous step, we can find the *3D motion vector* for each scene point at every time instant. The sequence of evolving 3D motion fields constitutes a general model of action (since action is the extension of shape into time). Let us make these ideas more explicit. In the case where the viewpoints are close to each other, the 3D transformation becomes the camera's 3D motion, and the 2D transformation becomes an image motion field. Considering a camera with the geometric model of Fig. 1a moving in a static environment with instantaneous translation $\mathbf{t} = (U, V, W)$ and instantaneous rotation $\boldsymbol{\omega} = (\alpha, \beta, \gamma)$ (measured in the coordinate system $OXYZ$), a scene point $\mathbf{R}$ moves with velocity (relative to the camera)

$$\dot{\mathbf{R}} = -\mathbf{t} - \boldsymbol{\omega} \times \mathbf{R} \tag{2}$$

The image motion field then consists of the sum of two vector fields, one due to the translational part of the 3D motion and the other due to the rotation. Equation (1) and (2) give [1]:

$$\dot{\mathbf{r}} = -\frac{1}{(\mathbf{R} \cdot \hat{\mathbf{z}})}(\hat{\mathbf{z}} \times (\mathbf{t} \times \mathbf{r})) + \frac{1}{f}\hat{\mathbf{z}} \times (\mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r})) = \frac{1}{Z}\mathbf{u}_{\mathrm{tr}}(\mathbf{t}) + \mathbf{u}_{\mathrm{rot}}(\boldsymbol{\omega}) \tag{3}$$

where Z is used to denote the scene depth $(\mathbf{R} \cdot \hat{\mathbf{z}})$, and $\mathbf{u}_{\mathrm{tr}}, \mathbf{u}_{\mathrm{rot}}$ the direction of the translational flow and the rotational flow respectively. Due to the scaling ambiguity, only the direction of translation (focus of expansion—FOE, or focus of contraction—FOC, depending on whether the observer approaches or moves away from the scene), and the three rotational parameters can be estimated from monocular image sequences [2].

Equation (3) demonstrates model construction. If the image motion vector $\dot{\mathbf{r}}$ is known at point $\mathbf{r}$, then knowledge of $\mathbf{t}$ (up to scale) and $\boldsymbol{\omega}$ provides Z (up to scale), i.e., the depth at point $\mathbf{r}$ in the camera's coordinate system. Knowledge of Z (or, equivalently, $\mathbf{R}$) for all image points $\mathbf{r}$ provides a model for the scene in view, for the current viewpoint of the camera. Knowledge of $\mathbf{t}, \boldsymbol{\omega}$ and $\mathbf{R}$ provides then, from eq. (2), knowledge of $\dot{\mathbf{R}}$ (up to scale), that is, the 3D motion vector. A sequence of 3D motion vector fields is a model of action, as it shows how different parts of space move.

Thus, a key to the basic problem of building models of space-time is the recovery of the two transformations described before and any difficulty in building such models can be traced to the difficulty of estimating these two transformations. Of course, there exist many issues to be addressed before models can be built, but recovery of the camera's 3D motion and the image motion field are the essential prerequisites for acquiring scene depth, which is the cornerstone of the model building process.

Naturally, the community addressed the problem in the

form of three modules having a hierarchical structure. The first module worries about finding the 2D transformation relating the pixels in the different images; it deals with the correspondence problem. The second module worries about recovering the 3D transformation relating different viewpoints. And the third module uses the first two to recover the depth of the scene and subsequently surfaces and models. Our work demonstrates that these modules do not work independently of each other but they are rather components of an intricate feedback loop. Nevertheless, in order to put our contributions into the context of current work, we choose to describe them in relation to these modules.

1.1 What the paper is about

In this paper we show that there exist inherent difficulties in both the estimation of the 2D and 3D transformation, if addressed using the classic bottom-up strategy. It has been known that discontinuities in depth and motion are a problem for the estimation of image motion or correspondence. There is another problem however, which is of a statistical nature. The estimation of image features is biased and thus there is an ambiguity in the computation of image motion as well as the correspondence of points and lines. Section 2 discusses this principle through a number of well known geometric optical illusions. The inherent limitations in the estimation of correspondence suggest that the estimation of 3D motion should use as input the movement perpendicular to image edges, which, in contrast, constitute a well-defined quantity. Section 3.1 describes constraints relating this movement directly to 3D motion and structure. Section 3.2 discusses a general problem in the estimation of 3D motion from image measurements. For conventional cameras with restricted field of view there is an ambiguity in the estimation of the parameters describing the rigid transformation; translation is confused with rotation. This confusion does not exist for cameras with a full 360-degree field of view (Section 3.2.1). This has motivated our efforts to design new imaging mechanisms (sphere-like cameras) with which it is possible to obtain 3D motion very accurately (Section 3.3). Section 4 discusses issues related to the estimation of structure. Erroneous 3D motion estimates lead to a distorted structure. The consequence is that exact models of the 3D scene cannot be obtained. These studies demonstrate that the estimates of the 2D and 3D transformation are inherently coupled. The 2D transformation cannot be computed accurately without knowledge about the structure of the scene, and 3D motion and structure cannot be estimated well before the 2D transformation is available. We argue that a complete solution to structure from motion requires a synergistic approach to the estimation of the two transformations, and a plausible way to achieve this is through feedback. First, using the movement of edges, an initial estimation of 3D motion and structure is performed from video. Then, using these estimates, the system recomputes both the image transformation and 3D transformation, but now using as input features of larger extent, not only points and lines, but image patches. Section 5 discusses an approach along this direction. It introduces new constraints which relate textured image patches to 3D motion and structure.

2 The 2D transformation: Bias

If the viewpoints are far apart, then points and lines are extracted in the two images and their correspondence is estimated, thus resulting in the 2D transformation [3]. If the viewpoints are close to each other, the image motion field is an estimate of the 2D transformation. In whatever way the process of matching is done, and it is not at all clear how this process could be achieved, it has to be preceded by a step where image features such as lines, intersections of lines, or local image movement must be derived. However, as we will show next, noise in the image intensity and its derivatives causes problems in the estimation of features; in particular, it causes bias. As a result, the locations of features are estimated erroneously. The bias occurs with any visual processing of line features, thus creating problems in the estimation of the transformation relating the images in a video sequence. Under average conditions the bias is not large enough to be noticeable, but one can construct patterns where it can clearly be perceived. Incidentally, illusory patterns, known as geometric optical illusions, are such that the bias is highly pronounced. In general, this bias cannot be avoided and any vision system, biological or artificial, must cope with it. This constitutes a general uncertainty principle governing the workings of vision systems, and demonstrates that if we start the process of structure from motion by localizing points and lines and matching them in the image sequence, or if we attempt to estimate the image motion field, we will have an ambiguity.

Let us now go deeper into the nature of this uncertainty. Even if we end up using points, lines or image motion fields, we should be aware of the uncertainties that are introduced.

(a) Line localization The best way to explain this principle as far as line localization is concerned, is through geometric optical illusions. Consider Figs. 2, 3 and 4. Although in all the figures the lines are straight and parallel, they do not appear so. In Figs. 2 and 3 they appear bulging and wiggly and in Fig. 4 they appear to be tilted. The reason is due to the uncertainty principle mentioned above. There is noise in the image intensities and there are many sources of noise. For example, the lenses cause blurring, there are errors due to quantization discretization, and there are errors due to temporal integration. The effect of all these sources

of noise is equivalent to smoothing the image with a Gaussian function. There is a mathematical framework which describes images, or signals in general, under smoothing, which is called scale space analysis.

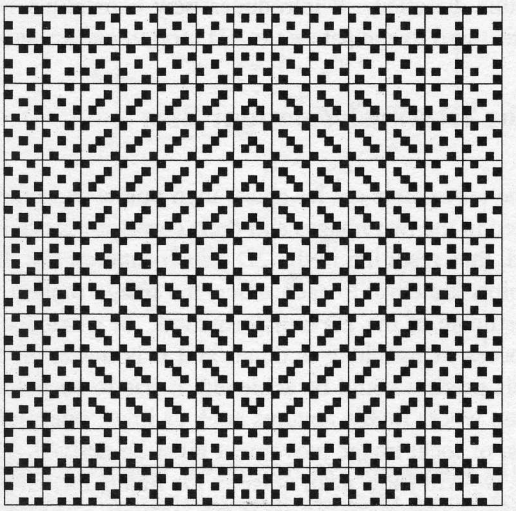


Figure 2. (a) "Spring" illusory pattern.

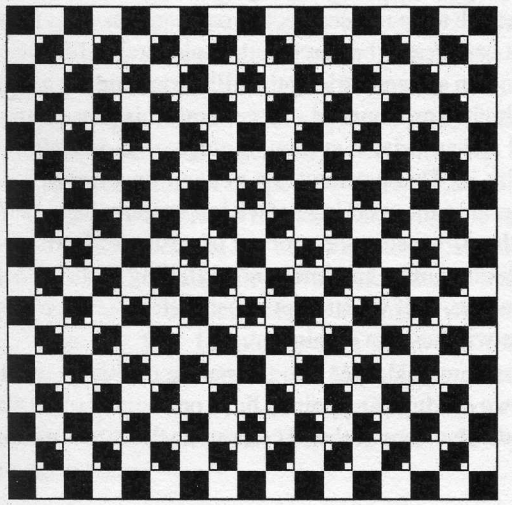


Figure 3. (a) "Waves" illusory pattern.

The scale space behavior of straight edges is illustrated in Fig. 5. There are three cases to be considered: Edges between a dark and a bright region do not change location under scale space smoothing (Fig. 5a). The two edges at the boundaries of a bright line, or bar, in a dark region (or, equivalently, a dark line in a bright region) drift apart, assuming the smoothing parameter is large enough that the

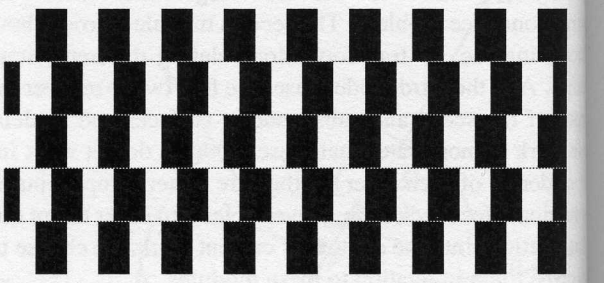


Figure 4. Café wall illusion.

whole bar affects the edges (Fig. 5b). Finally, the effect of smoothing on a line of medium brightness next to a bright and a dark region is to move the two edges towards each other (Fig. 5c).

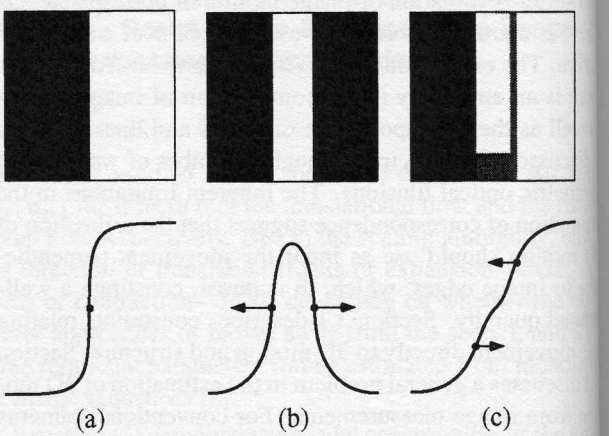


Figure 5. A schematic description of the behavior of edge movement (drift) in scale space: (a) no movement, (b) drifting apart, (c) getting closer.

This suffices to explain the main cause underlying the three illusions in Figs. 2–4, and several others [4]. Local edge elements which are tilted are estimated as illustrated in Figs. 6, 7 and 8. To fully explain the illusory perception of the wiggly and tilted lines we have to add, that local edge elements must be linked through a fitting process which considers the orientation and position of edge elements.

(b) Point localization Similarly, there is bias in the intersection of lines, i.e., points. Let us analyze the estimated position of an intersection of straight lines. Assuming the image to be $I(x, y)$, the inputs are edge elements, parame-

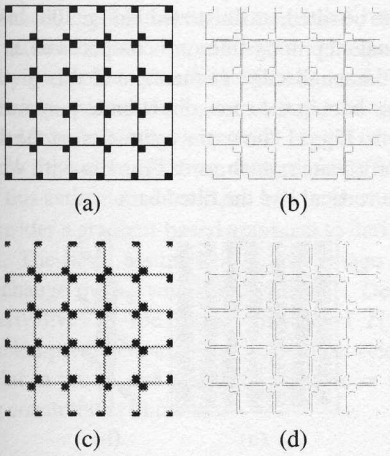


Figure 6. (a) Small part of the figure (b) edge detection without smoothing, (c) Gaussian smoothing, and (d) smoothing and edge detection have been applied.

terized by the image gradient (a vector in the direction normal to the edge) (I_x, I_y) and the position of the center of the edge element $\mathbf{x}_0 = (x_0, y_0)$.

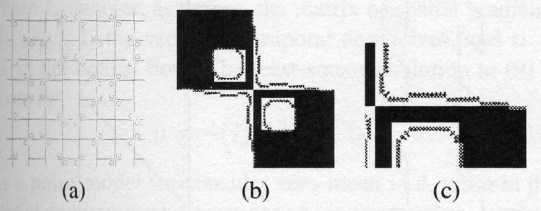


Figure 7. (a) The result of smoothing and edge detection on a part of the figure. (b) and (c) The drift velocity at edges in the smoothed image logarithmically scaled for parts of the figure.

Consider additive, independently identically distributed (i.i.d.) zero-mean noise in the parameters. In the sequel unprimed letters are used to denote estimates, primed letters to denote actual values, and δ's to denote errors, where $I_x = I'_x + \delta I_x$, $I_y = I'_y + \delta I_y$, $x_0 = x'_0 + \delta x_0$ and $y_0 = y'_0 + \delta y_0$.

For every point (x, y) on the lines the following equation holds:

$$I'_x x + I'_y y = I'_x x'_0 + I'_y y'_0 \tag{4}$$

This equation is approximated by the measurements. Let n be the number of measurements. Each measurement i provides one equation

$$I_{x_i} x + I_{y_i} y = I_{x_i} x_{0_i} + I_{y_i} y_{0_i} \tag{5}$$

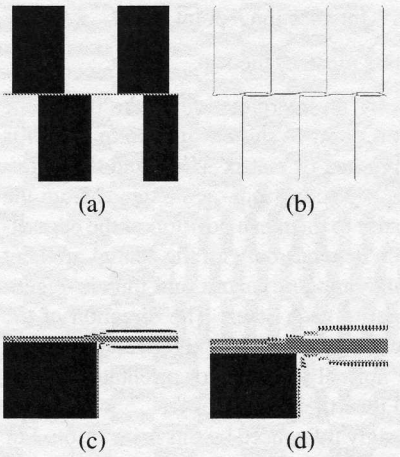


Figure 8. (a) Small part of the figure. (b) Result of smoothing and edge detection. (c) and (d) Zoom-ins on the drift velocity.

and we obtain a system of equations which are represented in matrix form as

$$I_s \mathbf{x} = \mathbf{C}$$

Here I_s is the n-by-2 matrix which incorporates the data in the I_{x_i} and I_{y_i}, and $\mathbf{C}$ is the n-dimensional vector with components $I_{x_i} x_{0_i} + I_{y_i} y_{0_i}$. The vector $\mathbf{x}$ denotes the intersection point whose components are x and y. The solution to the intersection point using standard least square (LS) estimation is given by

$$\mathbf{x} = (I_s^t I_s)^{-1} I_s^t \mathbf{C} \tag{6}$$

It is well known [5] that the LS solution to a linear system of the form $A\mathbf{x} = \mathbf{b}$ with errors in the measurement matrix A is biased. The statistics of the estimation have been studied for the case of i.i.d. noise in the parameters of A and $\mathbf{b}$. In our case $\mathbf{b}$ is the product of terms in A and two other noisy terms and thus the statistics are somewhat different.

