E

Edge Detection

James H. Elde
Centre for Vision Research, York University, Toronto, ON, Canada

Related Concepts

►Boun-dary Detection; ►Scale Selection

Definition

Edge detection is the process of label the image pixels that lie on the boundaries where abrupt intensity discontinuity occur.

Background

The light projected from a visual scene into an eye or camera is typically piecewise smooth as a function of visual angle. Since nearby points on a surface tend to have similar attitude, reflectance, and illumination, the pixels to which these surface points project tend to have similar intensity. This rule is broken when two adjacent pixels project from points on either side of an occlusion boundary, since the points now project from different surfaces that may well have different attitude, reflectance, and illumination, and typically an abrupt change in image intensity results. Intensity edges also arise when neighboring pixels project from points on the same surface that happens to straddle a surface crease, pigment change, or shadow boundary.

Since these abrupt changes in image intensity correspond to significant physical events in the scene, the problem of reliably detecting and localizing these edges is an important and fundamental early vision problem. While detection of object boundaries (occlusion edges) is sometimes seen as the main goal, reliable detection of surface creases and reflectance edges also has clear importance for shape estimation and object recognition, and even cast shadows provide important information about relief, surface contact, and scene layout.

Since edges are sparse, edge detection is also motivated from a differential image coding viewpoint: a large fraction of the information in an image can be captured by coding just the locations, 2D orientations, and intensity changes of these edges.

The problem of edge detection dates back to the first days of computer vision: back to Roberts' thesis at least [20]. In its simplest form, the goal is to label the image pixels that lie on (or very near) a step discontinuity in the image. This definition has been generalized in a number of useful ways: to allow for possible blurring of the step discontinuity, due to shading or defocus, for example, and to require an estimate of the local 2D orientation of the edge as well as its location.

– Biological basis [9]
– History [20]
– Problem definition [1, 2, 5, 6, 13, 15, 18]

Theory and Application

Roberts based his edge detector on a simple 2 pixel × 2 pixel discrete differential operator related to the gradient magnitude (Fig. 1). Since the operator relies

K. Ikeuchi (ed.), *Computer Vision*, DOI 10.1007/978-0-387-31439-6,

Edge Detection, Fig. 1 Edge detection example [4–6]. *Top left*: Original greyscale image. *Top right*: Edge map. *Bottom left*: Reconstruction of original image from brightness and contrast stored only at edge locations. *Bottom right*: Reconstruction including edge blur information

upon only 4 pixels, the results are highly sensitive to noise and are dominated by the fine structure of the image. Over the intervening years, most edge detection algorithms have continued to rely on a first stage of local differential filtering but have innovated in the design of smoothing kernels to increase signal-to-noise ratio, in the order of the differential operators used, and in how they are combined. This local differential filtering approach aligns well with physiological data showing that neurons early in the primate visual pathway can to some degree be approximated as stabilized low-order differential operators [9].

By far the most popular smoothing kernel has been the 2D Gaussian $G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$. Marr and Hildreth [17] proposed the use of second-order isotropic Laplacian-of-Gaussian ("Mexican hat") filters $\nabla^2 G(x, y)$. The $\nabla^2 G$ filter produces a signed response that crosses zero precisely at the location of an (ideal) step edge, and Marr and Hildreth proposed that edges thus be identified with such zero crossings. Due to the isotropy of the $\nabla^2 G$ filter, the response is invariant to the orientation of the edge, a desirable property, since edges can occur at any orientation. Marr and Hildreth also observed that different edges occur at different scales, and so employed $\nabla^2 G$ filters over a range of scales σ. Observing that spurious zero crossings at a single scale can occur due to interference between multiple distinct edges, Marr and Hildreth proposed a *scale combination rule*: edges are deemed valid only if zero crossings at the same location and orientation are found at more than one scale.

