Visual Space-Time Geometry and Statistics

Cornelia Fermüller, Patrick Baker and Yiannis Aloimonos
Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3275, USA
fer@cfar.umd.edu

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 (image-based rendering, motion capture) and robotics (navigation). 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

BMVC 2002 doi:10.5244/C.16.1

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.

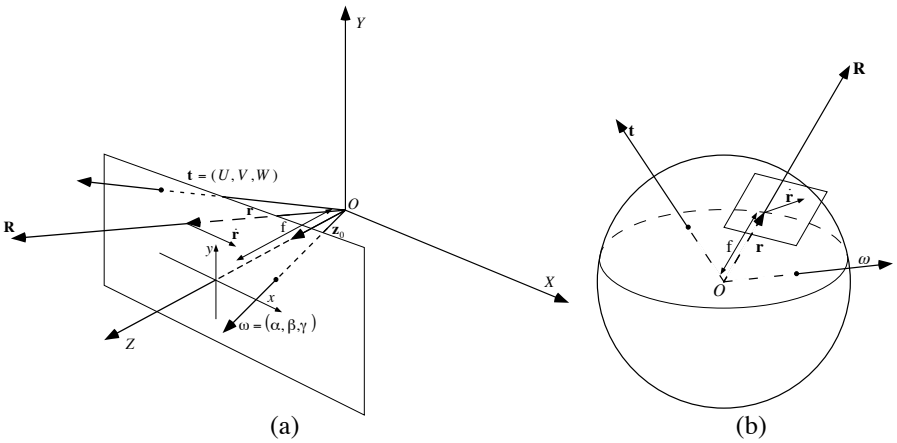


Figure 1: Image formation on the plane (a) and on the sphere (b). The system moves with a rigid motion with translational velocity $\mathbf{t}$ and rotational velocity $\boldsymbol{\omega}$. Scene points $\mathbf{R}$ project onto image points $\mathbf{r}$ and the 3D velocity $\dot{\mathbf{R}}$ of a scene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

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 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 [23]:

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

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 [6].

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. It turns out that this same principle is a cause of a large number of geometric optical illusions. Thus, we choose to explain this inherent uncertainty by employing a few examples of well-known (unexplained until now) 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, constitutes 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 the 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 [20]. 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. What does bias mean? In general, we have abailable noisy measurements and we use a procedure—which we call the estimator—to derive from these measurements a quantity, let's call it *parameter* x. Any particular small set of measurements leads to a different value for parameter x. Assume we perform the estimation of x using different sets of measurements many times. The mean of estimates x (that is, the average number of an infinite number of values) is called the *expected value* of x. If the expected value is equal to the true value, the estimate is called *unbiased*, otherwise it is *biased*. In the case of the interpretation of images a significant amount of data is used. Features are computed from image values in extended spatial areas acquired at different instants. The extraction of features is some form of estimation process, for which the mean (and the bias) are inherent properties. Thus the bias is justified in the explanation of the perception of features.

(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 2 and 3 they appear bulging and wiggly and in 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 [25].

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 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).

This suffices to explain the main cause underlying the three illusions in Figs. 2–4, and several others [14], as shown in Figs. 6, 7 and 8.

Fig. 2 (from [1]) consists of a black square grid on a white background with small black squares superimposed. It gives the perception of the straight grid lines being concave and convex curves. The effect can easily be understood using the above observations. The grid consists of lines (or bars), and the effect of smoothing on the bars is to drift the two edges apart. At the locations where a square is aligned with the grid, there is only one edge, and this edge stays in place. The net effect of smoothing is that edges of grid lines

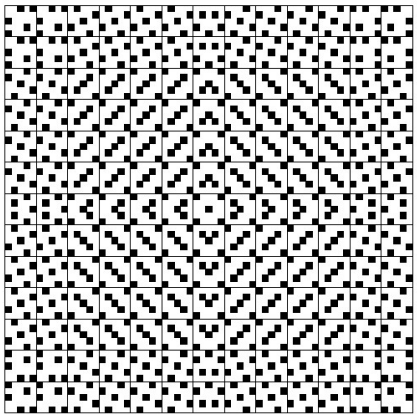


Figure 2: “Spring” illusory pattern.

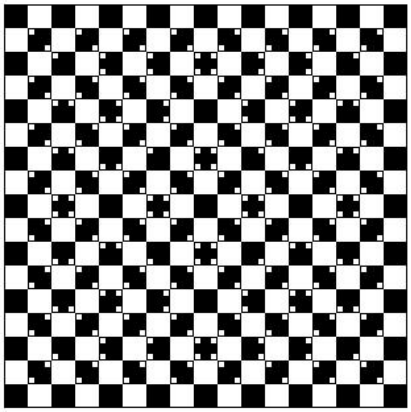


Figure 3: “Waves” illusory pattern.

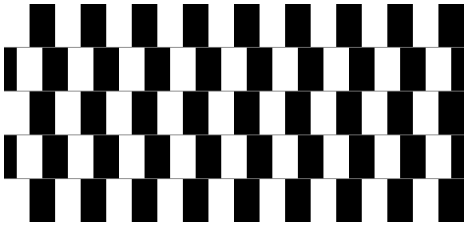


Figure 4: Café wall illusion.

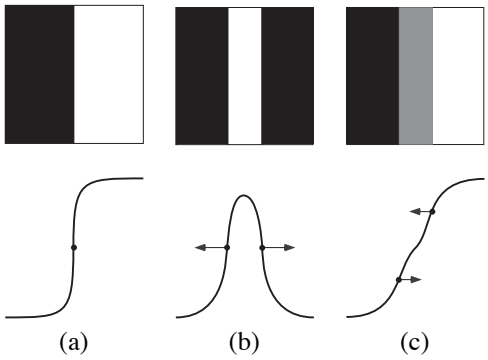


Figure 5: A schematic description of the behavior of edge movement (drift) in scale space: (a) no movement, (b) drifting apart, (c) getting closer.

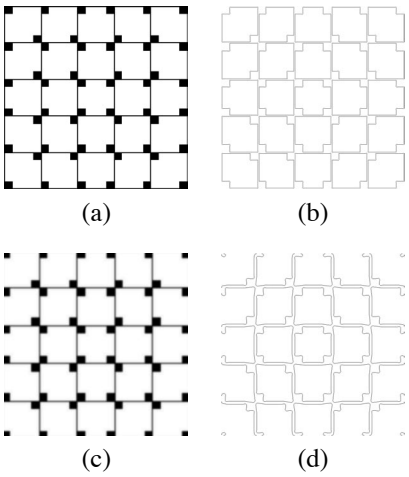


Figure 6: (a) Small part of Fig. 3, (b) edge detection without smoothing, (c) Gaussian smoothing, and (d) smoothing and edge detection have been applied.