To simplify the analysis, the variance of the noise in the spatial derivatives in the x and y directions is assumed to be the same, let it be σ_s^2, and also the expected values of higher-(than second) order terms are assumed to be negligible. In [6] the expected value of $\mathbf{x}$ is found by developing (6) into a second-order Taylor expansion at zero noise. It converges in probability to

$$\text{plim}_{n\to\infty} \mathbf{x} = \mathbf{x}' + nM'^{-1}(\bar{\mathbf{x}}_0 - \mathbf{x}')\sigma_s^2 \tag{7}$$

where

$$M' = I_s'^t I_s' = \begin{bmatrix} \sum_{i=1}^n I'^2_{x_i} & \sum_{i=1}^n I'_{x_i} I'_{y_i} \\ \sum_{i=1}^n I'_{x_i} I'_{y_i} & \sum_{i=1}^n I'^2_{y_i} \end{bmatrix}$$

$\mathbf{x}'$ is the actual intersection point and $\bar{\mathbf{x}}_0 = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n} x_{0_i} \\ \frac{1}{n}\sum_{i=1}^{n} y_{0_i} \end{bmatrix}$ is the mean of the $\mathbf{x}_{0_i}$.

Using (7) allows for an interpretation of the bias. The estimated intersection point is shifted by a term which is proportional to the product of matrix M'^{-1} and the difference vector $(\bar{\mathbf{x}}_0 - \mathbf{x}')$. Vector $(\bar{\mathbf{x}}_0 - \mathbf{x}')$ extends from the actual intersection point to the mean position of the edge elements. M'^{-1}, which depends only on the spatial gradient distribution, is a real symmetric matrix and thus its eigenvectors are orthogonal to each other. The direction of the eigenvector corresponding to the larger eigenvalue of M'^{-1} is dominated by the normal to the major orientation of the image gradients and thus the product of M'^{-1} with vector $(\bar{\mathbf{x}}_0 - \mathbf{x}')$ is most strongly influenced by this orientation. For the case of two intersecting lines in an acute angle, the intersection is between the lines, the size of the bias decreases as the angle increases, and there is more displacement of the intersection point in the direction perpendicular to the line with fewer edge elements.

The best known illusions due to intersecting lines are the Poggendorff and Zöllner illusions. A version of the Poggendorff illusion as described by Zöllner is displayed in Fig. 9 (for an interactive version see [7]). The upper-left portion of the interrupted, tilted straight line in this figure is apparently not the continuation of the lower portion on the right, but is too high.

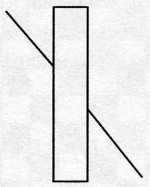
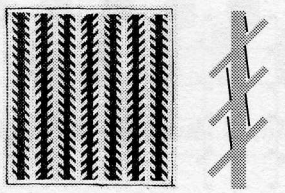


Figure 9. Poggendorff illusion.

The phenomenon is explained by the bias in the estimation of the intersection point. Referring to Fig. 9, the intersection point of the left vertical with the upper tilted line is moved up and to the left, and the intersection point of the right vertical with the lower tilted line is moved down and to the right. As a result the two line segments appear to be shifted in opposite directions and not to lie on the same line anymore. The model also predicts the findings of many parametric studies, for example, findings regarding the change in the size of the illusory percept with a change in the angle of the intersecting lines and the orientation of the figure [4].

Fig. 10 shows a version of the Zöllner illusion. The vertical bands in Fig. 10 are all parallel, but they look convergent or divergent. The biases in the intersection points of the edges of the bands with the short line segments cause the edge elements along the long edges between intersection points to be tilted, as illustrated in Fig. 10b. In a second computational step, long lines are computed as an approximation to the small edge elements, and this gives rise to tilted lines or bars in the same direction as perceived by the visual system. Fig. 11 shows the estimation of the tilted line elements for a pattern such as in Fig. 10a with 45 degrees between the vertical and the tilted bars.



(a) (b)

Figure 10. (a) Zöllner pattern. (b) The bias in the intersection points of the edges causes the line elements between intersection points to be tilted.

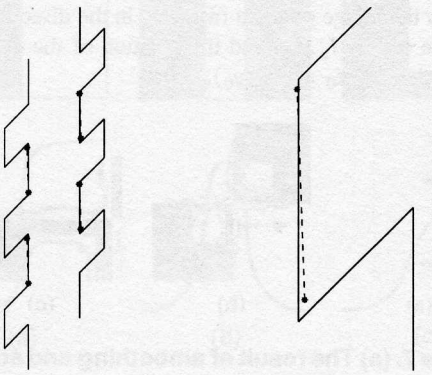


Figure 11. Estimation of edges in Zöllner pattern. The line elements are found by connecting two consecutive intersection points, resulting from the intersection of edges of two consecutive bars with the edge of the vertical bar (one in an obtuse and one in an acute angle). The data consists of edge elements uniformly distributed on the vertical and on the tilted line (with 1.5 times more elements on the vertical).

(c) Image motion The basic image representation when the viewpoints are close to each other, is the optical flow. Optical flow is derived in a two-stage process. In a first stage the velocity components perpendicular to linear features are computed from local image measurements. In the

computational literature this one-dimensional velocity component is referred to as "normal flow" and the ambiguity in the velocity component parallel to the edge is referred to as the "aperture problem." In a second stage the optical flow is estimated by combining, in a small region of the image, normal flow measurements from features in different directions, but this estimate is biased, as will be shown.

We consider a gradient-based approach to derive the normal flow. The basic assumption is that image gray level does not change over a small time interval. Denoting the spatial derivatives of the image gray level $I(x, y, t)$ by I_x, I_y, the temporal derivative by I_t, and the velocity of an image point in the x- and y-directions by $\mathbf{u} = (u, v)$, the following constraint is obtained:

$$I_x u + I_y v + I_t = 0 \tag{8}$$

This equation, called the optical flow constraint equation, defines the component of the flow in the direction of the gradient [8]. We assume the optical flow to be constant within a region. Each of the n measurements in the region provides an equation of the form (8) and thus we obtain the over-determined system of equations

$$I_s \mathbf{u} + \mathbf{I}_t = 0, \tag{9}$$

where I_s denotes, as before, the matrix of spatial gradients (I_{x_i}, I_{y_i}), $\mathbf{I}_t$ the vector of temporal derivatives, and $\mathbf{u} = (u, v)$ the optical flow. The least-squares solution to (9) is given by

$$\mathbf{u} = -(I_s^T I_s)^{-1} I_s^T \mathbf{I}_t. \tag{10}$$

As a noise model we consider zero-mean i.i.d. noise in the spatial and temporal derivatives. As in the previous section, we assume equal variance σ_s^2 for the noise in the spatial derivatives in the two directions and we assume that higher than second-order noise terms can be ignored.

The statistics of (10) are well understood, as these are classical linear equations. The expected value of the flow, using a second-order Taylor expansion, converges in probability to

$$\text{plim}_{n\to\infty} \mathbf{u} = \mathbf{u}' - n\sigma_s^2 M'^{-1} \mathbf{u}', \tag{11}$$

where, as before, the actual values are denoted by primes.

Equation (11) is very similar to (7) and shows the bias depends on the gradient direction (that is, the texture) in the region. The estimated flow is always underestimated in length and its orientation is biased towards the majority of gradients.

Fig. 12 shows a variant of a pattern created by Hajime Ouchi [9]. The pattern consists of two rectangular checkerboard patterns oriented in orthogonal directions—a background orientation surrounding an inner ring. Small retinal

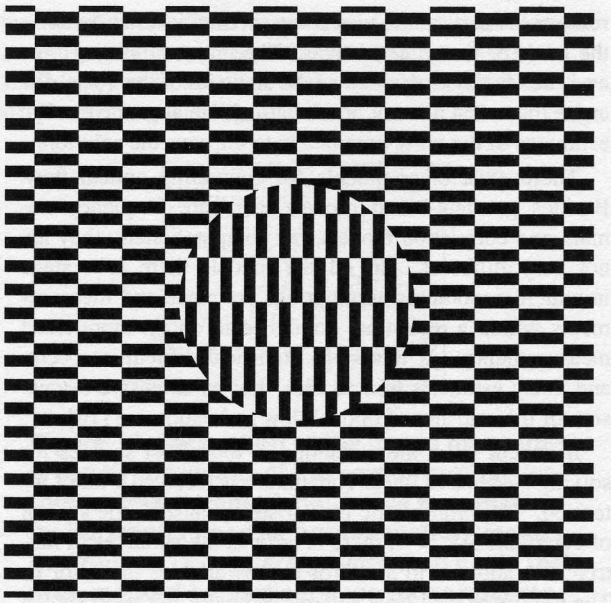


Figure 12. A pattern similar to the one by Ouchi.

motions, or slight movements of the paper, cause a segmentation of the inset pattern, and motion of the inset relative to the surround.

The tiles used to make up the pattern are longer than they are wide, leading to a gradient distribution in a small region with many more normal flow measurements in one direction than the other. Since the tiles in the two regions of the figure have different orientations, the estimated regional optical flow vectors are different. The difference between the bias in the inset and the bias in the surrounding area is interpreted as motion of the ring. In addition to computing flow, the visual system also performs segmentation, which is why a clear relative motion of the inset is seen.

2.1 The inherent problem

An important question arises. Is there bias because of the architecture of the vision system? Is the bias due to the linear estimation only? Could it be corrected using more sophisticated statistical techniques, or could it be avoided?

It is well known that linear estimation is biased if there are errors in all the measurement variables, but any method of compensating for the bias requires knowledge of the statistics of the noise. In the noise models considered in the previous sections, this amounts to knowledge of the covariance matrix of the noise. If this were available, inverse filters could be applied to reconstruct the gray level signal, and the corrected least squares estimator could be used to

remove the asymptotic bias when solving linear systems on the basis of image derivatives. The major problem, however, lies in the acquisition of the statistics of the noise. We argue that often it is not possible to obtain accurate enough estimates of the noise parameters to improve the solution.

There exist other models for computing optical flow. Besides gradient-based models there are frequency domain and correlation models, but computationally they are not very different. In all the models there is a stage in which smoothness assumptions are made and measurements within a region are combined to obtain more exact measurements. At this stage statistical difficulties occur, and noisy estimates lead to bias. For an extensive discussion of the statistics of optical flow estimation see [10].

In recent years the technique of total least squares for solving systems of linear equations has received a lot of attention. The problematic bias arises because in the system of equations $A\mathbf{x} = \mathbf{b}$, there is error in the variables in matrix A in addition to the error in the variables in vector $\mathbf{b}$. The nonlinear total least squares estimator has been shown to provide an asymptotically unbiased solution for such systems, if the noise variables are independent and identically distributed. This means that we have to know the relative amounts of noise in the error variables, that is, the ratios of the two spatial and temporal derivative noise terms or the noise in position, but information about noise ratios is difficult to compute. It can be obtained only from the variation in the estimated variables over the image. There is another problem with this technique: the variance is larger. Total least squares is known to perform very poorly if outliers are present, and these are difficult to detect from a few measurements.

Why is it so difficult to obtain accurate estimates of the noise parameters? To acquire a good noise statistic a lot of data is required, so data needs to be taken from large spatial areas acquired over a period of time, but the models used for the estimation can only be assumed to hold locally. Thus to integrate more data, models of the scene need to be acquired. Specifically, long edges and bars need to be detected, and in the case of motion, discontinuities due to changes in depth and differently moving entities need to be detected and the scene segmented.

If the noise parameters stayed fixed for extended periods of time it would be possible to acquire enough data to closely approximate these parameters, but usually the noise parameters do not stay fixed long enough. Sensor characteristics may stay fixed, but there are many other sources of noise besides sensor noise. The lighting conditions, the physical properties of the objects being viewed, the orientation of the viewer in 3D space, and the sequence of eye movements all have influences on the noise. Aside from all these factors, in order to estimate derivatives (or to compute Fourier transforms) the system needs to interpolate. The accuracy of interpolation can depend in complex ways on the pattern of gray levels in the image.

Thus, it appears that it is very hard to deal with the structure from motion problem using the local measurements of the sort considered. In the case where the cameras are far apart, we would need to make measurements in a whole image patch. This is, at the very least, a sensible alternative, if we need to move beyond points and lines. In the case where the cameras are close together, we have at least the option of considering local image motion measurements that are perpendicular to edges, the so-called normal flow. Unbiased estimates for normal flow are possible. The sensitivity of the problem, however, suggests that we should be looking for constraints that are of a global nature, so that little local mistakes should not matter.

Let us then concentrate on the second module, the one devoted to estimating 3D motion. Addressing the problem starting with the normal flow values, that is, without attempting to estimate correspondence or flow at the beginning stages, is known under the heading of "direct methods." Work on this subject is sparse and mainly due to two research groups [11, 12, 13, 1, 14].

In the sequel we develop global constraints on the basis of normal flow, i.e., constraints involving quantities that are the outputs of filters with finite extent (values in patches).