Marr and Hildreth obtained orientation invariance by using isotropic filters and approximate scale invariance by combining information across scales. However, these invariance properties came at the price detection and localization performance. This became clear partly through the work of John Canny [2], who used a variational approach to determine an edge detector that would be (nearly) optimal precisely in terms of detection rate and localization performance, while avoiding multiple responses to the same edge. The outcome was a an edge detection filter that is well approximated by a first-derivative-of-Gaussian function. Canny used two such filters to estimate stabilized partial derivatives and hence the local gradient vector ∇G:

$$\nabla G = \begin{pmatrix} G_x \\ G_y \end{pmatrix}, \quad \text{where } G_x = \frac{\partial G}{\partial x} \text{ and } G_y = \frac{\partial G}{\partial y}$$

The first-derivative-of-Gaussian filter can be shown to be steerable [7]: the filter can be synthesized at any orientation from a linear combination of the two basis filters G_x and G_y. Thus, Canny's approach amounts to computing the first derivative in the gradient direction at every pixel of the smoothed image. In this sense, the approach still enjoys the orientation invariance property emphasized by Marr and Hildreth: a rotation of the edge in the image will not change the estimated gradient magnitude. Canny's approach, however, delivers higher signal-to-noise because the filter more closely approximates the shape of an extended edge in the tangent direction. While Marr and Hildreth's $\nabla^2 G$

operator computes the second derivative along the edge, Canny's ∇G operator locally integrates along the edge over the support of the Gaussian kernel.

While Marr and Hildreth localized edges at the zero crossings of the $\nabla^2 G$ response, Canny localized edges at maxima of the gradient magnitude, taken in the gradient direction, found using a process called *non-maximum suppression*. Since not all of the resulting maxima correspond to significant edges, a threshold on the gradient magnitude must be applied to reduce the false positive rate. Canny devised a clever heuristic technique, dubbed *thresholding with hysteresis*, that removes many false positives without unduly affecting the hit rate (proportion of correct detections). The technique exploits the fact that real weak edges are often chain-connected along a contour to stronger edges, while false positives are more likely to be isolated. The technique employs two thresholds: all edges that are above the low threshold and chain-connected to edges above the high threshold are identified as edges.

The Canny edge detector may well be the most widely used algorithm in the history of computer vision. Its early adoption is likely derived in part from the open availability of the source code, but it's continuing widespread use reflects the fact that it continues to perform well in comparison to more recent algorithms.

While most versions of Canny's algorithm in use detect edges only at a single scale, in fact both Marr and Hildreth and Canny recognized the problem of scale, but dealt with it in slightly different ways. While Marr and Hildreth conjunctively combined responses across scale (zero crossings of the $\nabla^2 G$ response must be found at multiple scales at the same location and orientation to signal an edge), Canny proposed to *disjunctively* combine edges detected at different scales. This raises the *multiple response problem*: how do we know whether two extremal responses, at different scales but similar locations and orientations, signal two distinct edges in the image, or a single edge corrupted by noise? Canny proposed a *feature synthesis* method to deal with this problem, that signals the larger scale edge only if it could not be satisfactorily explained by the smaller scale responses in a local neighborhood.

Canny's proposed technique localizes edges using the smallest scale at which they are detected, an approach that Canny justified based upon his variational optimization, and this motivated the later development of *edge focusing* techniques [1] in which edges are detected at coarse scales and then tracked through scale space to finer scales for better localization. As it turns out, Canny's one-dimensional analysis does not generalize to two dimensions: Elder and Zucker [5, 6] later showed that for an ideal step or blurred step edge corrupted by noise, localization improves monotonically with increasing scale. This result highlights the difference between theory and practice: since edges in real images are always finite in extent, often curved, and surrounded by other kinds of image structure, neither detection nor localization is likely to be optimized by maximizing scale.

Elder and Zucker pointed out that while it is difficult to model all of the real-world effects that limit the effectiveness of large filters, the small-scale problem is tractable, since performance is limited by noise that can be modeled as white, and for which parameters can be estimated. Based on this observation, they proposed a method for selecting the scale of differential operators called *local scale control*, in which scales are search from fine to coarse to determine the *minimum reliable scale*, that is the smallest scale at which the sign of the derivative measurement is statistically reliable. This approach avoids spurious responses to noise, while minimizing bias due to the finite extent of the edge, curvature, and interference from neighbouring image structure. They also demonstrated methods for subpixel localization down to roughly $1/20$ of a pixel precision [5].