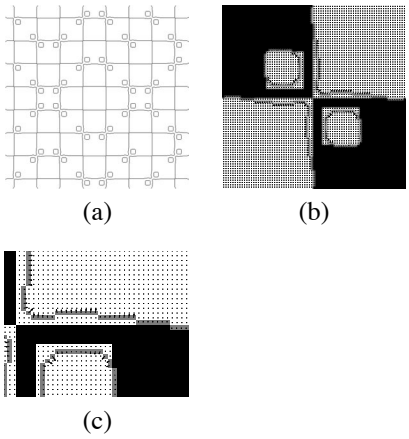


Figure 7: (a) The result of smoothing and edge detection on a part of the figure. (b) and (c) The drift velocity at edges in the smoothed image logarithmically scaled for parts of the figure.

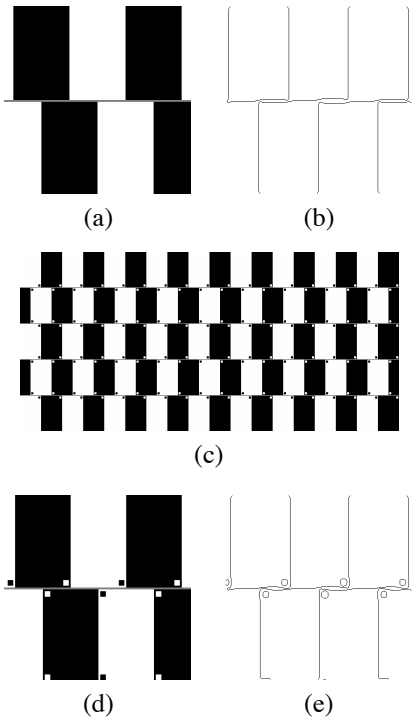


Figure 8: (a) Small part of the figure. (b) Result of smoothing and edge detection. (c) Modified café wall pattern, (d), (e), small part and edge detection.

are no longer straight as illustrated in (Fig. 6d) which shows the result of edge detection on the smoothed image in comparison to Fig. 6b which shows edge detection on the raw image. The illusion in Fig. 3 is explained in an exactly analogous manner (see Fig. 7).

The famous "café wall" illusion shown in Fig. 4 consists of a black and white checkerboard pattern with alternate rows shifted one half-cycle and with thin mortar lines midway in luminance between the black and white squares separating the rows.

At the locations where a mortar line borders both a dark tile and a bright tile the two edges move toward each other under smoothing, and for thin lines it takes a relatively small amount of smoothing for the two edges to merge into one. Where the mortar line is between two bright regions or where it is between two dark regions the edges move away from each other. The results of smoothing and edge detection are illustrated in Fig. 8b for a small part of the pattern shown in Fig. 8a. It can be seen that the edge elements which form the boundaries of the tiles are tilted with the same sign of slope as perceived. (There are two kinds of overlapping edge elements, the ones form the lower boundary of a white tile and the upper boundary of a black tile, and the others form the lower boundary of a black tile and the upper boundary of a white tile.)

If bias is indeed the main cause of the illusion then we should be able to counteract the effect by introducing additional elements. This has been pursued in Fig. 8c; the additional white and black squares put in the corners of the tiles remove the illusory effect. Fig. 8d shows a small part of the pattern and Fig. 8e shows the edges detected. As can be seen from the figure, the inserted squares partly compensate for the drifting in opposite directions of edges along the mortar line separating tiles of the same gray level. As a result slightly wavy edgels are obtained; but the "waviness" is too weak to be perceived (low amplitude, high frequency) and as a result a straight line without tilt is seen.

A full account of the perception of tilted lines requires additional explanation. The illusion is due to two processing stages. In the first stage local edge elements are computed and bias explains the tilting of these elements. The second stage consists of the integration of these local elements into longer lines. Our hypothesis is that this integration is computationally an approximation of the longer lines using as input the positions and orientations of the short line elements. Such an approximation gives rise to a line with a tilt. It also explains other illusions, such as one of the most forceful of all illusions, the Fraser spiral pattern, which consists of circles made of black and white elements which together form something rather like a twisted cord, on a checkerboard background. The twisted cord gives the perception of being spiral shaped, rather than like circles. The individual black and white elements which make up the cord are sections of spirals, thus also the edges at the borders of the black and white lines are along theses directions and the approximation process will fit spirals to them.

(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, parameterized 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 $\mathbf{x}_0 = (x_0, y_0)$.

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:

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

Equation (4) is the equation of the straight line on which an edgel with center (x'_0, y'_0) and gradient (I'_x, I'_y) lies. This equation is approximated by the measurements. Let n be the number of measurements. Each measurement i provides one equation

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

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 [19] 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 [17] 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

$$\mathbf{x} = \mathbf{x}' + nM'^{-1}(\bar{\mathbf{x}}_0 - \mathbf{x}')\sigma_s^2 \tag{7}$$

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 $\mathbf{x}_{0_i}$. The second term on the righthand side of (7) represents the bias.

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 [24]). 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.

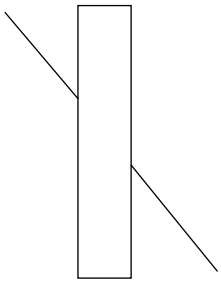
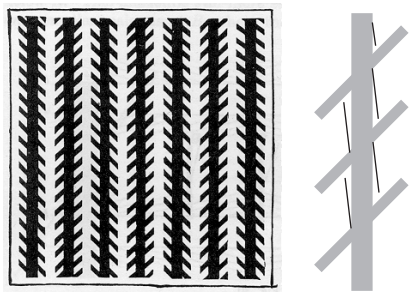


Figure 9: Poggendorff illusion.



(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.

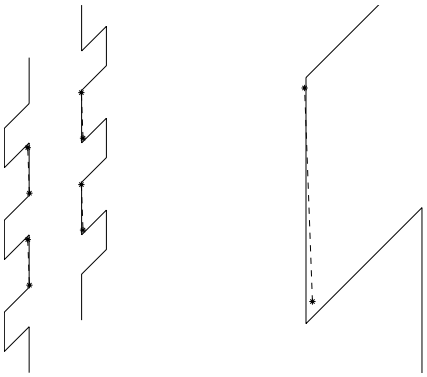


Figure 11: Estimation of edges in Zöllner pattern. The line elements are found by connecting two consecutive intersection points, resulting from the intersection of edges of two consecutive bars with the edge of the vertical bar (one in an obtuse and one in an acute angle). The data consists of edge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

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 [14].

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.

(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 [21]. We assume the optical flow to be constant within a small 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

$$\mathbf{u} = \mathbf{u}' - n\sigma_s^2 M'^{-1} \mathbf{u}', \tag{11}$$

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 [26]. The pattern consists of two rectangular checkerboard patterns oriented in orthogonal directions—a background orientation surrounding an inner ring. Small retinal motions, or slight movements of the paper, cause a segmentation of the inset pattern, and motion of the inset relative to the surround.