3 The 3D transformation

3.1 The visibility constraint

If $\mathbf{u}$ is the motion vector at a point (x, y) and $\mathbf{n} = (n_x, n_y)$ is a unit vector in gradient direction, the normal motion $\mathbf{u_n}$ is

$$\mathbf{u_n} = (\mathbf{u} \cdot \mathbf{n}) \cdot \mathbf{n}.$$

Let us substitute for the components of $\mathbf{u}$ from (3), with $(x_0, y_0) = (\frac{Uf}{W}, \frac{Vf}{W})$, the focus of expansion (i.e., the point where $\mathbf{t}$ pierces the image plane), and $\mathbf{u}_{\rm tr} = (u_{\rm tr}, u_{\rm tr}), \mathbf{v}_{\rm rot} = (v_{\rm rot}, v_{\rm rot})$. The point where the axis of rotation pierces the image will be denoted by AOR (direction of rotation). We obtain u_n for the value of the vector $\mathbf{u_n}$ along the gradient direction:

$$u_n = u_{\rm rot} n_x + u_{\rm tr} n_x + v_{\rm rot} n_y + v_{\rm tr} n_y \qquad (12)$$

and thus

$$\begin{aligned} &\frac{W}{Z}\left((x - x_0)\, n_x + (y - y_0)\, n_y\right) \\ &\quad = u_n - \left(\alpha \frac{xy}{f} - \beta\left(\frac{x^2}{f} + f\right) + \gamma y\right) n_x \\ &\quad - \left(\alpha\left(\frac{y^2}{f} + f\right) - \beta \frac{xy}{f} - \gamma x\right) n_y. \end{aligned} \qquad (13)$$

We are concerned with the question: Where in the image plane can the normal motion field take on a certain constant value $\mathbf{u_n}$ (i.e., where could normal motion vectors of a certain length u_n and direction (n_x, n_y) be?)? The only constraint that we can apply is that the depth has to be positive because the scene is in front of the camera. If we assume $W > 0$, we obtain the following inequality. This is the visibility constraint also known as the positive depth constraint.

$$\left[u_n - \left(\alpha\frac{xy}{f} - \beta\left(\frac{x^2}{f} + f\right) + \gamma y\right) n_x - \left(\alpha\left(\frac{y^2}{f} + f\right) - \beta\frac{xy}{f} - \gamma x\right) n_y\right] \cdot [(x - x_0)\, n_x + (y - y_0)\, n_y] > 0$$

$$h(u_n, \alpha, \beta, \gamma, x, y) \cdot g(x_0, y_0, x, y) > 0 \quad (14)$$

$h(u_n, \alpha, \beta, \gamma, x, y) = u_n - (\alpha\frac{xy}{f} - \beta(\frac{x^2}{f} + f) + \gamma y)n_x - (\alpha(\frac{y^2}{f} + f) - \beta\frac{xy}{f} - \gamma x)n_y$ and $g(x_0, y_0, x, y) = (x - x_0)n_x + (y - y_0)n_y$. The equation $h(x, y) = 0$ describes a hyperbola that splits the image plane in an area where $h(x, y) > 0$ and an area where $h(x, y) < 0$. The equation $g(x, y) = 0$ describes a line through the FOE, which is perpendicular to (n_x, n_y), and which separates the plane into an area where $g(x, y)$ is positive and an area where $g(x, y)$ is negative. Thus, through this inequality a region $I_{\mathbf{u_n}}$ consisting of two areas bounded by a hyperbola and a line are defined as the locations where the normal motion could take on a certain value $\mathbf{u_n}$. The two areas meet at one point, the intersection of the hyperbola and the line. This point, which contains information about the whole pattern, will be denoted by $\mathbf{S_{u_n}}$ (see Fig. 13).

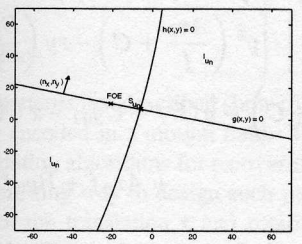


Figure 13. Iso-normal motion regions are bounded by a line ($g(x, y) = 0$) and a hyperbola ($h(x, y) = 0$).

In the other areas of the image plane the value of the motion vectors in direction (n_x, n_y) is constrained. Where $h(x, y) < 0$ we have $u_n > \mathbf{u_{rot}} \cdot \mathbf{n}$ (i.e., the rotational component of the normal motion is greater than u_n). Where $g(x, y) > 0$ the translational component of the normal motion is greater than zero. In the area where $h(x, y) < 0$ and $g(x, y) > 0$, we can thus conclude that the normal motion (the sum of the rotational and translational components) is greater than u_n. Similarly, where $h(x, y) > 0$ and $g(x, y) < 0$, the value of the normal motion has to be smaller than u_n (see Fig. 14).

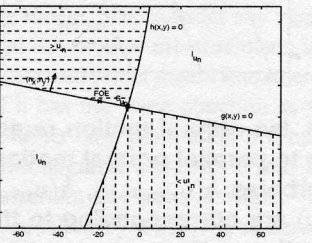


Figure 14. Separation of the image plane into areas by the values of the normal motion vectors in certain directions: In the area marked by horizontal lines all normal motion vectors in direction (n_x, n_y) are greater than u_n. In the area marked by vertical lines all normal motion vectors in direction (n_x, n_y) are smaller than u_n. Vectors of length u_n in direction (n_x, n_y) can only be in the complementary areas (the region $I_{\mathbf{u_n}}$).

To summarize these results, considering for a given normal motion field due to rigid motion $\mathbf{t}$ and $\boldsymbol{\omega}$ the vectors along the gradient direction (n_x, n_y), we find that the image plane is split by a hyperbola and a line into four areas. All vectors which are of length u_n are in two opposite areas. One of the two other areas contains only values greater than u_n, and the other only values smaller than u_n.

The line $(g(x, y) = 0)$ is defined by the translational motion; it passes through the FOE and is perpendicular to the gradient (n_x, n_y) of the normal motion vector. Therefore, this line is described by only one unknown parameter (its direction is known). Furthermore, the line is independent of u_n, the value of the normal motion vector. For any general $\mathbf{u_n}$ the hyperbola $(h(u_n, x, y))$ is defined by the three rotational parameters. For the case when $u_n = 0$, the number of unknowns reduces to two ($\frac{\alpha}{\gamma}$ and $\frac{\beta}{\gamma}$ expressing the direction of the rotation axis). If we consider parallel normal motion vectors, i.e., normal motion vectors of value $k(u_n, v_n)$, where k any scalar, we find areas in the image plane which are bounded by a line that is the same for all values and hyperbolas which differ only in their linear terms (see Fig. 15).

3.1.1 Coaxis and copoint vectors

In this section the concept of selection of normal motion vectors of a given length and direction is generalized. Instead of considering vectors of the same value, we examine

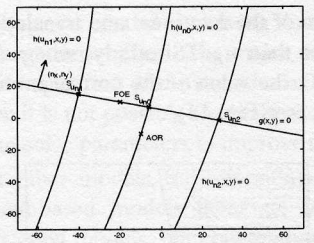


Figure 15. Iso-normal motion regions corresponding to parallel normal motion vectors: The hyperbolas $h(\mathbf{u_{n0}}, x, y)$, $h(\mathbf{u_{n1}}, x, y)$, and $h(\mathbf{u_{n2}}, x, y)$ are corresponding to the parallel normal motion vectors $\mathbf{u_{n0}}$, $\mathbf{u_{n1}}$, and $\mathbf{u_{n2}}$. The length of $\mathbf{u_{n0}}$ is zero, thus $h(\mathbf{u_{n0}}, x, y)$ passes through the AOR. The line $g(x, y) = 0$ is independent of the length of the normal motion vector and thus the same for all parallel normal motion vectors.

various classes of vector-valued functions. In particular, we investigate the coaxis and copoint vectors [15, 16].

The copoint vectors are defined with respect to a point. The (r, s) copoint vectors are defined as the normal vectors which are perpendicular to straight lines passing through the point (r, s). An (r, s) copoint vector at a point (x, y) is parallel to the vector $(s - y, x - r)$ (see Fig. 16). It becomes clear that the normal motion vectors of same length and direction can be considered as special cases of the copoint vectors. They represent the copoint vectors, for which r and s both are ∞ and $\frac{r}{s} = \frac{-n_y}{n_x}$.

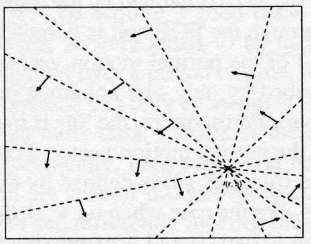


Figure 16. Copoint vectors (r, s).

The coaxis vectors are defined with respect to a direction in space. The (A, B, C) coaxis vectors are defined as follows: A line through the image formation center defined by the directional cosines (A, B, C) defines a family of cones with axis (A, B, C) and apex at the origin. The intersection of these cones with the image plane gives rise to conic sections. The normal vectors perpendicular to these conic sections are called (A, B, C) coaxis vectors . At every point (x, y) a coaxis vector is parallel to the vector $(-A(y^2 + f^2) + Bxy + Cx, Axy - B(x^2 + f^2) + Cy)$ (see Fig. 17).

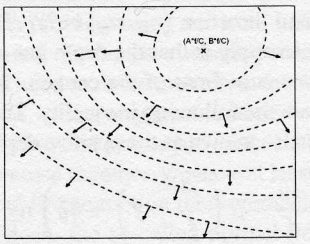


Figure 17. (A, B, C) coaxis vectors.

As in the case of the iso-normal motion vectors, we choose vectors of a given length and direction and evaluate the regions with positive depth measurements. We consider the (r, s) copoint vectors and the (A, B, C) coaxis vectors of length $u_n(x, y)$ (with $u_n(x, y)$ a function in x and y). Where $\frac{W}{Z} > 0$ the following inequalities hold:

$$[y(x_0 - r) - x(y_0 - s) - x_0 s + y_0 r] \cdot \left[u_n(x, y) - \frac{x^2}{f}(\beta s + \gamma f) - \frac{y^2}{f}(\alpha r + \gamma f) + xy\left(\frac{\alpha s}{f} + \frac{\beta r}{f}\right) + y(\gamma s + \beta f) + x(\gamma r + \alpha f) - (\alpha f r + \beta f s)\right] > 0 \quad (15)$$

$$\left[u_n(x, y) - \left(y\left(\frac{C\alpha}{f} - \frac{A\gamma}{f}\right) + x\left(\frac{B\gamma}{f} - \frac{C\beta}{f}\right) + A\beta - B\alpha\right)(x^2 + y^2 + f^2)\right] \cdot \left[y^2\left(\frac{Ax_0}{f} + C\right) - xy\left(\frac{Bx_0}{f} + \frac{Ay_0}{f}\right) + x^2\left(\frac{By_0}{f} + C\right) - y(Bf + Cy_0) - x(Af + Cx_0) + Ax_0 f + By_0 f\right] > 0 \quad (16)$$

For $u_n(x, y) = 0$ we obtain regions defined by a line and a conic section. In the case of the copoint vectors the line separates the translational components and the conic separates the rotational components of the vectors. The line passes through the FOE and also through the point (r, s), and thus it can be described by only one unknown. The conic is specified by only two unknowns, $\frac{\alpha f}{\gamma}$ and $\frac{\beta f}{\gamma}$. In the case of the coaxis vectors, the line separates the rotational components. It passes through the AOR and through the point (r, s) and thus it is also described by only one unknown. The conic separates the translational components. It is defined by the two coordinates of the FOE, (x_0, y_0). One of the two other regions defined by the above described

curves contains only vectors of length greater than zero, and the other contains only vectors smaller than zero. An illustration is given in Fig. 18, which shows these regions for the class of coaxis vectors displayed on Fig. 17. We call these structures on the image plane which define regions containing vectors of positive, negative or zero value, coaxis patterns and copoint patterns.

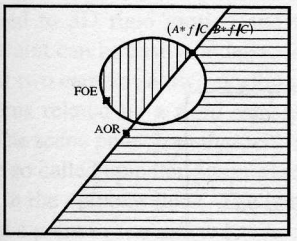


Figure 18. Separation of (A, B, C) coaxis vectors whose directions are shown in Fig. 17: A line passing through the AOR separates the positive and negative rotational components. A conic through the FOE separates the positive and negative translational components. In the area marked by horizontal lines all (A, B, C) coaxis vectors are greater than zero, i.e., the sign of the flow along the direction of the coaxis vectors is positive. In the area marked by vertical lines all (A, B, C) coaxis vectors are smaller than zero. The two other areas contain all the (A, B, C) coaxis vectors of length zero.

3.1.2 Algorithms

The geometric patterns just described show that the 3D motion is globally encoded in a motion field. This gives rise to pattern recognition algorithms for recovering 3D motion. There is an interesting way to design such patterns so that the directions of the translation $\mathbf{t}$ and rotation $\boldsymbol{\omega}$ vectors (points FOE and AOR) can be recovered, using very simple measurements, namely the sign of the flow along different directions. Consider a point (small patch) in the image. It is safe to say that today we have a number of signal processing techniques available (See [12] for a review) that can estimate the sign of the image movement along some orientation vector l. This is equivalent to deciding whether the angle between the actual flow and the vector l is less than, greater than, or equal to 90 degrees. There will be orientations for which we will be able to find this sign very reliably. When the flow has a very small positive or negative value, it will not be easy to distinguish the real value and there will be some uncertainty. Nevertheless, this operation can be performed for all image points and for a very large number of orientations, thus obtaining a repository of local image motion information, let's call it V1. Then we can pick any coaxis or copoint vector field and calculate its sign by comparing with the repository already built. Indeed, place any vector field on the image, and consider any vector $\mathbf{v}$ of the field. Now look at the motion measurements in V1 at the point in the base of $\mathbf{v}$ and select the measurement in the orientation of $\mathbf{v}$. If it has the same direction as $\mathbf{v}$, then call $\mathbf{v}$ positive and color its base point blue; if it has the opposite direction, call it negative and color it red. If it is close to zero, call it uncertain and color it white. Thus, given any video sequence taken by a moving camera, by selecting a vector field (coaxis or copoint) we can transform the original video into a new video containing only three colors, red, blue and white. See, for example, for the video in Video 1 [http://www.cfar.umd.edu/users/yiannis/procieee/video01.mpg] (See Fig. 19 for a frame), by selecting coaxis fields as in Fig. 20, we obtain the new videos in Video 2 [video02a.mpg, video02b.mpg, video02c.mpg] (See Fig. 21 for a frame). We already know, however, that in such a three-color image, there is a pattern of positive and negative areas, defined by a straight line and a conic, that depends only on the underlying 3D motion and not on the scene. By finding these patterns, we localize the FOE and AOR. In a perfect situation, these points are found through the intersection of the straight lines and conics that constitute the pattern (See Fig. 22). In an actual situation, instead of curves there will be bands of uncertainty and instead of a point signifying the direction of $\mathbf{t}$ or $\boldsymbol{\omega}$, there will be an area where the FOE or AOR will lie. Given that we can search for a very large number of patterns, the final solution will be an area containing the FOE and another area containing the AOR. But even if we have excellent signal processing algorithms and we obtain very small uncertainty in the local image motion measurements, we will still obtain an area as a solution. This may be a small area or it may appear somewhat elongated, and this is the best one can do regarding 3D motion recovery using sign of flow along a direction (1 bit).

On the other hand, traditional epipolar minimization approaches provide a specific answer for the 3D motion. What is not known, however, is that the solution obtained with this technique, a result of a minimization process, could be anywhere inside some area if the initial conditions of the minimization are slightly perturbed. In other words, whether we utilize the visibility constraint with the values of the normal flow or we employ the epipolar constraint with image correspondence, we will have uncertainty. The next section makes this explicit.

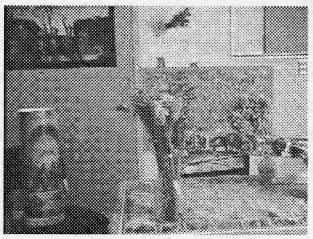


Figure 19.