A different scale selection approach for edge detection was proposed at roughly the same time by Lindeberg [15]. Lindeberg's method selects the scale maximizing the response of so-called γ-normalized differential operators. This approach has the property that the scales selected for an ideal, noise-free, isolated, blurred straight edge of infinite extent will be proportional to the blur scale of the edge, and independent of the edge contrast. Thus, small scales should be selected for sharp edges, and larger scales for blurred edges. Unlike the local scale control approach of Elder and Zucker, the selection of scale is not related to the signal-to-noise ratio: in general a large-scale filter will have a smaller response to an ideal edge than a small-scale filter, even when the signal-to-noise for the two filters is the same. As a result of this bias to smaller scales, many noisy edges are detected, and these must be eliminated using some form of post hoc thresholding.

These studies also raised issues around the order of differential operators to employ. While Canny

localized edges at gradient maxima, Elder and Zucker localized edges at zero crossings of the second derivative, steered in the gradient direction. Specifically edges are localized at the boundary between two pixels where the sign of the second derivative changes from positive to negative in the gradient direction (this avoids detections of minima in the gradient magnitude). Where the same scales used for these two operators, the results would be identical. However, the Elder and Zucker approach tunes the scales of the two operators independently, based upon the signal-to-noise properties of the operators, and as a result the gradient maxima and second derivative zero crossings are decoupled. While the statistical testing method cannot be used to distinguish real and spurious maxima, it can be used to test for response sign, and hence to detect zero crossings. Thus, the use of the second derivative here is critical. Lindeberg [15] has argued for explicit calculation and testing of the sign of the third derivative, but this is equivalent to checking the sign of the second derivative on either side of the zero crossing. However, Elder and Zucker do show that explicit computation of the third derivative is useful for estimating the blur of an edge, which may be useful for discriminating different types of edges, and for recovering depth from defocus.

In their 1980s paper [17], Marr and Hildreth speculated about the possibility that the locations and gradient magnitudes at oriented zero crossings over multiple scales might constitute a complete encoding of the image, allowing, in principle, for the image to be perfectly reconstructed from the edge representation. While this may seem to be going in the wrong direction from a computer vision point of view, the question is important because it addresses whether an edge code could serve as a *complete* early visual representation, providing sufficient information for all higher-level algorithms.

Since Marr and Hildreth's original conjecture, there have been numerous theoretical and empirical studies exploring the possible completeness of an edge code. Mallat and Zhong [16] demonstrated excellent reconstruction results based upon such a code. However, since edges are represented at many scales, and at the finest scales edge density is very high, this code is highly overcomplete. More compact codes can be derived, but at the expense of noticeable artifact in the reconstruction.

Elder [4] explored an alternative reconstruction approach based upon their edge representation. Rather than storing information from all scales at each edge point, only the location, 2-bit orientation, contrast, brightness, and blur of each edge point were stored, resulting in a far more compact code. Reconstruction is excellent as long as both the intensity and the blur information are employed in the reconstruction.

Since edge detection is often just the first computational step in a computer vision pipeline, and applications may have real-time requirements, efficiency has always been a factor in the design of edge detection algorithms. Both Marr and Hildreth [17] and Canny [2] deliberately employed filters that were $x-y$ separable, allowing 2D convolutions to be computed by composing much cheaper 1D convolutions. Rachid Deriche [3] further improved upon these efficiency by developing recursive filters based upon Canny's original design criteria, allowing very fast implementation on sequential hardware. More recent algorithms have tended to rely upon steerable filters [7] that allow oriented operators to be implemented using only a few basis functions.

While linear filtering forms the front end of most edge detection algorithms, a number of interesting non-linear techniques have been studied. In fact this has a long tradition, going back to early work that sees the problem as model fitting [10] and includes active contour ("snake") methods for fitting semi-local deformable contour models to visual data [12]. While effective in many applications, active contour methods tend to have more parameters to tune, and results are sensitive to these and to initial conditions.