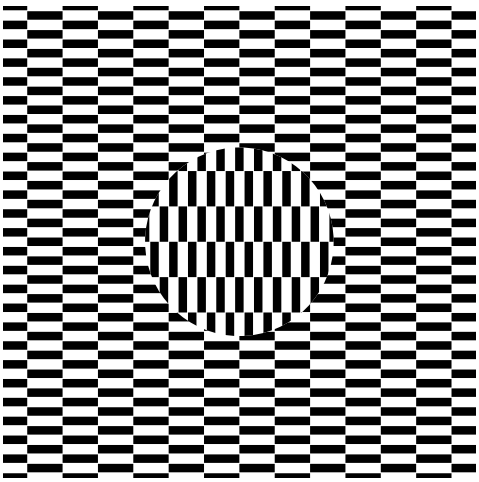


Figure 12: A pattern similar to the one by Ouchi.

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

We have thus shown that points, lines and image motion will be biased. 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 [18].

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 whatever way we extract points, lines or local movement, we start with an unavoidable error in the measurements. 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 mo-

tion. 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 [23, 9, 10, 12, 22]. In the sequel we discuss 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 epipolar and visibility constraints

If correspondence or image motion is available, the epipolar constraint shows how the image measurements are related to the 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 $(\mathbf{t} \times \mathbf{r}) \cdot (\dot{\mathbf{r}} + \boldsymbol{\omega} \times \mathbf{r}) = 0$ [8]. 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.

$$M_{ep} = \iint_{\text{image}} \left[(\hat{\mathbf{t}} \times \mathbf{r}) \cdot (\dot{\mathbf{r}} + \hat{\boldsymbol{\omega}} \times \mathbf{r}) \right]^2 d\mathbf{r} \tag{12}$$

If, however, the image motion field is not known and we have at our disposal only one component of it along the image gradient, i.e., if only the normal flow is known, then the epipolar constraint cannot be applied. In this case, if $\dot{\mathbf{r}}$ is the motion vector at a point (x, y) and $\mathbf{n} = (n_x, n_y, 0)$ is a unit vector in gradient direction, the normal motion $\mathbf{u}_\mathbf{n}$ is

$$\mathbf{u}_\mathbf{n} = (\dot{\mathbf{r}} \cdot n) \cdot \mathbf{n}. \tag{13}$$

The difference between the cases of correspondence or flow vs. normal flow is already clear. In the former case, (3) is valid. This is a vector equation on the image plane (i.e., it consists of two scalar equations). In the latter case, only (13) is available, and this is a single scalar equation. In this case, only the visibility constraint can be applied. This constraint states that the scene, to be visible, has to lie in front of the camera. In other words, the depth Z of the scene has to be positive. Substituting (3) into (13) and solving for the estimated depth $\hat{Z}$ or range $\hat{R}$, we obtain for an estimate $\hat{\mathbf{t}}, \hat{\boldsymbol{\omega}}$ at each point $\mathbf{r}$:

$$\hat{Z}(\text{or } \hat{R}) = \frac{\mathbf{u}^{\mathrm{T}}(\hat{\mathbf{t}}) \cdot \mathbf{n}}{(\dot{\mathbf{r}} - \mathbf{u}_{\text{rot}}(\hat{\boldsymbol{\omega}})) \cdot \mathbf{n}}. \tag{14}$$

If the numerator and denominator of (14) have opposite signs, negative depth is computed. Thus, to utilize the positivity constraint one must search for the motion $\hat{\mathbf{t}}, \hat{\boldsymbol{\omega}}$ that produces

a minimum number of negative depth estimates. Formally, if $\mathbf{r}$ is an image point, define the indicator function

$$I_{nd}(\mathbf{r}) = \begin{cases} 1 \text{ for } \left(\mathbf{u}^{\mathrm{T}}(\hat{\mathbf{t}}) \cdot \mathbf{n}\right) \left((\dot{\mathbf{r}} - \mathbf{u}_{\mathrm{rot}}(\hat{\boldsymbol{\omega}})) \cdot \mathbf{n}\right) < 0 \\ 0 \text{ for } \left(\mathbf{u}^{\mathrm{T}}(\hat{\mathbf{t}}) \cdot \mathbf{n}\right) \left((\dot{\mathbf{r}} - \mathbf{u}_{\mathrm{rot}}(\hat{\boldsymbol{\omega}})) \cdot \mathbf{n}\right) > 0 \end{cases} .$$

Then estimation of 3D motion from normal flow amounts to minimizing [10, 11, 23] the function

$$M_{nd} = \iint_{\text{image}} I_{nd}(\mathbf{r}) d\mathbf{r}. \tag{15}$$

Finding 3D motion through the minimization of negative depth amounts to a search problem which, if done blidnly, is five-dimensional. In [9, 10, 12] geometric constraints have been discovered that reduce the search to multiple three-dimensional searches. In addition, the introduced techniques become pattern recognition procedures because they amount to searching for global patterns of image measurements. These patterns encode the 3D motion parameters.

Algorithms based on minimizing negative depth are robust and fully automatic as they do not rely on correspondence. Extensive experimentation, however, demonstrates that there is always some uncertainty associated with the solution. For example, the direction of translation $\mathbf{t}$ is estimated as the intersection of $\mathbf{t}$ with the image plane. The solution found, however, is not a single point, but an area, which obviously contains the solution. This area could be small or elongated and this appears to be the best one can do using this constraint.

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.

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 [27]. Almost all techniques are based on the so-called epipolar constraint, which was explained before.

Experience has shown that estimating 3D motion by minimizing the epipolar error, 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 [7, 8] 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 [13] 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 (12). 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 (12) 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 (12). Expressing $\dot{\mathbf{r}}$ in terms of the real motion, function (12) 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

$$E_{ep} = \int_{Z=Z_{\min}}^{Z=Z_{\max}} M_{ep} dZ \tag{16}$$

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 (16) that we shall uncover should be interpreted as being likely to occur. Our approach expresses function (16) 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 (16) 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

$$\begin{aligned} E_{ep} = \pi e^2 \Bigg(& (Z_{\max} - Z_{\min}) \left(\frac{1}{3}\gamma_\epsilon^2 e^4 + \frac{1}{4}\Big(\gamma_\epsilon^2 \left(\hat{x}_0^2 + \hat{y}_0^2 \right) \right. \\ & \left. + 6\gamma_\epsilon \left(\hat{x}_0 \alpha_\epsilon + \hat{y}_0 \beta_\epsilon \right) + \alpha_\epsilon^2 + \beta_\epsilon^2 \Big) e^2 (\hat{x}_0 \alpha_\epsilon + \hat{y}_0 \beta_\epsilon)^2 \right) \\ & + (\ln(Z_{\max}) - \ln(Z_{\min})) \left(\frac{1}{2} (3\gamma_\epsilon (x_{0_\epsilon} y_0 - y_{0_\epsilon} x_0) + x_{0_\epsilon} \beta_\epsilon \right. \\ & \left. - y_{0_\epsilon} \alpha_\epsilon) e^2 + 2 \left(x_{0_\epsilon} y_0 - y_{0_\epsilon} x_0 \right) (\hat{x}_0 \alpha_\epsilon + \hat{y}_0 \beta_\epsilon) \right) + \left(\frac{1}{Z_{\min}} \right. \\ & \left. - \frac{1}{Z_{\max}} \right) \left(\frac{1}{4} \left(y_{0_\epsilon}^2 + x_{0_\epsilon}^2 \right) e^2 + \left(x_{0_\epsilon} y_0 - y_{0_\epsilon} x_0 \right)^2 \right) \Bigg) \end{aligned} \tag{17}$$