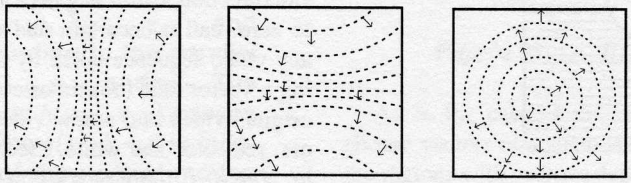


Figure 20. Coaxis vectors when the axis is the x, y or z-axis of the camera coordinate system. They are known as the α, β and γ vectors, respectively.

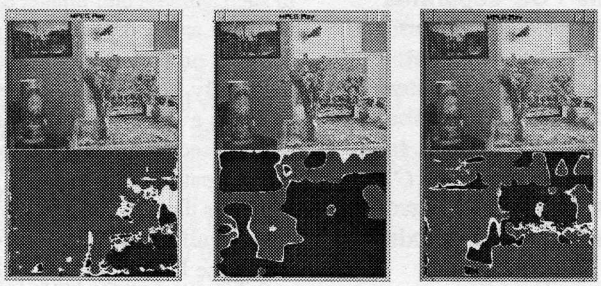


Figure 21.

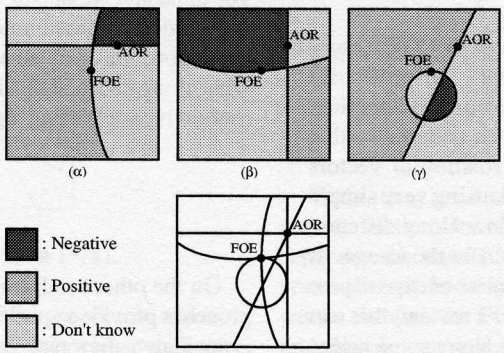


Figure 22. α, β and γ patterns. (α) Image motion measurements along the α-vectors form patterns of positive and negative values which are defined by a horizontal straight line and a hyperbola. (β) β-patterns are defined by a vertical straight line and a hyperbola and (γ) γ-patterns are defined by a straight line through the image center and a circle passing through the center of the image. The intersection of the straight lines gives the AOR and the intersection of the conics gives the FOE.

3.2 The ambiguity

Let's assume that, despite the problems mentioned, a motion field can be estimated to some degree of accuracy, and thus optic flow is available. There exists a veritable cornucopia of techniques for finding 3D motion from optic flow [17]. Almost all techniques are based on the so-called epipolar constraint, which shows how the motion of image points is related to 3D rigid motion and the scene. The epipolar constraint can be easily understood in the discrete case. Consider two cameras at two positions, with their coordinate systems related by a rigid transformation, and a scene point. The scene point, together with the camera centers define the so called epipolar plane which intersects the image planes in the epipolar lines. The epipolar constraint then states that a point in one image has to be matched with a point lying on the corresponding epipolar line in the other image. Deviation from the epipolar constraint is the epipolar error. Minimization of epipolar errors is the basis of most 3D motion estimation algorithms. For the differential case of video, the epipolar constraint is obtained from the image motion equations as $(\mathbf{t} \times \mathbf{r}) \cdot (\dot{\mathbf{r}} + \boldsymbol{\omega} \times \mathbf{r}) = 0$ [18]. One is interested in the estimates of translation $\hat{\mathbf{t}}$ and rotation $\hat{\boldsymbol{\omega}}$ which best satisfy the epipolar constraint at every point $\mathbf{r}$ according to some criteria of deviation. Usually the Euclidean norm is considered leading to the minimization of function.

$$M_{ep} = \iint_{\text{image}} \left[(\hat{\mathbf{t}} \times \mathbf{r}) \cdot (\dot{\mathbf{r}} + \hat{\boldsymbol{\omega}} \times \mathbf{r})\right]^2 d\mathbf{r} \qquad (17)$$

Experience has shown that estimating 3D motion by minimizing the above functional, or variations of it, is a very difficult problem. One main reason for this difficulty has to do with the apparent confusion between translation and rotation in the motion field. This is easy to understand at an intuitive level. If we look straight ahead at a shallow scene, whether we rotate around our vertical axis or translate parallel to the scene, the motion field at the center of the image is very similar in the two cases. Thus, for example, translation along the x axis is confused with rotation around the y axis. The basic understanding of this confusion has attracted few investigators over the years. (See [19, 18] for a review.) It has been shown that the confusion exists no matter what estimator is used, proving that there is an inherent limitation to the estimation of 3D motion from data of only a limited field of view. This has been shown in [20] through a statistical analysis of all the possible computational models that can be used to derive 3D motion is possible to perform. Next, this analysis is carried out for the classic epipolar minimization.

Any approach to 3D motion estimation using as input optic flow would minimize function (17). Thus, we perform a topographic analysis of the five-dimensional surface described by this function (two dimensions for $\mathbf{t}/|\mathbf{t}|$ and three for $\boldsymbol{\omega}$). We want to know how the valleys of (17) are structured and what the properties of the minima are at the locations that will be found by different estimators. Specifically, we are interested in the relationship between the 3D motion errors in the minima of (17). Expressing $\dot{\mathbf{r}}$ in terms of the real motion, function (17) can be expressed in terms of the actual and estimated motion parameters $\mathbf{t}$, $\boldsymbol{\omega}$, $\hat{\mathbf{t}}$ and $\hat{\boldsymbol{\omega}}$ (or, equivalently, the actual motion parameters $\mathbf{t}, \boldsymbol{\omega}$ and the errors $\mathbf{t}_\epsilon = \mathbf{t} - \hat{\mathbf{t}}$, $\boldsymbol{\omega}_\epsilon = \boldsymbol{\omega} - \hat{\boldsymbol{\omega}}$) and the depth Z of the viewed scene. To conduct any analysis, a model for the scene is needed. We are interested in the statistically expected values of the motion estimates resulting from all possible scenes. Thus, as our probabilistic model we assume that the depth values of the scene are uniformly distributed between two arbitrary values $Z_{\min}$ and $Z_{\max}$ $(0 < Z_{\min} < Z_{\max})$.

Thus, we obtain the function

$$E_{ep} = \int_{Z=Z_{\min}}^{Z=Z_{\max}} M_{ep} dZ \qquad (18)$$

measuring deviation from the epipolar constraint. Since for the scene in view we employ a probabilistic model, the results are of a statistical nature, that is, the geometric constraints between $\mathbf{t}_\epsilon, \boldsymbol{\omega}_\epsilon$ at the minima of (18) that we shall uncover should be interpreted as being likely to occur. Our approach expresses function (18) in terms of $\mathbf{t}$, $\boldsymbol{\omega}$, $\mathbf{t}_\epsilon$ and $\boldsymbol{\omega}_\epsilon$ and finds the conditions that $\mathbf{t}_\epsilon$ and $\boldsymbol{\omega}_\epsilon$ satisfy at the local minima which represent solutions of the different estimation algorithms. Procedures for estimating 3D motion can be classified into those estimating either the translation or rotation as a first step and the remaining component (that is, the rotation or translation) as a second step, and those estimating all components simultaneously. Procedures of the former kind result when systems utilize inertial sensors which provide them with estimates of one of the components of the motion, or when two-step motion estimation algorithms are used.

Thus, three cases need to be studied: the case were no prior information about 3D motion is available and the cases where an estimate of translation or rotation is available with some error. Imagine that somehow the rotation has been estimated, with an error $\boldsymbol{\omega}_\epsilon$. Then our function becomes two-dimensional in the variables $\mathbf{t}_\epsilon$ and represents the space of translational error parameters corresponding to a fixed rotational error. Similarly, given a translational error $\mathbf{t}_\epsilon$, the functions become three-dimensional in the variables $\boldsymbol{\omega}_\epsilon$ and represent the space of rotational errors corresponding to a fixed translational error. To study the general case, one needs to consider the lowest valleys of the functions in 2D subspaces which pass through 0. In the image processing literature, such local minima are often referred to as ravine lines or courses.

The following convention is employed. We use letters with hat signs to represent estimated quantities, unmarked letters to represent the actual quantities and the subscript "ϵ" to denote errors, where the error quantity is defined as the actual quantity minus the estimated one. For example, $\mathbf{u}_{\text{rot}}(\boldsymbol{\omega})$ represents actual rotational flow, $\mathbf{u}_{\text{rot}}(\hat{\boldsymbol{\omega}})$ estimated rotational flow, $\mathbf{t}_\epsilon$ the translational error vector, $x_{0_\epsilon} = x_0 - \hat{x}_0, \alpha_\epsilon = \alpha - \hat{\alpha}$, etc.

Let $\mathbf{t} = (x_0, y_0, 1)$ and $\boldsymbol{\omega} = (\alpha, \beta, \gamma)$. Assuming a small field of view, the quadratic terms in the image coordinates are very small relative to the linear and constant terms, and are therefore ignored. The case of noise-free flow is studied, in which case the analysis becomes a study of the inherent geometric confusion between rotation and translation.

Considering a circular aperture of radius e, setting the focal length $f = 1, W = 1$ and $\hat{W} = 1$, the function in (18) becomes

$$E_{ep} = \int_{Z=Z_{\min}}^{Z_{\max}} \int_{r=0}^{e} \int_{\phi=0}^{2\pi} \left\{ \left(\left(\frac{x - x_0}{Z} - \beta_\epsilon + \gamma_\epsilon y \right) (y - \hat{y}_0) - \left(\frac{y - y_0}{Z} + \alpha_\epsilon - \gamma_\epsilon x \right) (x - \hat{x}_0) \right)^2 r \right\} dr\, d\phi\, dZ$$

where (r, ϕ) are polar coordinates ($x = r\cos\phi, y = r\sin\phi$). Performing the integration, one obtains

$$E_{ep} = \pi e^2 \left((Z_{\max} - Z_{\min}) \left(\frac{1}{3}\gamma_\epsilon^2 e^4 + \frac{1}{4}\left(\gamma_\epsilon^2 \left(\hat{x}_0^2 + \hat{y}_0^2 \right) + 6\gamma_\epsilon \left(\hat{x}_0\alpha_\epsilon + \hat{y}_0\beta_\epsilon \right) + \alpha_\epsilon^2 + \beta_\epsilon^2 \right) e^2 (\hat{x}_0\alpha_\epsilon + \hat{y}_0\beta_\epsilon)^2 \right) + (\ln(Z_{\max}) - \ln(Z_{\min})) \left(\frac{1}{2}(3\gamma_\epsilon(x_{0_\epsilon}y_0 - y_{0_\epsilon}x_0) + x_{0_\epsilon}\beta_\epsilon - y_{0_\epsilon}\alpha_\epsilon)e^2 + 2\left(x_{0_\epsilon}y_0 - y_{0_\epsilon}x_0\right)\left(\hat{x}_0\alpha_\epsilon + \hat{y}_0\beta_\epsilon\right) \right) + \left(\frac{1}{Z_{\min}} - \frac{1}{Z_{\max}} \right) \left(\frac{1}{4}\left(y_{0_\epsilon}^2 + x_{0_\epsilon}^2 \right) e^2 + \left(x_{0_\epsilon}y_0 - y_{0_\epsilon}x_0 \right)^2 \right) \right) \tag{19}$$

a Assume that the translation has been estimated with a certain error $\mathbf{t}_\epsilon = (x_{0_\epsilon}, y_{0_\epsilon}, 0)$. Then the relationship among the errors in 3D motion at the minima of (19) is obtained from the first-order conditions $\frac{\partial E}{\partial \alpha_\epsilon} = \frac{\partial E}{\partial \beta_\epsilon} = \frac{\partial E}{\partial \gamma_\epsilon} = 0$, which yield

$$\begin{aligned} \alpha_\epsilon &= \frac{y_{0_\epsilon}(\ln(Z_{\max}) - \ln(Z_{\min}))}{Z_{\max} - Z_{\min}} \\ \beta_\epsilon &= \frac{-x_{0_\epsilon}(\ln(Z_{\max}) - \ln(Z_{\min}))}{Z_{\max} - Z_{\min}} \\ \gamma_\epsilon &= 0 \end{aligned} \tag{20}$$

It follows that $\alpha_\epsilon/\beta_\epsilon = -x_{0_\epsilon}/y_{0_\epsilon}, \gamma_\epsilon = 0$. The first of these constraints is called the orthogonality constraint ($(\alpha_\epsilon, \beta_\epsilon) \perp (x_{0_\epsilon}, y_{0_\epsilon})$).

b Assuming that rotation has been estimated with an error $(\alpha_\epsilon, \beta_\epsilon, \gamma_\epsilon)$, the relationship among the errors is obtained from $\frac{\partial E}{\partial x_{0_\epsilon}} = \frac{\partial E}{\partial y_{0_\epsilon}} = 0$. In this case, the relationship is very elaborate and the translational error depends on all the other parameters—that is, the rotational error, the actual translation, the image size and the depth interval.

c In the general case, we need to study the subspaces in which E changes least at its absolute minimum; that is, we are interested in the direction of the smallest second derivative at 0, the point where the motion errors are zero. To find this direction, we compute the Hessian at 0, that is the matrix of the second derivatives of E with respect to the five motion error parameters, and compute the eigenvector corresponding to the smallest eigenvalue. The scaled components of this vector amount to

$$\begin{aligned} & x_{0_\epsilon} = x_0 \quad y_{0_\epsilon} = y_0 \quad \beta_\epsilon = -\alpha_\epsilon \frac{x_0}{y_0} \quad \gamma_\epsilon = 0 \\ & \alpha_\epsilon = 2y_0 Z_{\min} Z_{\max} \left(\ln(Z_{\max}) - \ln(Z_{\min}) \right) \Big/ \\ & \quad \Big((Z_{\max} - Z_{\min}) (Z_{\max} Z_{\min} - 1) + \Big((Z_{\max} - Z_{\min})^2 (Z_{\max} Z_{\min} - 1)^2 \\ & \quad + 4 Z_{\max}^2 Z_{\min}^2 (\ln(Z_{\max}) - \ln(Z_{\min}))^2 \Big)^{1/2} \Big) \end{aligned}$$

As can be seen, for points defined by this direction, the translational and rotational errors are characterized by the "orthogonality constraint" $\alpha_\epsilon/\beta_\epsilon = -x_{0_\epsilon}/y_{0_\epsilon}$ and by the constraint $x_0/y_0 = \hat{x}_0/\hat{y}_0$, which is called the "line constraint." It basically means that (x_0, y_0)—the direction of the real translation, and $(\hat{x}_0, \hat{y}_0)$—the direction of the estimated translation, lie on a line passing from the origin.

The practical significance of this result is that, in general, it is not possible to find the 3D motion or 3D transformation using two views of a scene. No matter what procedure is followed, the best one can hope for is to find a set of solutions. The above mentioned results, translated into plain language, mean that when one sets up an optimization function to find the 3D motion, no matter what technique one is using, the function is such that it has valleys at the locations of its minima. It's very hard to show valleys of the five-dimensional function (three rotational and two translational parameters), so we resort to showing the valleys for the translation only (two parameters). Let's say that E is the function one chooses to optimize in order to find the 3D motion (or rigid transformation), that is, the desired translation and rotation constituting the global minimum of E.