Still to include:

- Scale [13]
- Color edge detection [14]
- Nonlinear filtering [11]
- Nonlocal methods [19]
- Evaluation [8, 13, 18]
- Redefinition in terms of "salient" boundaries [13, 18]

References

1. Bergholm F (1987) Edge focusing. IEEE Trans Pattern Anal Mach Intell 9(6):726–741
2. Canny J (1986) A computational approach to edge-detection. IEEE Trans Pattern Anal Mach Intell 8(6): 679–698
3. Deriche R (1987) Using Canny's criteria to derive a recursively implemented optimal edge detector. Int J Comput Vis 1(2):167–187

4. Elder JH (1999) Are edges incomplete? Int J Comput Vis 34(2–3):97–122
5. Elder JH, Zucker SW (1996) Scale space localization, blur and contour-based image coding. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). IEEE Computer Society, Los Alamitos, pp 27–34
6. Elder JH, Zucker SW (1998) Local scale control for edge detection and blur estimation. IEEE Trans Pattern Anal Mach Intell 20(7):699–716
7. Freeman WT, Adelson EH (1991) The design and use of steerable filters. IEEE Trans Pattern Anal Mach Intell 13(9):891–906
8. Heath M, Sarkar S, Sanocki T, Bowyer K (1998) Comparison of edge detectors – a methodology and initial study. Comput Vis Image Underst 69(1):38–54
9. Hubel DH, Wiesel TN (1968) Receptive fields and functional architecture of monkey striate cortex. J Physiol 195:215–243
10. Hueckel MH (1971) An operator which locates edges in digitized pictures. J Assoc Comput Mach 18:113–125
11. Iverson LA, Zucker SW (1995) Logical/linear operators for image curves. IEEE Trans Pattern Anal Mach Intell 17(10):982–996
12. Kass M, Witkin A, Terzopoulos D (1987) Snakes – active contour models. Int J Comput Vis 1(4):321–331
13. Konishi S, Yuille AL, Coughlan JM, Zhu SC (2003) Statistical edge detection: learning and evaluating edge cues. IEEE Trans Pattern Anal Mach Intell 25(1):57–74
14. Lee HC, Cok DR (1991) Detecting boundaries in a vector field. IEEE Trans Signal Process 39(5):1181–1194
15. Lindeberg T (1998) Edge detection and ridge detection with automatic scale selection. Int J Comput Vis 30(2):117–154
16. Mallat S, Zhong S (1992) Characterization of signals from multiscale edges. IEEE Trans Pattern Anal Mach Intell 14(7):710–732
17. Marr D, Hildreth E (1980) Theory of edge-detection. Proc R Soc Lond B 207(1167):187–217
18. Martin DR, Fowlkes CC, Malik J (2004) Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Trans Pattern Anal Mach Intell 26(5):530–549
19. Perona P, Malik J (1990) Scale-space and edge-detection using anisotropic diffusion. IEEE Trans Pattern Anal Mach Intell 12(7):629–639
20. Roberts L (1965) Machine perception of 3-dimensional solids. In: Tippett J (ed) Optical and electro-optical information processing. MIT, Cambridge, MA

Eigenspace Methods

Tomokazu Takahashi and Hiroshi Murase
Faculty of Economics and Information, Gifu Shotoku Gakuen University, Gifu, Japan

Synonyms

Methods of image recognition in a low-dimensional eigenspace

Related Concepts

▸Principal Component Analysis (PCA)