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 (17) 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

$$\begin{aligned} \alpha_\epsilon &= \frac{y_{0_\epsilon} (\ln(Z_{\max}) - \ln(Z_{\min}))}{Z_{\max} - Z_{\min}} \\ \beta_\epsilon &= \frac{-x_{0_\epsilon} (\ln(Z_{\max}) - \ln(Z_{\min}))}{Z_{\max} - Z_{\min}} \\ \gamma_\epsilon &= 0 \end{aligned} \tag{18}$$

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

$$x_{0_\epsilon} = x_0 \quad y_{0_\epsilon} = y_0 \quad \beta_\epsilon = -\alpha_\epsilon \frac{x_0}{y_0} \quad \gamma_\epsilon = 0$$

$$\alpha_\epsilon = 2y_0 Z_{\min} Z_{\max} \left(\ln\left(Z_{\max}\right) - \ln\left(Z_{\min}\right)\right) \Big/ \Big(\left(Z_{\max} - Z_{\min}\right)\left(Z_{\max} Z_{\min} - 1\right) + \Big(\left(Z_{\max} - Z_{\min}\right)^2 \left(Z_{\max} Z_{\min} - 1\right)^2 + 4 Z_{\max}^2 Z_{\min}^2 \left(\ln\left(Z_{\max}\right) - \ln\left(Z_{\min}\right)\right)^2 \Big)^{1/2} \Big)$$

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. 13 shows such a valley for two frames of a video sequence. Video 1 [http://www.cfar.umd.edu/users/yiannis/proc-ieee/video01.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. 14 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 2 [video02.mpg] we show the valleys for all video frames for the translation for the underlying camera motion that captured the video in Video 3 [video03.mpg]. Fig. 15a shows just the valley for two frames of the video, one frame of which is shown in Fig. 15b. 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 4 [video04.mpg]. Video 5 [video05.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.

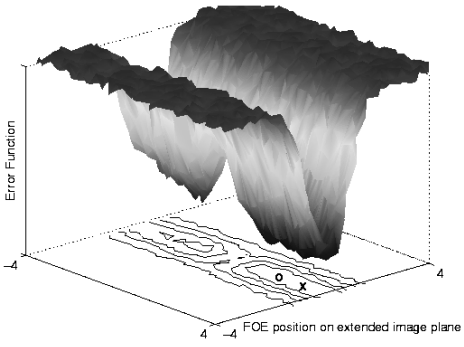


Figure 13: Minimizing E (translation only).

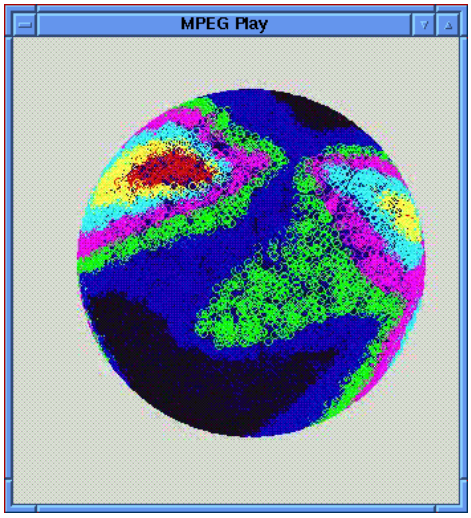
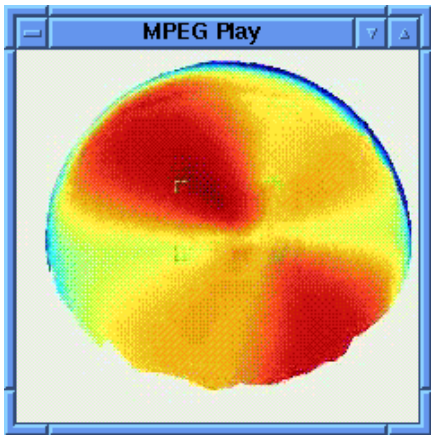


Figure 14: The valley of the optimizing function painted on the sphere.

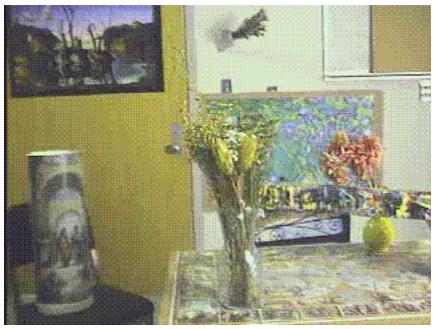
The preceding analysis on the ambiguity was based on the minimization of the epipolar constraint. But if we minimize instead the negative depth constraint, we will find very similar results (see [13]). Thus, the ambiguity is not an artifact of the technique used but is inherent in the problem.

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 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



(a) The translation valley for two frames of the sequence. The image size is denoted by four corners



(b) One frame from a sequence.

Figure 15:

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!

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

$$\dot{\mathbf{r}} = \frac{1}{|\mathbf{R}|f}\left((\mathbf{t}\cdot\mathbf{r})\,\mathbf{r} - \mathbf{t}\right) - \boldsymbol{\omega}\times\mathbf{r} = \frac{1}{R}\,\mathbf{u}_{\mathrm{tr}}(\mathbf{t}) + \mathbf{u}_{\mathrm{rot}}(\boldsymbol{\omega}). \tag{19}$$

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 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 $\boldsymbol{\omega}_\epsilon \perp \mathbf{t}$.

The analysis of the negative depth optimization constraint, which is much harder than the case of epipolar minimization, provides similar results (see [13]).

Table 1 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

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 $\gamma_\epsilon = 0$, $\alpha_\epsilon/\beta_\epsilon = -x_{0_\epsilon}/y_{0_\epsilon}$, $x_0/y_0 = x_{0_\epsilon}/y_{0_\epsilon}$

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 [10]. 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. 16, 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!

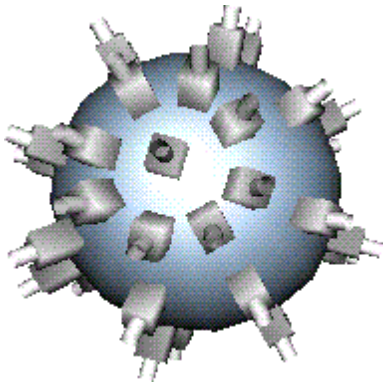


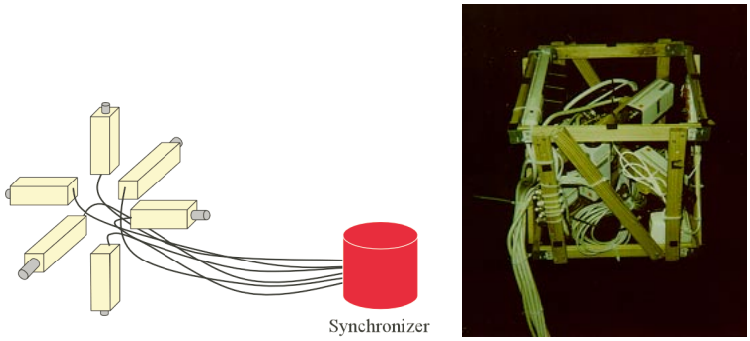
Figure 16: A compound-like eye composed of conventional video cameras.