The following procedure shows the valleys for the translation. For each possible translation, estimate the corresponding rotation from the data. One can then plot E as a function of the translation. Fig. 23 shows such a valley for two frames of a video sequence. Video 3 [video03.mpg] shows the valley for the translation, with the function E painted on the sphere. (During the illustration, points corresponding to negative depth are removed, thus reducing the ambiguity.) Fig. 24 shows just one view of the sphere. Every point of the sphere represents the direction of a possible translation. Red indicates the lowest points of the optimizing function. One can clearly see a valley. It is worth noting that the actual solution lies inside the valley but it is not necessarily the lowest point in the valley, if such a point exists. The valley may turn out to be elongated and thin, or quite large, depending on the uncertainty in the data. One can attempt a reconstruction of the scene from two frames in a video only if the corresponding valley has a small extent. In Video 4 [video04.mpg] we show the valleys for all video frames for the translation for the underlying camera motion that captured the video in Video 5 [video05.mpg]. Fig. 25a shows just the valley for two frames of the video, one frame of which is shown in Fig. 25b. Deep red signifies the lowest part of the valley. Clearly there are parts of the video where the valley is highly restricted, as there are parts where the valley has a very large extent. Failure to accurately localize the solution for the 3D motion will create problems in shape reconstruction. See, for example, the object in Video 6 (video06.mpg). Video 7 (video07.mpg) shows the depth recovery as one moves along the corresponding translation valley obtained from two consecutive frames of the video (as the slider moves from left to right, the recovered translation moves along all points in the valley). Clearly, even in the case of this smooth object, there is quite a lot of variability in the recovered shape for different translations inside this small valley.

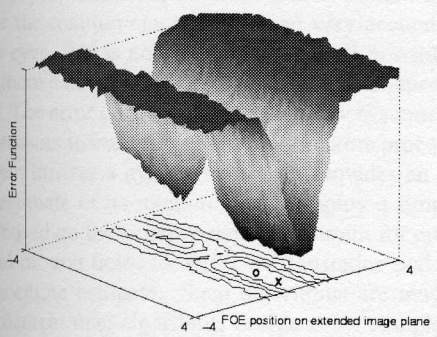


Figure 23. Minimizing E (translation only).

3.2.1 Spherical eyes

Why is it that biological systems that need to fly and thus require good estimates of 3D motion (insects, birds) have panoramic vision implemented either as a compound eye or by placing camera-type eyes on opposite sides of the head? This is a fascinating question that has remained open since the time of the pioneer investigator, Sigmund Exner, at the beginning of this century. The obvious answer is, of course, that flying systems should perceive the whole space around them—thus panoramic vision emerged. There is, however, a deeper mathematical reason and it has to do with the ability of a system to estimate 3D motion when it analyzes panoramic images, as shown in this section. Put simply, a spherical eye (360 degree field of view) is superior to a planar eye (restricted field) with regard to 3D motion estimation. Recall that, given a sequence of images, 3D motion is estimated by minimizing function E that represents deviation from the epipolar constraint. It was shown that in the case of images captured by a planar eye (e.g., a common video camera), this function has a special topography which is such that the errors in the motion are mingled, causing confusion between rotation and translation and thus producing a wrong result. If, however, the field of view goes to 360 degrees, the topography of the surface drastically changes with the minimum clearly standing out. It is no wonder then that flying organisms possess panoramic vision!

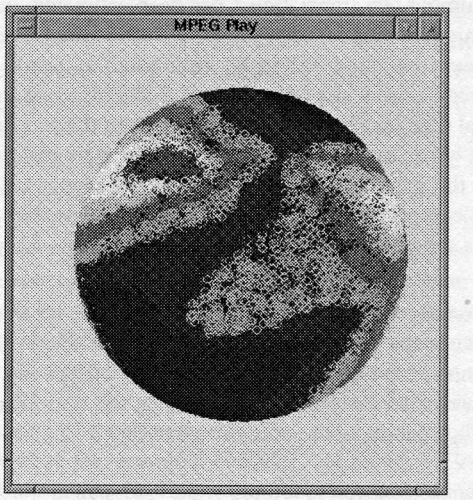


Figure 24.

The analysis that leads to this result is almost identical to the analysis performed for planar eyes. Panoramic vision is modeled by projecting onto a sphere, with the sphere's center as the center of projection (Fig. 1b). In this case, the image $\mathbf{r}$ of any point $\mathbf{R}$ is $\mathbf{r} = \frac{\mathbf{R}f}{|\mathbf{R}|}$, with R being the norm

of $\mathbf{R}$ (the range), and the image motion is

$$\dot{\mathbf{r}} = \frac{1}{|\mathbf{R}|f}\left((\mathbf{t}\cdot\mathbf{r})\,\mathbf{r} - \mathbf{t}\right) - \boldsymbol{\omega}\times\mathbf{r} = \frac{1}{R}\,\mathbf{u}_{\mathrm{tr}}(\mathbf{t}) + \mathbf{u}_{\mathrm{rot}}(\boldsymbol{\omega}). \tag{21}$$

The function M_{ep} representing deviation from the epipolar constraint on the sphere has the exact same form as in the plane for our nomenclature. We integrate over the range R within an interval bounded by $R_{\min}$ and $R_{\max}$ and obtain

$$E_{ep} = \int_{R_{\min}}^{R_{\max}} \iint_{\text{sphere}} \left\{\left(\frac{\mathbf{r}\times(\mathbf{r}\times\mathbf{t})}{R} - (\boldsymbol{\omega}_\epsilon\times\mathbf{r})\right)\cdot(\hat{\mathbf{t}}\times\mathbf{r})\right\}^2 dA\,dR$$

where A refers to a surface element. Due to the sphere's symmetry, for each point $\mathbf{r}$ on the sphere, there exists a point with coordinates $-\mathbf{r}$. Since $\mathbf{u}_{\mathrm{tr}}(\mathbf{r}) = \mathbf{u}_{\mathrm{tr}}(-\mathbf{r})$ and $\mathbf{u}_{\mathrm{rot}}(\mathbf{r}) = -\mathbf{u}_{\mathrm{rot}}(-\mathbf{r})$, when the integrand is expanded the product terms integrated over the sphere vanish. Thus

$$E_{ep} = \int_{R_{\min}}^{R_{\max}} \iint_{\text{sphere}} \left\{\frac{\left((\mathbf{t}\times\hat{\mathbf{t}})\cdot\mathbf{r}\right)^2}{R^2} + \left((\boldsymbol{\omega}_\epsilon\times\mathbf{r})\cdot(\hat{\mathbf{t}}\times\mathbf{r})\right)^2\right\} dA\,dR$$

a Assuming that translation $\hat{\mathbf{t}}$ has been estimated, the $\boldsymbol{\omega}_\epsilon$ that minimizes E_{ep} is $\boldsymbol{\omega}_\epsilon = 0$, since the resulting function is non-negative quadratic in $\boldsymbol{\omega}_\epsilon$ (minimum at zero). The difference between sphere and plane is already clear. In the spherical case, as shown here, if an error in the translation is made we do not need to compensate for it by making an error in the rotation ($\boldsymbol{\omega}_\epsilon = 0$), while in the planar case we need to compensate to ensure that the orthogonality constraint is satisfied!

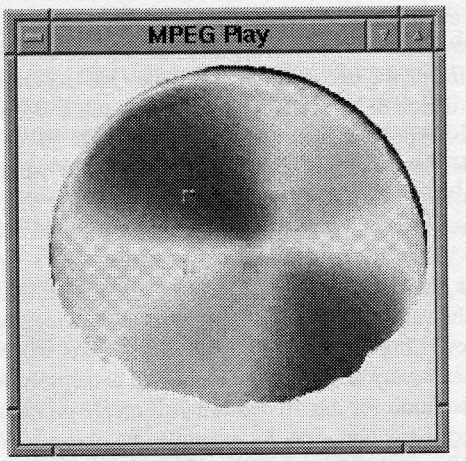
b Assuming that rotation has been estimated with an error $\boldsymbol{\omega}_\epsilon$, what is the translation $\hat{\mathbf{t}}$ that minimizes E_{ep}? Since R is assumed to be uniformly distributed, integrating over R does not alter the form of the error in the optimization. Thus, E_{ep} consists of the sum of two terms:

$$K = K_1 \iint_{\text{sphere}} \left((\mathbf{t}\times\hat{\mathbf{t}})\cdot\mathbf{r}\right)^2 dA$$

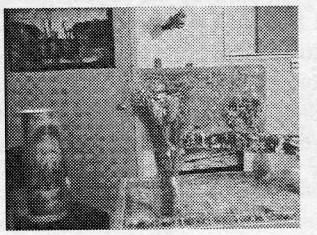
and

$$L = L_1 \iint_{\text{sphere}} \left((\boldsymbol{\omega}_\epsilon\times\mathbf{r})\cdot(\hat{\mathbf{t}}\times\mathbf{r})\right)^2 dA,$$

where K_1, L_1 are multiplicative factors depending only on $R_{\min}$ and $R_{\max}$. For angles between $\mathbf{t}, \hat{\mathbf{t}}$ and $\hat{\mathbf{t}}, \boldsymbol{\omega}_\epsilon$ in the



(a) The image size is denoted by four corners



(b)

Figure 25.

range of 0 to $\pi/2$, K and L are monotonic functions. K attains its minimum when $\mathbf{t} = \hat{\mathbf{t}}$ and L when $\hat{\mathbf{t}} \perp \boldsymbol{\omega}_\epsilon$. Fix the distance between $\mathbf{t}$ and $\hat{\mathbf{t}}$ leading to a certain value K, and change the position of $\hat{\mathbf{t}}$. L takes its minimum when $(\mathbf{t} \times \hat{\mathbf{t}}) \cdot \boldsymbol{\omega}_\epsilon = 0$, as follows from the cosine theorem. Thus E_{ep} achieves its minimum when $\hat{\mathbf{t}}$ lies on the great circle passing through $\mathbf{t}$ and $\boldsymbol{\omega}_\epsilon$, with the exact position depending on $|\boldsymbol{\omega}_\epsilon|$ and the scene in view.

c For the general case where no information about rotation or translation is available, we study the subspaces where E_{ep} changes the least at its absolute minimum, i.e., we are again interested in the direction of the smallest second derivative at 0. For points defined by this direction we calculate, using Maple, $\mathbf{t} = \hat{\mathbf{t}}$ and $\boldsymbol{\omega}_\epsilon \perp \mathbf{t}$.

There are only two fundamentally different constraints that relate image measurements to 3D motion parameters. The first one is the epipolar constraint, which is applied with estimates of correspondence or flow, and any techniques which utilize these measurements minimize some measure of deviation from the epipolar constraint. The second one, which is used with normal flow measurements, is the positive depth constraint. Algorithms that utilize this constraint search for the motion parameters that produce a minimum number of negative depth values, as explained in Section 3.1. The analysis of this constraint is harder and we will only provide the results from the proofs that appeared recently [21, 22].

The table below summarizes the results for both constraints.

The preceding results demonstrate the advantages of spherical eyes for the process of 3D motion estimation. Table 1 lists the eight out of ten cases which lead to clearly defined error configurations. It shows that 3D motion can be estimated more accurately with spherical eyes. Depending on the estimation procedure used—and systems might use different procedures for different tasks—either the translation or the rotation can be estimated very accurately. For planar eyes, this is not the case, as for all possible procedures there exists confusion between the translation and rotation. The error configurations also allow systems with inertial sensors to use more efficient estimation procedures. If a system utilizes a gyrosensor which provides an approximate estimate of its rotation, it can employ a simple algorithm based on the negative depth constraint for only translational motion fields to derive its translation and obtain a very accurate estimate. Such algorithms are much easier to implement than algorithms designed for completely unknown rigid motions, as they amount to searches in 2D as opposed to 5D spaces [12]. Similarly, there exist computational advantages for systems with translational inertial sensors in estimating the remaining unknown rotation.

3.3 Camera technology: Building new eyes

Since it turns out that spherical eyes such as the ones of insects, or, in general, panoramic vision provides much better capability for 3D motion estimation, and since the problem of building accurate space and action descriptions depends on accurate 3D motion computation, it makes sense to reconsider what the best eye for model building should be. There are a few ways to create panoramic vision cameras, and the recent literature is rich in alternative approaches, but there is a way to take advantage of both the panoramic vision of flying systems and the high resolution vision of primates. An eye like the one in Fig. 26, assembled from a few video cameras arranged on the surface of a sphere and capable of simultaneous recording,[1] can easily estimate 3D motion since, while it is moving, it is sampling a spherical motion field!

An eye like the one in Fig. 26 not only has panoramic properties, eliminating the rotation/translation confusion, but it has the unexpected benefit of making it easy to estimate image motion with high accuracy. Any two cameras with overlapping fields of view also provide high-resolution stereo vision, and this collection of stereo systems makes it possible to locate a large number of depth discontinuities. It is well known that, given scene discontinuities, image motion can be estimated very accurately. As a consequence, the eye in Fig. 26 is very well suited to developing accurate models of the world.

We built our own imaging system, called the Argus eye.[2] (We are currently working on a new version consisting of twelve cameras). Fig. 27a shows a schematic version of the Argus eye, consisting of six cameras arranged to point outwards. Fig. 27b shows an actual view of the system, and Video 8 [video08.mpg] shows graphically what such a system sees (it samples parts of the visual sphere). When the Argus eye is moving with an unrestricted 3D motion collecting synchronized video from all six cameras, it becomes easy to compute its 3D motion using data from all six videos, if the cameras are calibrated in an extrinsic sense. If we analyze each video separately we find, at each instant, a valley for the translation of each of the cameras, as shown in Fig. 28. As the rotation of each camera is the same as the rotation of the system, there are easy ways to compute the rotation. For example, consider one camera. For each translation inside the valley, there is a corresponding rotation. For all translations the corresponding rotational values lie on a surface in 3D space. Considering all cameras, these surfaces intersect at one point in space which provides the rotation. (Video 9 [video09.mpg] shows the growth of these surfaces and their intersection.) If we then derotate each one

[1] Like a compound eye with video cameras replacing ommatidia

[2] Named after a mythological figure, the guardian of Hera (the goddess of Olympus).