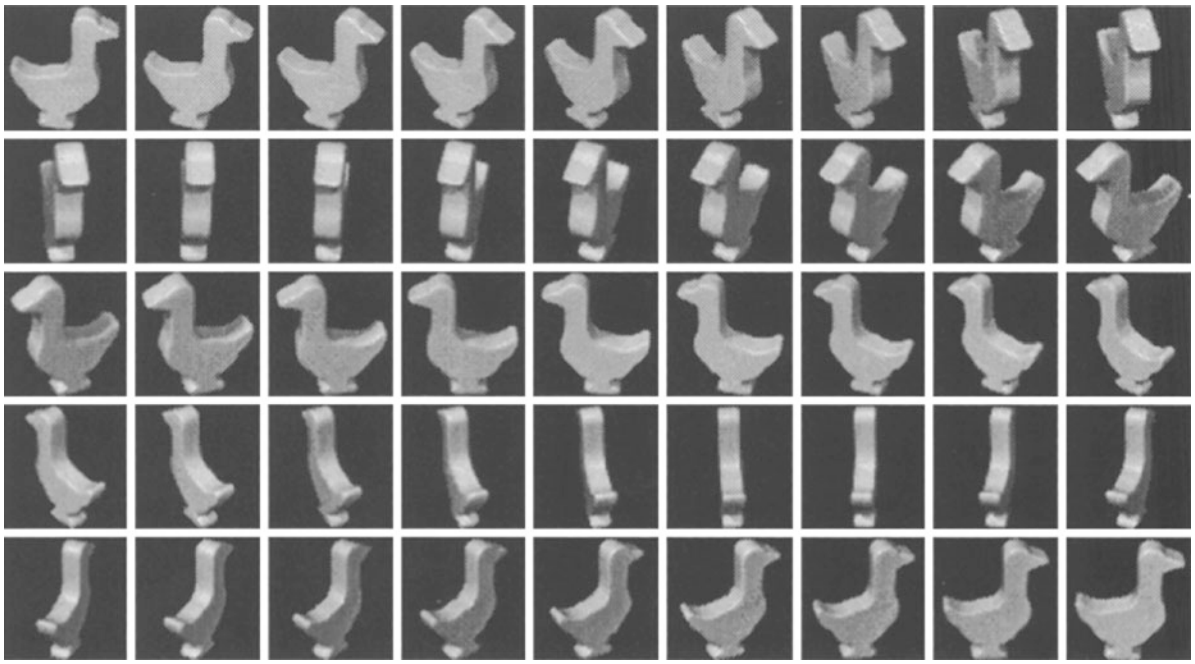
Definition

The eigenspace method is an image recognition technique that achieves object recognition, object detection, and parameter estimation from images using the distances between input and gallery images in a low-dimensional eigenspace. Here, the eigenspace is constructed based on a statistical method, such as principal component analysis or Karhunen-Loève transform, so that the variation in the appearances of target objects can be represented in a low-dimensional space efficiently. In particular, a technique called the parametric eigenspace method represents the rotation and translation of a target object or a light source as a manifold in an eigenspace. Accordingly, this method performs object recognition and parameter estimation using distances in the manifold.

Background

Appearance-based object recognition is a technique that recognizes a target object by matching between input and preregistered gallery images. One of the simplest methods to achieve this image matching calculates the distances between the pixel values of these images. However, it is difficult for many applications to apply this method due to two problems: (1) processing time needed to calculate the distance between images increases depending on the size of the images and (2) memory space needed to store the gallery images of target objects grows depending on the number of the objects.

Principal component analysis (PCA) can be used as one of the unsupervised dimensionality reduction techniques that transform a sample set of high-dimensional vectors to the set of low-dimensional vectors with minimum information loss. This technique first calculates eigenvectors that are at right angles to each other and maximizes the variances in their directions; it then constructs a low-dimensional eigenspace defined by a small number of eigenvectors. The low-dimensional vectors are obtained by projecting the high-dimensional vectors to the eigenspace. An eigenspace method constructs a low-dimensional



Eigenspace Methods, Fig. 1 Image samples in an object image set containing object images with different horizontal poses

eigenspace from preregistered gallery images of target objects and calculates the distances between input and gallery images in the low-dimensional space. Thus, the eigenspace method can reduce the processing time and memory space efficiently without degrading the recognition performance.

Theory

An image recognition procedure using an eigenspace method is divided into learning and recognition stages. The learning stage, which is performed beforehand, constructs a low-dimensional eigenspace from a large number of learning images of target objects and then projects each learning image to the eigenspace. On the other hand, the recognition stage projects an input image of a target object to the eigenspace and then recognizes the input object by matching between input and learning images in the low-dimensional space.