An eye like the one in Fig. 16 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. 16 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. 17a shows a schematic version of

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

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



(a) (b)

Figure 17: The Argus eye. (a) Schematic. (b) Implementation.

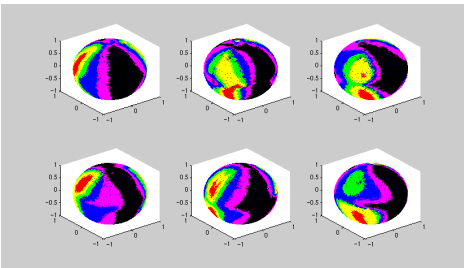


Figure 18: Initial valleys from each camera (translation).

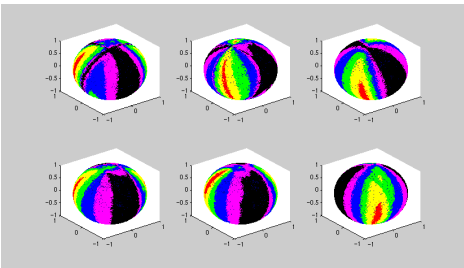


Figure 19: Translation valleys after derotation.

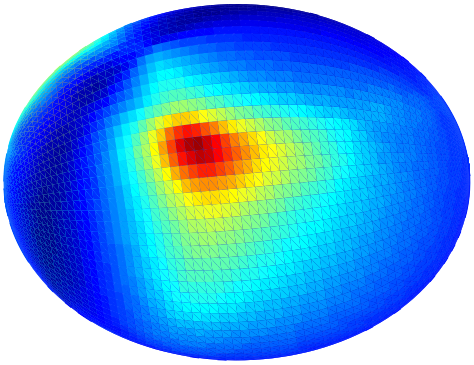


Figure 20: Valley for the whole system (translation).

the Argus eye, consisting of six cameras arranged to point outwards. Fig. 17b shows an actual view of the system, and Video 6 [video06.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. 18. 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 7 [video07.mpg] shows the growth of these surfaces and their intersection.) If we then derotate each one of the videos the valleys become much thinner (Fig. 19), and bringing them all to the same coordinate system provides a unique translation for the whole system, as shown in Fig. 20. Using these values, one can perform impressive reconstructions of the scene, with many applications to graphics and augmented reality [4].

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 8 [video08.mpg] denoting a sequence and Video 9 [video09.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 10a [video10a.mpg] (a Pooh game), we recovered the shape shown in

Video 10b [video10b.mpg], using the algorithms in [5]. 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 [16] that the transformation relating the computed depth to the actual depth is a Cremona transformation. This is easy to see. If $\hat{\mathbf{t}}, \hat{\boldsymbol{\omega}}$ are the estimated 3D motion parameters and $\mathbf{t}_\epsilon = \mathbf{t} - \hat{\mathbf{t}}$, $\boldsymbol{\omega}_\epsilon = \boldsymbol{\omega} - \hat{\boldsymbol{\omega}}$ 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

$$\hat{Z} = Z \cdot D \quad \text{with} \quad D = \frac{\mathbf{u}_{\rm tr}(\hat{\mathbf{t}}) \cdot \mathbf{n}}{\mathbf{u}_{\rm tr}(\mathbf{t}) \cdot n + Z\mathbf{u}_{\rm rot}(\hat{\boldsymbol{\omega}}_\epsilon) \cdot \mathbf{n}} \tag{20}$$

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 [16].

Equation (20) 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 [16].

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]

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

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

5.1 The reason for feedback

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. However, this view does not offer an effective way to uncover the intricacies of the feedback process. In [3] a new theory is developed which introduces a feedback loop that operates on patches rather than points and lines. We outline this theory here.

The solution to the structure from motion problem is contaminated by errors in the measurement of features, as we have seen in the preceding analysis. We call this the "small correspondence problem", and it stands in opposition to the "large correspondence problem", which is the standard correspondence problem known in the image processing literature. The large correspondence problem considers matching features in two images, rather than localizing a feature in one image. One can think of the small correspondence problem as a sort of self-correspondence within an image and the large correspondence as a correspondence between images. Both of these problems affect the accuracy of any structure from motion problem.

Let us outline how the small correspondence problem affects structure from motion algorithms. The bias would not cause a problem if there were equal bias independent of which image a feature appears in, because in that case we would only be considering a different world point than we thought, but still the same world point, so our equations would hold. The difficulty comes when our bias is different in the different images, and because of foreshortening effects, this is often the case. In the situation of figure 21, the localization of the corner point in the blue image will be closer to the true corner while the corner point in the red image will be farther from the true corner. In this case we will not be using the same world point as input to our structure from motion equations, violating their assumptions. We will show in the following sections how this small correspondence

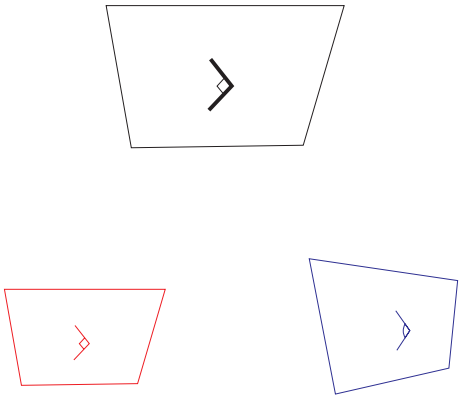


Figure 21: The red camera is at the same orientation as the black world plane while the blue camera is at a different orientation. The corner angle in the blue image is greater than ninety degrees.

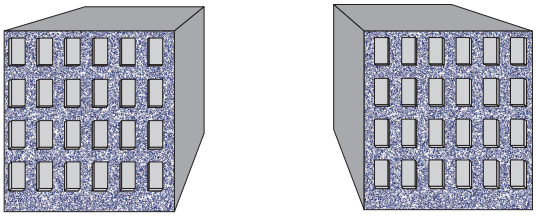


Figure 22: Correspondence between these images of the same building is impossible with current techniques

problem can be overcome considering only the rotational transformation between cameras and independent of the translation between cameras.