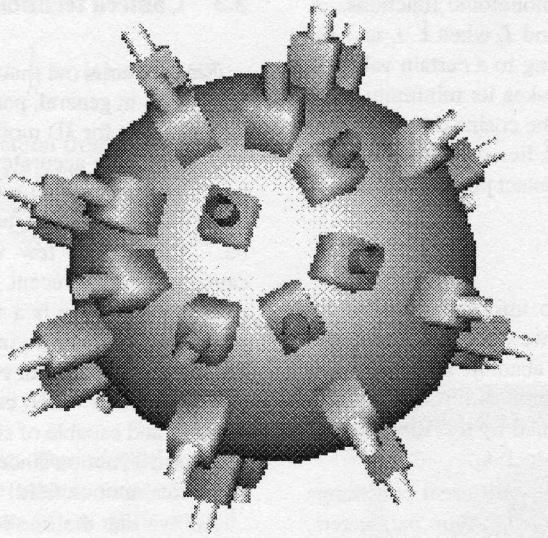
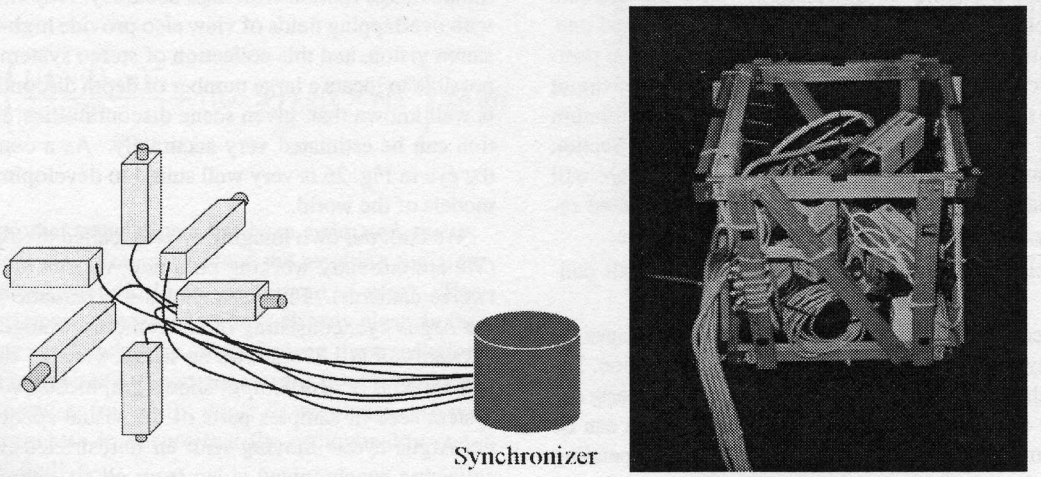


Figure 26. A compound-like eye composed of conventional video cameras.



(a) (b)

Figure 27. The Argus eye. (a) Schematic. (b) Implementation.

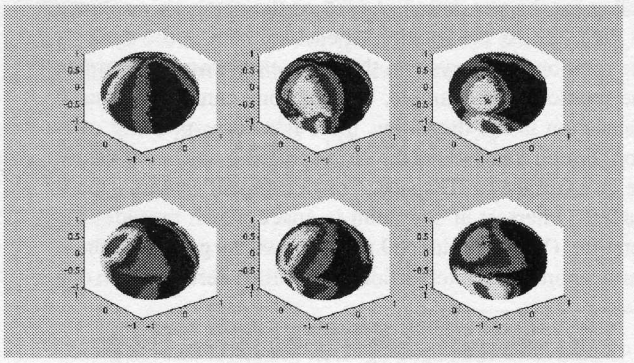


Figure 28. Initial valleys from each camera (translation).

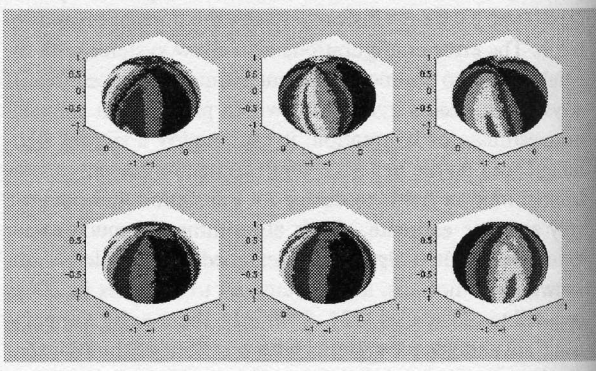


Figure 29. Translation valleys after derotation.

Table 1: Summary of results

	Spherical Eye	Camera-type Eye
Epipolar minimization, given optic flow	(a) Given a translational error $\mathbf{t}_\epsilon$, the rotational error $\boldsymbol{\omega}_\epsilon = 0$ (b) Without any prior information, $\mathbf{t}_\epsilon = 0$ and $\boldsymbol{\omega}_\epsilon \perp \mathbf{t}$	(a) For a fixed translational error $(x_{0_\epsilon}, y_{0_\epsilon})$, the rotational error $(\alpha_\epsilon, \beta_\epsilon, \gamma_\epsilon)$ is of the form $\gamma_\epsilon = 0$, $\alpha_\epsilon/\beta_\epsilon = -x_{0_\epsilon}/y_{0_\epsilon}$ (b) Without any a priori information about the motion, the errors satisfy $\gamma_\epsilon = 0$, $\alpha_\epsilon/\beta_\epsilon = -x_{0_\epsilon}/y_{0_\epsilon}$, $x_0/y_0 = x_{0_\epsilon}/y_{0_\epsilon}$
Minimization of negative depth volume, given normal flow	(a) Given a rotational error $\boldsymbol{\omega}_\epsilon$, the translational error $\mathbf{t}_\epsilon = 0$ (b) Without any prior information, $\mathbf{t}_\epsilon = 0$ and $\boldsymbol{\omega}_\epsilon \perp \mathbf{t}$	(a) Given a rotational error, the translational error is of the form $-x_{0_\epsilon}/y_{0_\epsilon} = \alpha_\epsilon/\beta_\epsilon$ (b) Without any error information, the errors satisfy $\gamma_\epsilon = 0$, $\alpha_\epsilon/\beta_\epsilon = -x_{0_\epsilon}/y_{0_\epsilon}$, $x_0/y_0 = x_{0_\epsilon}/y_{0_\epsilon}$

of the videos the valleys become much thinner (Fig. 29), and bringing them all to the same coordinate system provides a unique translation for the whole system, as shown in Fig. 30. Using these values, one can perform impressive reconstructions of the scene, with many applications to graphics and augmented reality [23].

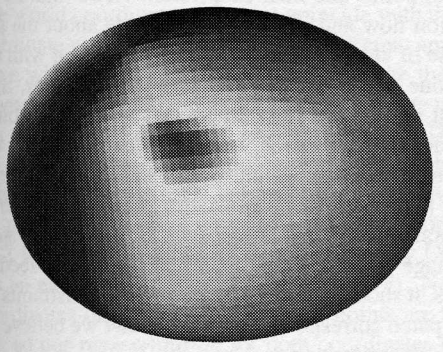


Figure 30. Valley for the whole system (translation).

4 Shape: State of the art

If we put aside panoramic vision and we concentrate on conventional cameras, given a video we should be able to recover some information about the camera's 3D motion and the shape of the scene. According to the preceding discussion, the best we can hope for is an answer with an associated uncertainty. We can, for example, reconstruct the camera path, that is, place the camera coordinate system in space for each video frame. See, for example, Video 10 [video10.mpg] denoting a sequence and Video 11 [video11.mpg] showing the placement of the camera coordinate system in space. The employed algorithms give promising results in the recovery of shape. For example, from a small part of the video in Video 12a [video12a.mpg] (a Pooh game), we recovered the shape shown in Video 12b [video12b.mpg], using the algorithms in [24]. We know, however, that from every two consecutive frames we can only hope for a valley where the translation lies; there is, of course, a corresponding uncertainty in the rotational estimate. As a result, we do not know exactly how the cameras are placed and, consequently, we can have a whole set of scene models consistent with the data and the uncertainty in the 3D motion. Simply put, there will be uncertainty in the estimation of the scene structure. It is worth asking the question: How is a model for the scene distorted because of errors in the viewing geometry? It turns out [25] that the transformation relating the computed depth to the actual depth is a Cremona transformation. This is easy to see. If

$\hat{\mathbf{t}}, \hat{\omega}$ are the estimated 3D motion parameters and $\mathbf{t}_\epsilon = \mathbf{t} - \hat{\mathbf{t}}$, $\boldsymbol{\omega}_\epsilon = \boldsymbol{\omega} - \hat{\omega}$ the errors, then (3) provides the value u_n of the flow on direction $\mathbf{n}$:

$$\hat{Z} = \frac{\mathbf{u}_{\rm tr}(\hat{\mathbf{t}}) \cdot \mathbf{n}}{\mathbf{u}_n - \mathbf{u}_{\rm rot}(\hat{\boldsymbol{\omega}}) \cdot \mathbf{n}}$$

or

$$\hat{Z} = Z \cdot D \quad \text{with} \quad D = \frac{\mathbf{u}_{\rm tr}(\hat{\mathbf{t}}) \cdot \mathbf{n}}{\mathbf{u}_{\rm tr}(\mathbf{t}) \cdot n + Z\mathbf{u}_{\rm rot}(\hat{\boldsymbol{\omega}}_\epsilon) \cdot \mathbf{n}} \quad (22)$$

Thus, the estimated depth $\hat{Z}$ is related to the real depth Z by a multiplicative factor D, the distortion, which is a Cremona transformation [25].

Equation (22) can be written as

$$[(D-1)\mathbf{r} + (\hat{\mathbf{t}} - D\mathbf{t}) + DZ\mathbf{u}_{\rm rot_\epsilon}] \cdot \mathbf{n} = 0$$

which for a fixed $\mathbf{n}, D, \mathbf{t}, \hat{\mathbf{t}}, \boldsymbol{\omega}_\epsilon$ represents a surface in $(\mathbf{r}, Z)$ space, which has the obvious property that points on it are distorted by the same factor D. The distortion space has very interesting properties with consequences in algorithms estimating shape. Its structure, in conjunction with psychophysical measurements, predicts various illusions of misperception of shape and structure [25].

By now it is clear that from a video sequence we can hope to recover valleys containing the translation and rotation (instantaneous motion) and a set of possible scene models M_i using images i and $i+1$ in the sequence. To produce the ultimate 3D percept, we must put all the models extracted from different viewpoints into the same coordinate system. Somehow, the video frames need to be linked, but as they are linked they should provide an accurate model of the scene. It should be emphasized, however, that up to this point no correspondence process has been yet attempted.

If we assume for a moment that correspondence is known for a small number of points in any two frames, then this becomes very valuable information. Consider having two views, 1 and 2, of a scene and for each view we also have sets of possible models M_1 and M_2. Since both views look at the same scene, the correct answer for model $m_1 \in M_1$ and $m_2 \in M_2$ is such that m_1 and m_2 are related by a similarity transformation (Euclidean plus scale), which is trivial to check since a number of point correspondences is available. Thus, by checking all models in M_1 and M_2 to find out if they are related to each other via a similarity transformation, we can reduce the ambiguity. As a matter of fact by doing so we can shrink the size of the valleys to a few pixels.[3]

So, it would really pay off if we could perform matching of a number of points. But we should recall that asking a question about correspondence at this point in the development is very different than asking correspondence questions when one is given two images and no more information. At this stage we already have some knowledge about camera placement and the scene model, we already know about the viewing geometry, with some uncertainty no doubt.

The analysis up to now demonstrates that the problem we consider is a chicken-egg problem. To find the 2D transformation (flow or correspondence) we need some information about depth boundaries; that way we will know not to smooth across boundaries. But to obtain depth information we need to know the 3D transformation which in turn depends on the 2D transformation.

It thus appears, on the basis of computational arguments, that the problem of structure from motion needs a number of feedback loops, a new way in which the three different modules interact. We believe this problem opens very important and new research avenues and can lead to solutions of the correspondence problem.

5 Feedback loops: Matching image patches and blending statistics with geometry

When neuroscience revealed that feedback loops appear to be involved at several levels of cortical processing, vision theorists became comfortable with the idea that feedback could be some form of iteration implementing an optimization process. This view, however, is not very useful because it does not offer an effective way to uncover the intricacies of the feedback process. Let us investigate at a high level what properties a feedback process should have:

a It is clear that by now we understand a lot about the relationship of points and lines in different views (that is, what information flow and normal flow provide about the scene in the case of video). The uncertainty associated with these features due to the statistical bias of any operator, makes it difficult to obtain a fully automated and robust solution for a model of the scene in view. Whatever form the feedback scheme takes, it should not consider the correspondence of points or lines. But what is there in the image besides points and lines? An answer is, naturally, patches, whole image patches. That is, whatever form a feedback loop takes, it should consider the geometric constraints arising from patch correspondence, a topic that we believe constitutes a very important problem.

b A second issue regarding feedback is the quantities that are involved in the loop. We will need to further analyze these quantities and, if needed, split them into more primitive quantities. We have:

[3] Having correspondences and associated depth makes things very easy because the motion problem becomes linear.

1. The 2D transformation. This is the image motion or correspondence map about which we know quite a lot; it has with it a number of more primitive maps, the normal flow and the sign of the flow along a number of orientations at each point.

2. The 3D motion or viewing geometry. It consists of a rotation and a translation, and

3. The model of the scene, i.e., its 3D structure. But what is really the scene's 3D model? The geometry of solid shape [26] is highly developed and an analysis is possible along many paths. But we need an analysis relevant to our problem. The 3D model has a local and a global characteristic. If we concentrate on a little patch, a plane, then shape basically amounts to the orientation of the plane, its normal with regard to the viewer's coordinate system. So, let's agree that knowledge of *shape* amounts to knowledge of the orientation of the local planar patches. The global aspect of the 3D model, that we call *structure*, has to do with knowing how to put all the patches together to make the scene, that is, knowing their depth up to some scale.

So, shape and structure are two likely candidates for decomposing the 3D model. Surprisingly, there exists a very simple observation that makes the decomposition ideal. Shape may be estimated from rotation only and structure requires translation. Given two views of a scene separated by a rigid transformation consisting of a rotation and a translation, the shape may be estimated on the basis of rotation only. Then, knowledge of the translation provides the structure. This simple fact can be shown in a variety of ways; we choose the following as it gives us the opportunity to introduce the nomenclature needed for introducing the new geometric constraints for patch correspondence.

In contrast to the case of differential motion where we took the world coordinate system to coincide with the camera coordinate system, here we follow a different approach. Since we will consider multiple views, we take the world coordinate system at a fiducial position and denote the center of a camera by a vector $\mathbf{T}$ (the translation). Then the camera is oriented by some rotation, and we will also consider the camera's intrinsic calibration parameters. Denote a point in 3D ($\mathbb{R}^3$) with its Euclidean coordinates $\mathbf{P} = [XYZ]^T$, and denote its image as the ray $[XYZ]^T$ seen as an element of the projective plane ($\mathbb{P}^2$). Note that the coordinates of the world and image points are both 3-vectors, so our *representation* for both coordinates is identical. We can think of a camera as as device for considering the coordinates in $\mathbb{R}^3$ to be coordinates in $\mathbb{P}^2$. However, our world points exist in one fiducial coordinate system, while the image points exist in the particular camera coordinate systems. Therefore our camera is defined in relation to this fiducial coordinate system as follows. A **camera** C is a map $C : \mathbb{R}^3 \to \mathbb{P}^2$ from world points to image points. Given a fiducial coordinate system, we may represent this map with a pair $(B, \mathbf{T})$, where $B : \mathbb{R}^3 \to \mathbb{R}^3$ is a linear function (represented by a 3×3 matrix), and $\mathbf{T}$ is a 3-vector representing the **camera center**. The action of the map on a world point coordinate $\mathbf{P}$ is:

$$C(\mathbf{P}) = B(\mathbf{P} - \mathbf{T})$$

where $C(\mathbf{P})$ is considered as a member of $\mathbb{P}^2$.

Here we have defined a camera as taking a world point and mapping it to an image point through a general linear transformation on the world coordinates. Note that we have defined the transformation as first a translation and then a matrix multiplication on the world point. This allows us to easily undo the matrix multiplication on the image point by applying B^{-1}. Each camera will then be only a translation away from the fiducial coordinate system, which allows easier derivation of our constraints. Additionally, note that the matrix B is applied to the 3-vector resulting from the translations of the point $\mathbf{P}$. B may be considered to be a transformation on the world points $\mathbb{R}^3$ or of the image points $\mathbb{P}^2$. In fact, B is usually split apart using a QR decomposition, with the orthogonal matrix representing a rotation of the camera (a transformation on $\mathbb{R}^3$) and the residual matrix representing a linear transformation of the image (a transformation on $\mathbb{P}^2$), depending on the camera's calibration parameters. Since the coordinates are the same, we ignore such distinctions for the present. The rotation is included in B and the translation is $\mathbf{T}$.

Let us introduce the Plücker coordinates for lines. A **world line** L is the set of all the points $P \in \mathbb{R}^3$ such that $\mathbf{P} = (1-\lambda)\mathbf{Q}_1 + \lambda\mathbf{Q}_2$ for two points $\mathbf{Q}_i$, and some scalar λ. If we consider $\lambda = 1$ and $\lambda = 0$, we see that the line L contains both Q_1 and Q_2. The Plücker coordinates of this line are $\mathbf{L} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \end{bmatrix}$, where:

$$\mathbf{L}_d = \mathbf{Q}_2 - \mathbf{Q}_1 \qquad \text{direction of } L \qquad (23)$$

$$\mathbf{L}_m = \mathbf{L}_d \times \mathbf{P} \qquad \text{moment of } L \qquad (24)$$

Note that regardless of the choice of λ to define $\mathbf{P}$, the definition of $\mathbf{L}_m$ is the same. Also, the coordinates of $\mathbf{L}$ are homogeneous, and $\mathbf{L}_m^T \mathbf{L}_d = 0$.

An image line is a line in the projective plane, and may be given coordinates $\boldsymbol{\ell} = [l_1 l_2 l_3]^T$. The projection of a line onto a camera is as simple as the projection of a point onto a camera. If we have a line L and a camera $(B, \mathbf{T})$, then the image line associated with L is

$$\hat{\boldsymbol{\ell}} = B^{-\mathrm{T}}(\mathbf{L}_m - \mathbf{T} \times \mathbf{L}_d) \qquad (25)$$

It should be noted that when the image points $\mathbf{P}$ are transformed by a map B to $\hat{\mathbf{p}}$ with $\hat{\mathbf{p}} = B\mathbf{p}$, then the image lines l are transformed to $\hat{\boldsymbol{\ell}} = B^{-T}\boldsymbol{\ell}$. If we have a world

coordinate system, and a camera in that coordinate system with parameters $(B, \mathbf{T})$, then we consider the $\hat{\mathbf{p}}$ and $\hat{\boldsymbol{\ell}}$ to be the actual point and line measured in the image. For most of the derivations, we use the normalized image lines/points

$$\mathbf{p} = B^{-1}\hat{\mathbf{p}} \qquad \boldsymbol{\ell} = (B^{-\mathrm{T}})^{-1}\hat{\boldsymbol{\ell}} = B^{\mathrm{T}}\hat{\boldsymbol{\ell}}$$

Whenever we assume that we already have the B, we will use the normalized image lines/points for the calculations. We multiply the appropriate B or $B^{-\mathrm{T}}$ back later. We show that by considering the prismatic line constraint, a new constraint we introduce later, we are able, when using three cameras, to factor out the B matrices, thus giving this notational convenience a theoretical basis.

Let us now consider the reconstruction of a world line. If we have a line L in space which projects to two image lines $\hat{\boldsymbol{\ell}}_1$ and $\hat{\boldsymbol{\ell}}_2$ in cameras $(B_1, \mathbf{T}_1)$, and $(B_2, \mathbf{T}_2)$, then we can calculate the following coordinates for $\mathbf{L}$ if $|\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2| \neq \mathbf{0}$ as:

$$\mathbf{L} = \begin{bmatrix} \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 \\ \boldsymbol{\ell}_2 \mathbf{T}_1^{\mathrm{T}} \boldsymbol{\ell}_1 - \boldsymbol{\ell}_1 \mathbf{T}_2^{\mathrm{T}} \boldsymbol{\ell}_2 \end{bmatrix}$$

Clearly the directional component of the line depends only on the rotation (recall that $\boldsymbol{\ell} = B^T\hat{\boldsymbol{\ell}}$). That is, from at least two views of two lines on a planar patch, we can recover the plane's orientation from the recovered directional components of the two lines, without any knowledge of the translation between the views.

The first part of our feedback loop then becomes clear. Whatever we do to make 3D measurements, we have to start from image measurements. Whatever measurements we make in the image, we are constrained by the distortion that has happened because of the projection. If we could recover the rotation, we could in principle perform the appropriate transformation in the image so that we minimize distortion. Another way to think about this, is that knowing the rotation allows us to perform signal processing on the object's surface, so that we can better estimate the rotation and the remaining translation. So, the feedback loop acquires two distinct steps for processing multiple views. In the first step, signal processing in the image provides answers for at least rotation using the prismatic line constraint, which allows the beginning of the second step that amounts to signal processing on the world plane. This classification makes intuitive sense also. If we hope to do better with a feedback loop, we must have a place in the loop where some new information comes in. If we stick to the original measurements, there wouldn't be much hope for improvement. So, somewhere in the loop we must make measurements again. We think that the appropriate place for it is after rotation between views is estimated because then shape (orientation of planes) is easily obtained. We can then map image texture on the scene planes and, redo the problem from the beginning. But after we do this, we must look for new constraints arising from patch correspondence. Points and lines have no more information to provide.

5.1 Corresponding textured patches

We assume that we can correspond patches in different views. This is reasonable given the state of the art in this field; even if there are inaccuracies in the viewing geometry, very often it becomes easy to correspond patches. But what is the primitive nature of a textured planar patch in a form that does not depend on localizing points or lines? Equivalently, what is good representation for a patch where measurements depend on the output of filters which have a finite area of support? An example of a primitive such candidate is a periodic function in one dimension extended to span a plane. Such a function can be represented as repeated parallel lines. By putting many of these objects together with appropriate constraints, we may represent any arbitrary textured plane, but in a fashion which allows for geometric reasoning about correspondence and reconstruction. Thus, we consider as a fundamental object repeated lines in space. We can represent the plane and the texture embedded in the plane together in a geometrical way as follows:

Consider one line from the set of equally spaced lines in the plane, and call it $\mathbf{L}_0$. We may represent this line using Plücker coordinates as $\mathbf{L}_0 = \left[\begin{smallmatrix} \mathbf{L}_d \\ \mathbf{L}_m \end{smallmatrix}\right]$. We take $\mathbf{Q}_0$ to be a point on $\mathbf{L}_0$, and the point $\mathbf{Q}_n = \mathbf{Q}_0 + n\mathbf{d}$ to be on the n^{th} line in the texture for some direction $\mathbf{d}$. Then it is easy to obtain that

$$\mathbf{L}_d \times \mathbf{Q}_0 = \mathbf{L}_{m,0}$$

so that to get $\mathbf{L}_{m,n}$

$$\begin{aligned} \mathbf{L}_{m,n} &= \mathbf{L}_d \times \mathbf{Q}_n \\ &= \mathbf{L}_d \times \mathbf{Q}_0 + n\mathbf{L}_d \times \mathbf{d} \\ &= \mathbf{L}_m + n\mathbf{L}_\lambda \end{aligned}$$

where $\mathbf{L}_\lambda = \mathbf{L}_d \times \mathbf{d}$. Note that since both $\mathbf{L}_d$ and $\mathbf{d}$ are vectors which lie inside the plane, we must have that $\mathbf{L}_\lambda$ is normal to the textured plane. This leads us to the following definition. A **singly textured plane** H is a set of lines, equally spaced, embedded in a world plane. We give the textured plane coordinates

$$\mathbf{H} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \\ \mathbf{L}_\lambda \end{bmatrix}$$

with $\mathbf{L}_d^{\mathrm{T}}\mathbf{L}_m = 0$ and $\mathbf{L}_d^{\mathrm{T}}\mathbf{L}_\lambda = 0$. The coordinates of each line in the plane, indexed by n are:

$$\mathbf{L}_n = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m + n\mathbf{L}_\lambda \end{bmatrix}$$

It can be shown that, if we have two textured planes $\mathbf{H}_1$ and $\mathbf{H}_2$, then they lie on the same world plane if and only

if:

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0$$

and

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{\lambda,2} = 0$$

and

$$\mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{\lambda,1} = 0$$

We can then easily relate image lines to world lines. The set of image lines $\{\hat{\boldsymbol{\ell}}_n\}$ of a singly textured plane H in a camera with parameters $(B, \mathbf{T})$ is

$$\{\hat{\boldsymbol{\ell}}_n : \hat{\boldsymbol{\ell}}_n = B^{-\mathrm{T}}(\mathbf{L}_m + n\mathbf{L}_\lambda - \mathbf{T}\times\mathbf{L}_d)\} \qquad (26)$$

where $n \in \mathbb{Z}$. Note that the image lines have homogeneous coordinates so that the image lines are not equally spaced, as would appear from first glance at the equation. To the contrary, as n goes to infinity, the image lines will approach the image line $\mathbf{L}_\lambda$, since that term will dominate. If we observe a textured plane in many views, then we can easily obtain the rotation between the views without knowing the correspondence between lines in the views. This is a direct consequence of the prismatic line constraint which is explained below.

5.2 The prismatic constraint

Assume a set of parallel lines in space and their projections on three cameras (B_i, T_i). Pick any image line $\boldsymbol{\ell}_i$ in each camera. Note that the $\boldsymbol{\ell}_i$'s do not correspond to each other. The three planes Π_i defined by a camera center and the line $\boldsymbol{\ell}_i$ obviously form a prism in space, and it is a simple property of the prism that the normals of its faces are coplanar. But these normals are (in projective coordinates) the vectors $\boldsymbol{\ell}_i$, and so $\boldsymbol{\ell}_2^T(\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_3) = 0$ or $\boldsymbol{\ell}_2^T B_2^{-1}(B_1^{-T}\hat{\boldsymbol{\ell}}_1 \times B_3^{-T}\boldsymbol{\ell}_3) = 0$. This is the prismatic line constraint which gives rise to linear algorithms for recovering the rotation given patch correspondence without knowledge of how the lines in each patch correspond to each other in the different views.

5.3 Reconstructing a textured plane: New constraints

We propose to represent all textured planes by means of a linear combination of singly textured planes through the use of the Fourier transform, with a singly textured plane representing a single exponential. Therefore we will be concerned not only with the reconstruction of a singly textured plane, but with the constraint that two singly textured planes must be coincident.

We must first describe the notation

$$\sum_{[i_1..i_n]\in \mathrm{P}+[1..n]} \qquad (27)$$

This is a summation which goes over all of the *positive* permutations of $[1..n]$, putting each permutation into the indices $[i_1..i_n]$.

One can show that if we have a textured plane $\mathbf{H}$ which is imaged by four cameras into image lines $\hat{\boldsymbol{\ell}}_i$, and we know that our cameras have parameters $(B_i, \mathbf{T}_i)$, and further, we know that the image lines have indices n_i, then we may reconstruct the textured plane as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \\ \mathbf{L}_\lambda \end{bmatrix} = \sum_{[i_1..i_4]\in \mathbf{P}^+[1..4]} \begin{bmatrix} n_{i_1}n_{i_2}|\boldsymbol{\ell}_{i_3}\times\boldsymbol{\ell}_{i_4}|(\boldsymbol{\ell}_{i_1}\times\boldsymbol{\ell}_{i_2}) \\ 2n_{i_1}n_{i_2}|\boldsymbol{\ell}_{i_1}\times\boldsymbol{\ell}_{i_2}|\boldsymbol{\ell}_{i_3}\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ 2n_{i_1}|\boldsymbol{\ell}_{i_2}\times\boldsymbol{\ell}_{i_3}|\boldsymbol{\ell}_{i_1}\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \end{bmatrix} \qquad (28)$$

Note that $|\cdot|$ is the *signed* magnitude, and since the coordinates are homogeneous, it does not matter which sign is chosen.

It is now easy to discover constraints between patches that involve the translation between views. To set up these constraints in their simplest form we employ five or more views. However, these constraints can be applied to fewer cameras by collapsing cameras on each other. We describe three constraints: The constraint on a single texture (5 views), the constraint on a texture and a line (6 views) and the constraint on two textures (8 views).

The concept of line indices figures heavily in this framework. Essentially, to form these constraints, we assume that we know the index of the line in the textured plane. In practice, this means that if we know the translational positions, we can find the indices, and if we know the indices, we can find the translational positions. For now, we describe the constraints.