The large correspondence problem presents a different difficulty, and it is generally recognized as an impediment to successful automatic structure from motion algorithms. Corresponding between two images is hard because of the aliasing problem. That is, if you have two scenes which have similar looking features it is difficult to tell them apart. In figure 22 we see two views of the same building. Since the corners of the windows are identical, we cannot correspond them with purely local information. The formulas in our new framework show how to add the global positional information to the local signal information to resolve this ambiguity when it is possible. These formulas consider translation, but operate after the rotation has already been calculated. Based on this framework, we postulate an ordering in egomotion calculation where rotation is computed first and translation is computed later.

A central idea of this framework is the idea of patch correspondence. We make the assumption that when we look at a scene with multiple cameras, we can find regions which belong to the same planar patch. Using this assumption, we can formulate theorems about our ability to find structure and motion. In figure 23, the blue regions are in patch correspondence.

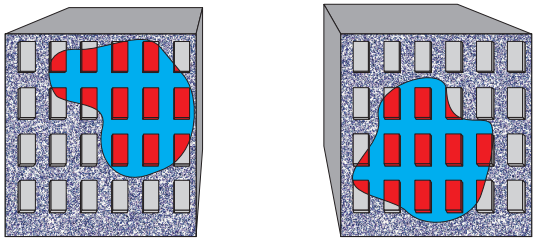


Figure 23: The blue regions are considered to be in patch correspondence

5.2 Cameras and Projection in a Fiducial Coordinate System

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, although we will often use "rotation" as a shorthand for rotation/internal calibration since the only different is an orthogonality constraint on a 3×3 matrix. 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 $C : \mathbb{R}^3 \rightarrow \mathbb{P}^2$ from world points to image points. Given a fiducial coordinate system, we may represent this map with a pair $(B, \mathbf{T})$, where $B : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is a linear function (a 3×3 matrix representing rotation and internal calibration), 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. The rotation is included in B and the translation is $\mathbf{T}$.

This formulation is somewhat different from the standard formulation for projection of points, but it allows us to easily project world lines into our cameras. We use Plücker coordinates for lines because they allow for easy projection equations. 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:

$$\mathbf{L}_d = \mathbf{Q}_2 - \mathbf{Q}_1 \qquad \text{direction of } L \tag{21}$$

$$\mathbf{L}_m = \mathbf{L}_d \times \mathbf{P} \qquad \text{moment of } L \tag{22}$$

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 also $\mathbf{L}_m^T \mathbf{L}_d = 0$.

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

$$\hat{\boldsymbol{\ell}} = B^{-\mathrm{T}}(\mathbf{L}_m - \mathbf{T} \times \mathbf{L}_d) \tag{23}$$

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.

We promised to formulate a theory based on patches rather than points and lines, so now we introduce the singly textured plane. We would like to represent a periodic function in one dimension extended to span a plane. We represent this object as a set of equally spaced parallel lines embedded in a plane. That is, a set of parallel lines is a representation of a sinusoid. According to Fourier's theorem, any (nice) function is the sum of harmonic components. Thus, a set of parallel lines is a new atom, a new fundamental object that is hidden in a textured patch. Pulling a harmonic component out of an image patch, is an important question but not addressed in this paper.

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 $\mathbf{L}_0 = \begin{bmatrix} \mathbf{L}_d \\ \mathbf{L}_m \end{bmatrix}$. 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

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

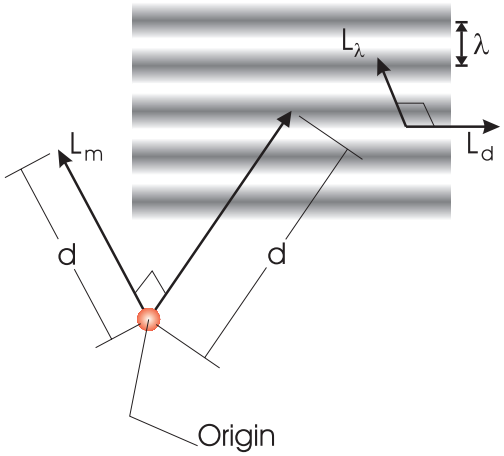


Figure 24: The Parameters of a Textured Plane

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, illustrated in figure 24. 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:

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

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

$$\{\hat{\boldsymbol{\ell}}_n : \hat{\boldsymbol{\ell}}_n = B^{-\mathrm{T}}(\mathbf{L}_m + n\mathbf{L}_\lambda - \mathbf{T} \times \mathbf{L}_d)\} \tag{24}$$

where $n \in \mathbb{Z}$.

Finally, a comment on nomenclature. In this section we do not consider the case of a moving camera collecting video but the general case of a set of cameras looking at the same scene. In this case, the structure from motion problem amounts to finding the rigid transformation relating any two cameras. This same problem is also known as (extrinsic) calibration. Thus, we will be referring to the terms structure from motion and calibration interchangeably.

5.3 Reconstruction of textured plane

In order to formulate the constraints our philosophy is to find equations to reconstruct world objects using image objects and then to use world constraints to find the constraints on the images. To obtain the standard epipolar, trilinear, and quadrilinear constraints, we can reconstruct a world line using two cameras and then intersect world lines to obtain constraints on four cameras. We do not consider constraints on the world lines but instead consider only constraints on the textured plane, which can be specialized to include the standard constraints.

Also, note that these equations are formulated on many cameras (up to eight), but that cameras can be considered to be at the same position and therefore identical. We might measure two, or even three lines from the image of a textured plane and therefore the eight camera constraint might operate on two cameras.

Let us now consider the reconstruction of a singly textured plane. The derivation is too long for this paper. For full details see [3]. We must first describe the notation

$$\sum_{[i_1..i_n]\in \mathrm{P}^+[1..n]} \tag{25}$$

This is a summation which goes over all of the *positive* permutations of $[1..n]$, putting each permutation into the indices $[i_1..i_n]$. Also, as Fig. 25 shows, each camera can look at a different line. The index of the line which the camera is viewing is called its line index, and figures in the equations which follow.

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 shown in figure 25 and in the following equation.

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

Note that $|\cdot|$ is the *signed* magnitude, and since the coordinates are homogeneous, it does not matter which sign is chosen. It is important, however, that the sign is consistent across the three components of the reconstruction. This equation is multilinear in all its parameters, which allows for easy analysis for later proofs.

We do not need to know that integer index of the lines in order to reconstruct the $\mathbf{L}_d$. This is important as it relates to the correspondence problem, because it means that correspondence is irrelevant to find the direction of the lines in the singly textured plane. We can also find the component $\mathbf{L}_d$ of the textured plane even if we only have two image lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ which form a nonzero cross product. This cross product $\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$ will be the direction of the lines of the singly textured plane.

If we have a plane which contains two textures a and b which are in different directions, we can use the above result to find the normal to the plane if this plane is viewed

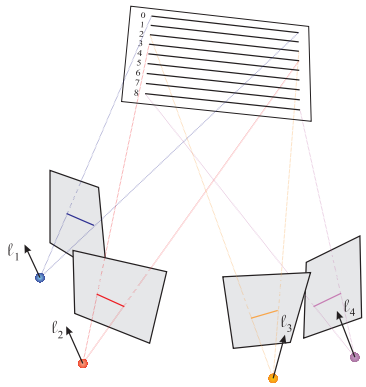


Figure 25: Reconstructing a textured plane.

by cameras 1 and 2. We merely must form the normal $\mathbf{n}$ as

$$\mathbf{n} = (\boldsymbol{\ell}_{1,a} \times \boldsymbol{\ell}_{2,a}) \times (\boldsymbol{\ell}_{1,b} \times \boldsymbol{\ell}_{2,b}) \tag{26}$$

Note that the translation plays no part in the calculation of the surface normal, and we only need the derotated $\boldsymbol{\ell}$, so that shape can be computed just by knowing rotation.

5.4 Prismatic line constraint and rotational feedback

Using the reconstruction of the direction of the textured plane lines we can form a constraint on three cameras. 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 or the line indices. This means that merely having correspondence between patches is enough to obtain rotation without having to find exact correspondence.

Pick three image lines $\boldsymbol{\ell}_i$ in three cameras. Note that the $\boldsymbol{\ell}_i$'s do not necessarily 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 $\hat{\boldsymbol{\ell}}_2^T B_2^{-1}(B_1^{-T}\hat{\boldsymbol{\ell}}_1 \times B_3^{-T}\hat{\boldsymbol{\ell}}_3) = 0$. 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. See figure 26 for a diagram. Notice that the lines do not need to lie within a plane, but we will not use this property. This constraint does not depend on correspondence, as long as we know the cameras are looking at the same textured plane. We therefore have a constraint on rotation which is independent of exact correspondence.

We can now formulate the first part of our feedback loop. 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, because knowing

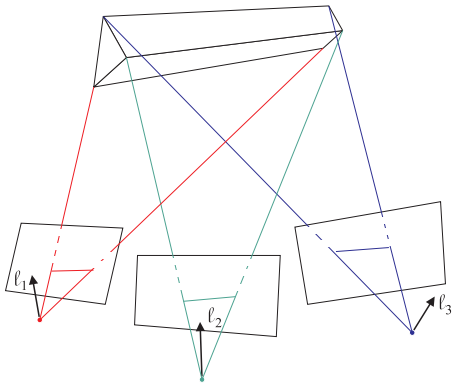


Figure 26: The prismatic line constraint operates on *parallel* lines.

just the rotation provides the orientation of the patch. Another way to think about this, is that knowing the rotation allows us to perform signal processing on the object's surface, so that although our measurements are biased, they are biased the same in both cameras. With this feedback loop we can overcome the small correspondence problem. We now look at the large correspondence problem and the constraints which can solve it.

5.5 Textured plane constraints and the large correspondence problem

In this section we introduce three new linear constraints on our textured plane object. We show how these constraints can be put into a feedback mechanism which can find correspondence if it is possible, and also tell us when it is impossible.

5.5.1 Quintilinear Constraint

The constraint developed in this section is formed on a singly textured plane with five cameras. The constraint is symmetric to all five cameras, but can be thought of as the constraint resulting from the transfer of one line in the singly textured plane reconstructed by four cameras to a fifth camera. See Fig. 27 for a diagram.

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:

$$\boldsymbol{\ell}_5 = \mathbf{L}_m + n_5 \mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d \tag{27}$$

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

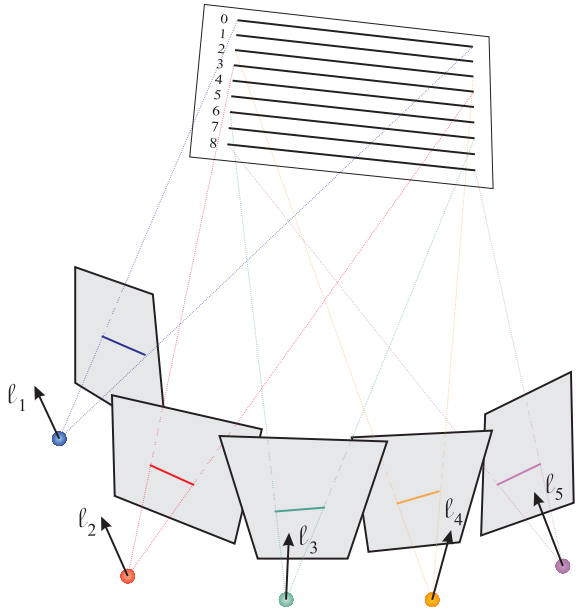


Figure 27: The Quintilinear Constraint operates on five image lines

only one equation in our constraint, and we use $\mathbf{p}_5 = \mathbf{L}_d \times \boldsymbol{\ell}_5$. We can derive

$$0 = (\mathbf{L}_d \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\mathbf{L}_m + n_5\mathbf{L}_\lambda - \mathbf{T}_5 \times \mathbf{L}_d) \tag{28}$$

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

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

$$\begin{aligned} = \sum_{[i_1..i_4] \in \mathbf{P}^+(1..4)} & [2n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + 2n_{i_1}n_5(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_1})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})^{\mathrm{T}}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})\mathbf{T}_5^{\mathrm{T}}\boldsymbol{\ell}_5 \end{aligned} \tag{30}$$

which we can expand to

$$\begin{aligned} = \sum_{[i_1..i_4] \in \mathbf{P}^+[1..4]} & [n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_{i_2}n_{i_1}(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_1})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_5)\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_5n_{i_1}(\boldsymbol{\ell}_5 \times \boldsymbol{\ell}_{i_1})^{\mathrm{T}}(\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_{i_1}n_5(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_5)^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_2})\mathbf{T}_{i_4}^{\mathrm{T}}\boldsymbol{\ell}_{i_4} \\ & + n_{i_1}n_{i_2}(\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}}(\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4})\mathbf{T}_5^{\mathrm{T}}\boldsymbol{\ell}_5] \end{aligned} \tag{31}$$

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:

$$0 = \sum_{[i_1..i_5]\in \mathrm{P}^+(1..5)} n_{i_1} n_{i_2} (\boldsymbol{\ell}_{i_1} \times \boldsymbol{\ell}_{i_2})^{\mathrm{T}} (\boldsymbol{\ell}_{i_3} \times \boldsymbol{\ell}_{i_4}) \mathbf{T}_{i_5}^{\mathrm{T}} \boldsymbol{\ell}_{i_5} \tag{32}$$

5.5.2 Incidence of textured plane

To derive the final two constraints, we use the following incidence condition between two textured planes. From this you can guess that our next two constraints consider not a singly textured plane but a single world plane with two sets of parallel lines in it. 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 use the first of these incidence conditions for the octilinear constraint and the final two for the hexalinear constraint.

5.5.3 The hexalinear 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):

$$0 = \sum_{[i_1..i_4]\in \mathrm{P}^+[1..4]} n_{i_1} |\boldsymbol{\ell}_5 \; \boldsymbol{\ell}_6 \; \boldsymbol{\ell}_{i_1}| |\boldsymbol{\ell}_{i_2} \times \boldsymbol{\ell}_{i_3}| \mathbf{T}_{i_4} \boldsymbol{\ell}_{i_4} \tag{33}$$

Notice in particular in this equation that n_5 and n_6 are not used, so that correspondence is irrelevant for the second textured plane. We need to know correspondence for one plane but not the other, so that this is a mixed rotational/translational constraint. See Fig. 28 for a diagram.