Consider that we have five cameras $(B_i, \mathbf{T}_i)$, and measure five lines $\hat{\boldsymbol{\ell}}_i$, which have indices n_i. We may form $\boldsymbol{\ell}_i$ using the $\hat{\boldsymbol{\ell}}_i$ and the B_i. Using the above result, we may reconstruct the textured plane to obtain the $\mathbf{H}$ using the lines one through four. Using this reconstruction, we can find the fifth image line as:

$$\boldsymbol{\ell}_5 = \mathbf{L}_m + n_5\mathbf{L}_\lambda - \mathbf{T}_5\times\mathbf{L}_d \qquad (29)$$

If $\mathbf{p}_5$ is a point on $\boldsymbol{\ell}_5$, we know that $\mathbf{p}_5$ is perpendicular to $\boldsymbol{\ell}_5$, so that $\mathbf{p}_5^{\mathrm{T}}\boldsymbol{\ell}_5 = 0$. We can use this with the above equation to formulate the constraint. Note that since $\mathbf{L}_d$ is perpendicular to $\boldsymbol{\ell}_5$ that $\mathbf{L}_d$ is a point on the line $\boldsymbol{\ell}_5$, but if we

set $\mathbf{p} = \mathbf{L}_d$, all of the right hand side terms disappear and we have no constraint. Therefore we know that there is only one equation in our constraint, and we use $\mathbf{p}_5 = \mathbf{L}_d \times \boldsymbol{\ell}_5$. We can derive

$$0 = (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\mathbf{L}_m + n_5\mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d) \quad (30)$$

$$= |\mathbf{L}_d\ \boldsymbol{\ell}_5\ \mathbf{L}_m| + n_5|\mathbf{L}_d\ \boldsymbol{\ell}_5\ \mathbf{L}_\lambda| - (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\mathbf{T}_5 \times \mathbf{L}_d) \quad (31)$$

we use vector algebra and the fact that $\mathbf{L}_d^{\mathrm{T}}\boldsymbol{\ell}_5 = 0$ to obtain $-\mathbf{L}_d^{\mathrm{T}}\mathbf{L}_d\mathbf{T}^{\mathrm{T}}\boldsymbol{\ell}_5$ for the last term

$$= \sum_{[i_1..i_4]\in \mathrm{P}^+(1..4)} [2n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \quad (32)$$

$$+ 2n_{i_1}n_5(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_1})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \quad (33)$$

$$+ n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})^{\mathrm{T}}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})\mathbf{T}_5^{\mathrm{T}}\boldsymbol{\ell}_5 \quad (34)$$

which we can expand to

$$= \sum_{[i_1..i_4]\in \mathrm{P}^+[1..4]} [n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \quad (35)$$

$$+ n_{i_2}n_{i_1}(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_1})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_5)\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \quad (36)$$

$$+ n_5n_{i_1}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_1})^{\mathrm{T}}(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \quad (37)$$

$$+ n_{i_1}n_5(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_2})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \quad (38)$$

$$+ n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})\mathbf{T}_5^{\mathrm{T}}\boldsymbol{\ell}_5] \quad (39)$$

and this is equal to our constraint

The viewing geometry of a singly textured plane (quintilinear constraint): Suppose we have five cameras $(B_i, \mathbf{T}_i)$, and measure five lines $\hat{\boldsymbol{\ell}}_i$, which have indices n_i from a textured plane $\mathbf{H}$. We may form the $\boldsymbol{\ell}_i$ using the $\hat{\boldsymbol{\ell}}_i$ and the B_i and have the following constraint:

$$0 = \sum_{[i_1..i_5]\in \mathrm{P}^+(1..5)} n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})\mathbf{T}_{i_5}^{\mathrm{T}}\boldsymbol{\ell}_{i_5} \quad (40)$$

This constraint is the only constraint on a singly textured plane. Our other two constraints operate on doubly textured planes. The first constraint is a line/texture constraint. Let us assume that we have a doubly textured plane. We assume that we have six cameras, four of which view one singly textured plane and two of which view the other. We may reconstruct the $\mathbf{L}_{\lambda,1}$ of the singly textured plane from the first four cameras if we know the indices of the lines. We may reconstruct the $\mathbf{L}_{d,2}$ of the world line using the last two cameras, without knowing the indices. We know that these two quantities must be perpendicular, so that we get as a constraint:

The asymmetric viewing geometry of doubly textured plane (hexalinear constraint):

$$0 = \sum_{[i_1..i_4]\in \mathrm{P}^+[1..4]} n_{i_1}|\boldsymbol{\ell}_5\ \boldsymbol{\ell}_6\ \boldsymbol{\ell}_{i_1}||\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}|\mathbf{T}_{i_4}\boldsymbol{\ell}_{i_4} \quad (41)$$

The final constraint uses the textured plane constraint that

$$\mathbf{L}_{d,1}^{\mathrm{T}}\mathbf{L}_{m,2} + \mathbf{L}_{d,2}^{\mathrm{T}}\mathbf{L}_{m,1} = 0$$

We obtain

The symmetric viewing geometry of a doubly textured plane (octilinear constraint):

$$0 = \sum_{[i_1..i_8]\in \mathrm{sP}^+[1..8]} [n_{i_1}n_{i_2}n_{i_5}n_{i_6}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})^{\mathrm{T}}(\boldsymbol{\ell}_{i_5} \times \boldsymbol{\ell}_{i_6}) \cdot |\boldsymbol{\ell}_{i_1}\ \boldsymbol{\ell}_{i_2}\ \boldsymbol{\ell}_{i_7}|\boldsymbol{\ell}_{i_8}\mathbf{T}_{i_8}] \quad (42)$$

where sP^+ indicates the positive permutations among the first four and the last four indices, plus switching the first and last four indices wholesale. In other words, the indices are

$$\mathrm{P}^+[1..4]\mathrm{P}^+[5..8] \quad (43)$$

and

$$\mathrm{P}^+[5..8]\mathrm{P}^+[1..4] \quad (44)$$

Let us now discuss these constraints and their role in feedback mechanisms. Let's assume that we have a number of cameras looking at a scene and we know, with some uncertainty, the viewing geometry, i.e., the rigid transformation between any two views. Another way of saying this is that we have some knowledge about extrinsic calibration.

These patch equations (40, 41, 42) are the first geometric equations which constrain objects which do not, in the normal sense, correspond to each other. Instead, these objects exist only in "patch correspondence" to each other. The operation of the constraints is explicitly predicated on the assumption that we have some initial rough calibration, both rotationally and translationally. The constraints are also designed to operate only in a feedback mechanism which can take rough rotational or translational calibration and either improve said calibration, or state that it is not possible to improve said calibration.

First is the prismatic line constraint, which can be used to find rotation independently of translation. This constraint is most simply applied to the singly textured plane, though a more complicated theory can be constructed for generally textured planes. If we have patch correspondence with enough singly textured planes, then we may determine rotational calibration using the constraint. Note that this is a constraint which only needs patch correspondence.

The next three constraints contain integer indices. Recall our definition of the singly textured plane. For each

image line, its corresponding index indicates the ordinality of its generating world line in the singly textured plane. The next three constraints concern image lines together with their corresponding indices, plus the translational position of the cameras. All these constraints can be used in two opposite directions. One may use image lines plus approximate translations to obtain indices, or one may use image lines plus indices to obtain more accurate translations. This dual use of the constraints provides the basis of the feedback mechanism for going from approximate camera positions to more accurate camera positions.

The next three constraints also are defined on many cameras, probably far more than one would like to use in actual practice. Note, however, that we may "collapse" cameras if we have more than one measurement from the same textured plane. That is, if we have a constraint on lines ℓ_i, $1 \leq i \leq M$, we may have that for some j and k, $1 \leq \{j,k\} \leq M$, that $\mathbf{T}_j = \mathbf{T}_k$. Thus our quintilinear constraint could be considered to be on five, four, three, three, or even two cameras.

The first of these constraints is the quintilinear constraint. This equation constrains five image lines which are projections of the same singly textured plane. We may use it as described above to find indices or translations. This constraint is similar to the usual trilinear constraint as applied to lines.

The second of these constraints is the hexalinear constraint. This is a mixed constraint on a doubly textured plane. If you have images of six world lines, and you know that four are from one direction and two are from the other, then if you have indices and translations for the first four, then the constraint can be formed. So this constraint can be used in the two ways described above, but can also, if given the indices and translations, be used to constraint the rotations in the two cameras. This constraint has no analogue in the current equations.

The third of these constraints is the octilinear constraint. This is again a purely translational constraint on a doubly textured plane. If you have images of eight cameras, and you know that four are from one direction and two are from the other, then if you have indices and translations for all eight, the constraint can be formed. We can again use this constraint in the two ways outlined above as part of a feedback mechanism. This constraint is similar to the usual epipolar (or quadrilinear) constraint.

We now show how these constraints could be used for image processing. We first argue that one always has some initial calibration information. This is usually couched in terms of having multiple cameras "looking at the same object", but this is not a precise formulation. There is a relation between the depth of the viewed scene and the distance between the cameras which is the important factor for this initial calibration information. Indeed, if we were to have a camera on Earth and a camera on Venus looking at terrain, we would not expect to be able to correspond anything. However, if we turned the cameras towards the sky, we certainly could correspond, since our baseline is small related to the distance to the object. Thus any assumption which forms the basis for a correspondence method can always be expressed as a constraint on the error in calibration with respect to the distance to the scene.

Let us first look at an example of a scene for which it is impossible to compute correspondence. If we have a collection of textured planes, and these planes do not contain any wavelength greater than λ, then it is clear that if our camera positions are not known to accuracy at least less than λ, then it is impossible to compute any sort of correspondence. We may be able to compute the rotational calibration for the cameras from the patch correspondence, but after that we are stuck.

On the other hand, if we know our camera positions to within $\alpha << \lambda$, and we have many textures with wavelengths greater than λ, then it *is* possible to find our integer indices with a high degree of probability. The smaller α is, the higher the degree of probability that we can find the integer indices. Once we have the integer indices, we can turn the constraint around to improve the camera positions.

One can see the beginnings of the second part of the feedback mechanism taking shape. If we at first use patches with wavelengths much greater than our calibration error, we can then improve our camera position estimates. With these new estimates, we can then use patches with smaller wavelengths, which will be more accurate. Thus, a small number of these iterations should result in excellent positional estimates, depending on the frequency content of the scene. This geometric/statistical framework is developed in [27].

6 Conclusions

We have described a number of processes underlying structure from motion. We concentrated on basic aspects of the distortion of image motion or correspondence, the distortion of the viewing geometry or 3D motion and the distortion of shape. Our computational arguments suggest new avenues for studying the problem namely the development of feedback loops where all modules interact. We argued that an important component of the feedback should be patch (region, area) correspondence, i.e., the constraints that arise by corresponding entities where the relevant measurements are the outputs of filters with finite support. Textures are such a candidate. We developed a number of new constraints relating primitive textures to structure and motion. The study of the statistical properties of a patch in conjunction with multiple view geometry is one of our current projects. For example, one can develop mathematical theo-

rems relating the statistics of texture and the uncertainty in the viewing geometry to the ability to perform a match between textured patches [27]. In this paper we concentrated on a static scene. The real problem, however, must address scenes which contain independent motion. We are working on this problem by addressing the structure from motion problem in scale space. For an example of our approach, see [28]. Finally, our ability to initiate feedback processes allows us to perform calibration of networks of hundreds of cameras, to an unprecedented degree of accuracy. Thus, we can calibrate negative Argus eyes, such as camera networks observing a specific area (Fig. 31). Observing then non-rigid movement in that area allows reconstruction of shape through a combination of stereo and video cues. This gives rise to 3D video and other applications, but, most importantly, it provides a new representation for action, namely 3D motion fields. This is discussed in detail in [29].

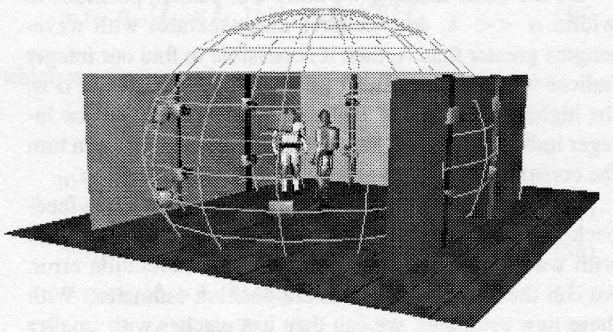


Figure 31.

References

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

[2] A. Bruss and B. K. P. Horn, "Passive navigation," *Computer Vision, Graphics, and Image Processing*, vol. 21, pp. 3–20, 1983.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[4] C. Fermüller and Y. Aloimonos, "Statistics explains geometric optical illusions," in *Foundations of Image Understanding* (L. S. Davis, ed.), pp. 409–446, Boston: Kluwer Academic Publishers, 2001.

[5] W. Fuller, *Measurement Error Models*. New York: Wiley, 1987.

[6] C. Fermüller, H. Malm, and Y. Aloimonos, "Statistics explains geometrical optical illusions," Tech. Rep. CAR-TR-968, Center for Automation Research, University of Maryland, 2001.

[7] "`http://www.illusionworks.com`."

[8] B. K. P. Horn, *Robot Vision*. New York: McGraw Hill, 1986.

[9] H. Ouchi, *Japanese and Geometrical Art*. New York: Dover, 1977.

[10] C. Fermüller, D. Shulman, and Y. Aloimonos, "The statistics of optical flow," *Computer Vision and Image Understanding*, vol. 82, pp. 1–32, 2001.

[11] C. Fermüller, "Passive navigation as a pattern recognition problem," *International Journal of Computer Vision*, vol. 14, pp. 147–158, 1995.

[12] C. Fermüller and Y. Aloimonos, "Direct perception of three-dimensional motion from patterns of visual motion," *Science*, vol. 270, pp. 1973–1976, 1995.

[13] C. Fermüller and Y. Aloimonos, "On the geometry of visual correspondence," *International Journal of Computer Vision*, vol. 21, pp. 223–247, 1997.

[14] B. K. P. Horn and E. J. Weldon, "Computationally efficient methods for recovering translational motion," in *Proc. International Conference on Computer Vision*, pp. 2–11, 1987.

[15] C. Fermüller, "Navigational preliminaries," in *Active Perception* (Y. Aloimonos, ed.), Advances in Computer Vision, ch. 3, Hillsdale, NJ: Lawrence Erlbaum Associates, 1993.

[16] C. Fermüller and Y. Aloimonos, "Qualitative egomotion," *International Journal of Computer Vision*, vol. 15, pp. 7–29, 1995.

[17] G. Xu and Z. Zhang, *Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach*. Kluwer Academic Publishers, 1996.

[18] K. Daniilidis and M. E. Spetsakis, "Understanding noise sensitivity in structure from motion," in *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles* (Y. Aloimonos, ed.), Advances in Computer Vision, ch. 4, Mahwah, NJ: Lawrence Erlbaum Associates, 1997.

[19] K. Daniilidis, *On the Error Sensitivity in the Recovery of Object Descriptions*. PhD thesis, Department of Informatics, University of Karlsruhe, Germany, 1992. In German.

[20] C. Fermüller and Y. Aloimonos, "Observability of 3D motion," *International Journal of Computer Vision*, vol. 37, pp. 43–63, 2000.

[21] C. Fermüller and Y. Aloimonos, "Ambiguity in structure from motion: Sphere versus plane," *International Journal of Computer Vision*, vol. 28, pp. 137–154, 1998.

[22] C. Fermüller and Y. Aloimonos, "Geometry of eye design: Biology and technology," Tech. Rep. CAR-TR-901, Center for Automation Research, University of Maryland, 1998.

[23] P. Baker, C. Fermüller, and Y. Aloimonos, "A spherical eye from multiple cameras (or how to make better models)," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, (Hawaii), 2001.

[24] T. Brodský, C. Fermüller, and Y. Aloimonos, "Structure from motion: Beyond the epipolar constraint," *International Journal of Computer Vision*, vol. 37, pp. 231–258, 2000.

[25] C. Fermüller, L. Cheong, and Y. Aloimonos, "Visual space distortion," *Biological Cybernetics*, vol. 77, pp. 323–337, 1997.

[26] J. J. Koenderink, *Solid Shape*. MIT Press, 1992.

[27] P. Baker, *Geometry and Statistics of Visual Correspondence*. PhD thesis, Department of Computer Science, University of Maryland, 2002. in preparation.

[28] C. Fermüller, T. Brodský, and Y. Aloimonos, "Motion segmentation: A synergistic approach," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 226–231A, 1999.

[29] Y. Aloimonos and C. Fermüller, "Analyzing action representations," in *Algebraic Frames for the Perception-Action Cycle* (G. Sommer and Y. Y. Zeevi, eds.), Springer-Verlag, 2000.