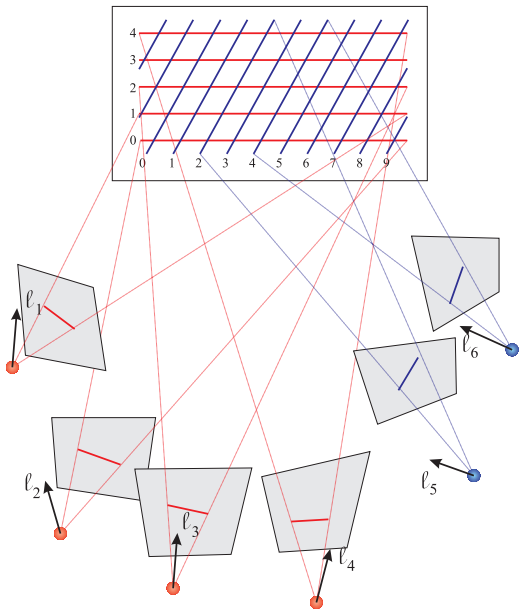


Figure 28: The Hexalinear Constraint operates on six image lines

5.5.4 The octilinear constraint

This constraint uses the textured plane incidence condition that

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

We obtain immediately that the symmetric viewing geometry of a doubly textured plane (octilinear constraint) is:

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

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

$$\mathrm{P}^+[1..4]\mathrm{P}^+[5..8] \tag{34}$$

and

$$\mathrm{P}^+[5..8]\mathrm{P}^+[1..4] \tag{35}$$

In this constraint we need to have two textured planes and to know the correspondence for *both* planes, since all the integer indices $n_1 \ldots n_8$ are used. See Fig. 29 for a diagram.

5.5.5 Translational feedback

In this section we explain a general process by which we may improve upon the translational calibration of a set of cameras which are in patch correspondence. We first argue

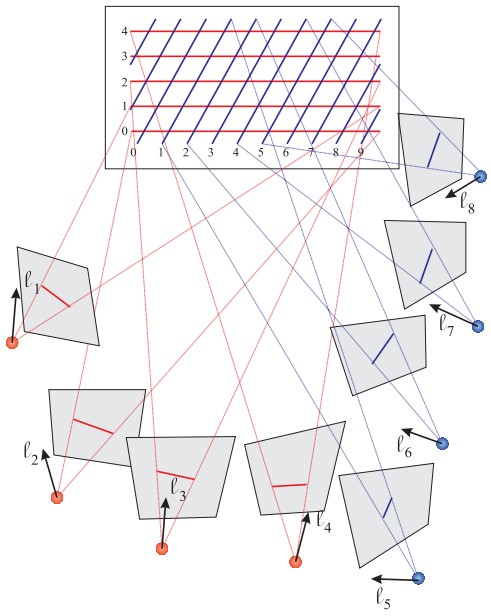


Figure 29: The Octilinear Constraint operates on eight image lines

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 and use it to improve the camera positions.

Table 2: Concepts associated with small versus large correspondence

Small Correspondence	*Large Correspondence*
Prismatic Constraint	Quintilinear and Octilinear Constraints
Hexilinear Constraint	
Shape (slant and tilt)	Structure (depth)
Rotation	Translation
Patch Correspondence	Line Index Correspondence

5.6 The Feedback Loop

5.6.1 Overview of patch results

Let us go over the results in this section to start to put it together into a coherent theory. First of all, we have divided the feature matching problem on images into the small and large correspondence problems. We can see that different types of correspondence and scene properties belong to these two problems, so this separation makes sense theoretically. Our prismatic line constraint needs only patch correspondence, while our quintilinear and octilinear constraints need known integer indices. Our hexilinear is a mixed constraint and this needs patch correspondence for one texture and integer index correspondence for the other.

Further, we have seen that the small correspondence problem can be analyzed independent of translation, while the large correspondence problem is inherently a translational problem. This comes from the basic fact that rotation is associated with local shape (surface normal), while translation is associated with global shape (depth). See Table 2 for a synopsis.

5.6.2 Putting it all together

This section describes how a vision system might operate, using these constraints to remove the bias shown in the previous sections of this paper, and also to overcome the correspondence problem which has plagued vision systems for two decades. We first assume that the vision system has some sort of rotational inertial sensor, like the inner ear of humans. This provides an initial estimate of rotation, and this estimate can be used to overcome the small correspondence problem, because we can find local shape and compute our measurements on the object surface. With these measurements, some initial estimate of translation obtained from egomotion algorithms, and patch correspondences, we can compute integer index correspondences using our multilinear constraints. We can only compute these correspondences in places where we have long enough wavelengths. These correspondences should allow us to improve our translational calibration, which then will allow us to use smaller wavelengths for calibration. Two or three iterations of this should be enough to obtain very accurate camera positions.

We finish with a summary of the feedback process. 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. The answer is textures, specifically harmonic components of them. That is equivalent to parallel lines and the constraints we introduced are the appropriate tools.

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 theorems relating the statistics of texture and the uncertainty in the viewing geometry to the ability to perform a match between textured patches [3]. 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 [15]. 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. 30). Observing then nonrigid 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 [2].

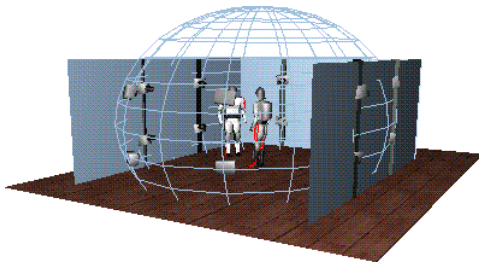


Figure 30:

References

[1] `http://www.akita-u.ac.jp/~kmori/img/kitaoka.html`.

[2] Y. Aloimonos and C. Fermüller. Analyzing action representations. In G. Sommer and Y. Y. Zeevi, editors, *Algebraic Frames for the Perception-Action Cycle*. Springer-Verlag, 2000.

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

[4] 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.

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

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

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

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

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

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

[11] C. Fermüller and Y. Aloimonos. Qualitative egomotion. *International Journal of Computer Vision*, 15:7–29, 1995.

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

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

[14] C. Fermüller and Y. Aloimonos. Statistics explains geometric optical illusions. In L. S. Davis, editor, *Foundations of Image Understanding*, pages 409–446. Kluwer Academic Publishers, Boston, 2001.

[15] C. Fermüller, T. Brodský, and Y. Aloimonos. Motion segmentation: A synergistic approach. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 226–231A, 1999.

[16] C. Fermüller, L. Cheong, and Y. Aloimonos. Visual space distortion. *Biological Cybernetics*, 77:323–337, 1997.

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

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

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

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

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

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

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

[24] `http://www.illusionworks.com`.

[25] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

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

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