Learning Stage

For each learning image, the method first normalizes the image size so that the number of pixels can be N, and it represents them as N-dimensional vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$. M represents the number of learning images. Here, we assume that each target object has one learning image; therefore, M also represents the number of target objects. A matrix $\boldsymbol{X}$ is constructed as follows:

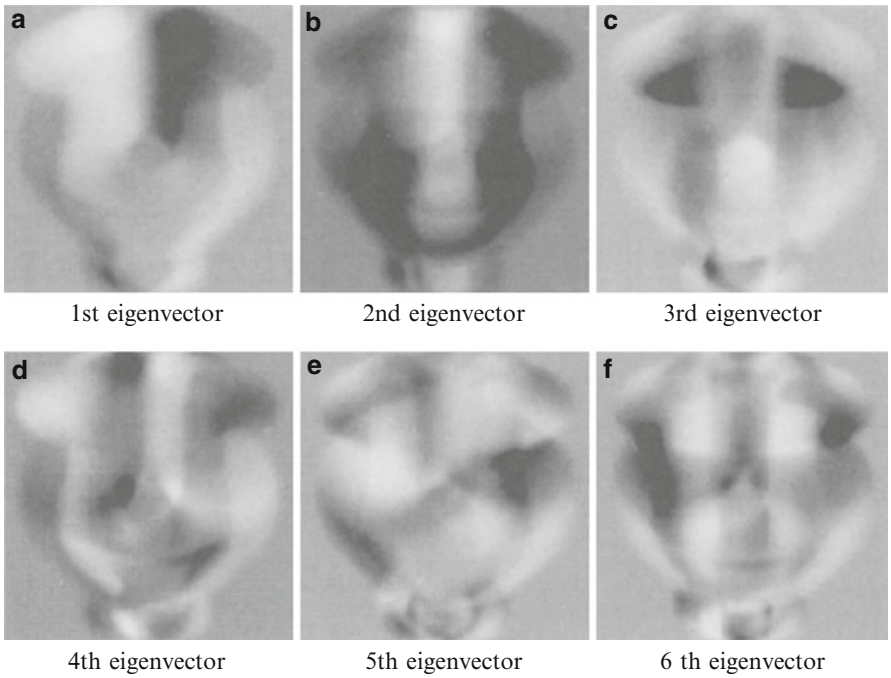
$$\boldsymbol{X} = \left[\frac{\boldsymbol{x}_1 - \boldsymbol{c}}{||\boldsymbol{x}_1 - \boldsymbol{c}||} \cdots \frac{\boldsymbol{x}_M - \boldsymbol{c}}{||\boldsymbol{x}_M - \boldsymbol{c}||} \right], \tag{1}$$

$$\boldsymbol{c} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{x}_m. \tag{2}$$

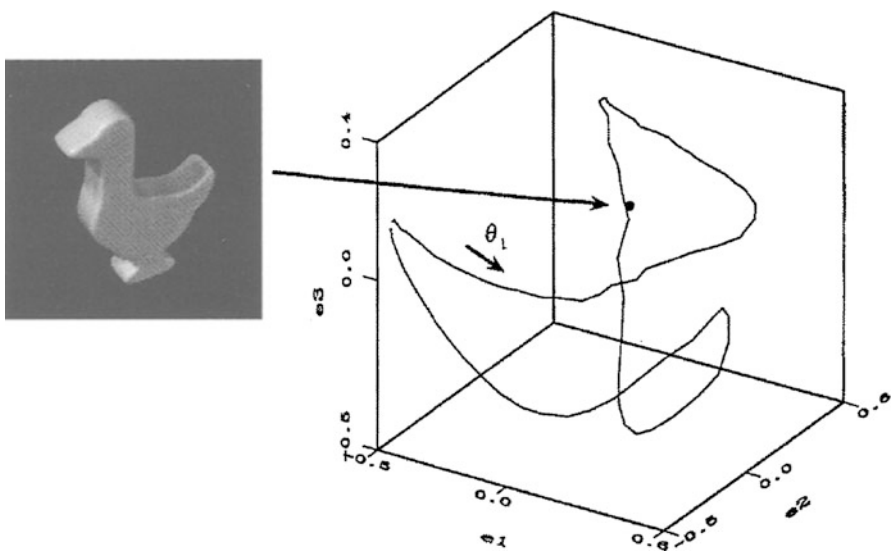
Eigenvectors are obtained by solving the following eigenequation:

$$\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i, \tag{3}$$

where $\boldsymbol{u}_i$ represents an eigenvector of $\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$ corresponding to an eigenvalue λ_i. Eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ are arranged in descending order of their eigenvalues. The method constructs a low-dimensional eigenspace from $k(<< N)$ eigenvectors corresponding to the k largest eigenvalues and then obtains k-dimensional vectors $\boldsymbol{f_m}$ by projecting N-dimensional vectors $\boldsymbol{x}_m$ to the eigenspace using the following equation:



Eigenspace Methods, Fig. 2 Eigenvectors calculated from the object image set shown in Fig. 1. (**a**)–(**f**) represent the first through sixth eigenvectors, respectively



Eigenspace Methods, Fig. 3 A manifold in the object eigenspace constructed from the eigenvectors shown in Fig. 2. Transition of object appearances according to the horizontal pose parameter θ_1 draws a closed smooth curve in the eigenspace

$$\boldsymbol{f_m} = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_k]^{\mathrm{T}} \frac{\boldsymbol{x}_m - \boldsymbol{c}}{||\boldsymbol{x}_m - \boldsymbol{c}||}. \quad (4)$$

Consequently, the memory space needed to store the learning images can be greatly reduced.

Recognition Stage

In the same manner as in the learning stage, the method first normalizes the image size of an input image and represents it as an N-dimensional vector $\boldsymbol{y}$. The method then obtains a k-dimensional vector $\boldsymbol{g}$ by projecting $\boldsymbol{y}$ to the low-dimensional eigenspace

constructed in the learning stage using the following equation:

$$\boldsymbol{g} = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_k]^{\mathrm{T}} \frac{\boldsymbol{y} - \boldsymbol{c}}{||\boldsymbol{y} - \boldsymbol{c}||}. \quad (5)$$

The method recognizes the input object as an object $\hat{m}$ that minimizes the distance between the input image $\boldsymbol{g}$ and the learning image $\boldsymbol{f_m}$ using the following equation:

$$\hat{m} = \arg\min_m ||\boldsymbol{g} - \boldsymbol{f_m}||^2. \quad (6)$$

Accordingly, the processing time needed to calculate the distance between the images can be greatly reduced.

Based on an efficient representation of human face images using PCA as proposed by Sirovich and Kirby [1], Turk and Pentland [2] proposed a method using "eigenfaces" to detect and recognize human faces in images. This method calculates a small number of eigenvectors from a large number of learning face images and measures the distances between input and learning face images in a low-dimensional eigenspace. The eigenvectors calculated from the face images are called eigenfaces. Since this method was reported, image recognition using the eigenspace method has become an active area of research.

Murase and Nayar [3] proposed a "parametric eigenspace method" for recognizing three-dimensional objects and estimating their poses simultaneously. This method constructs a manifold as a smooth curved line approximated from a point set in an eigenspace for each target object. The point set is obtained by projecting learning images with various poses to the eigenspace. Object recognition and pose estimation are achieved by calculating the distances between an input image and manifolds. In order to achieve object recognition and pose estimation simultaneously, two types of eigenspaces are constructed: a universal eigenspace from the learning images of all target objects and an object eigenspace for each target object. The method first recognizes an object in the universal eigenspace and then estimates a pose of the object in the object eigenspace. Figure 1 shows samples in an object image set containing object images with different horizontal poses, and Fig. 2 shows eigenvectors that were calculated from the image set. On the other hand, the closed smooth curve shown in Fig. 3 represents a manifold in the object eigenspace that was constructed from the eigenvectors. The manifold was approximated from the points that were obtained by projecting the image in the image set shown in Fig. 1 to the eigenspace.

Ohba and Ikeuchi [4] proposed a method using "eigen window" to recognize partially occluded objects accurately and estimate their poses. The method extracts multiple local regions with high detectability, uniqueness, and reliability from an object region in each learning image. Eigen windows are constructed from the extracted local regions by PCA. Object recognition and pose estimation are achieved by matching between the eigen windows and local regions extracted from an input image in a similar way.

In addition to the approaches described above, there have been a number of techniques related to the eigenspace method. These include techniques of density estimation of samples in a high-dimensional space [5], nonlinear expansion of PCA [6, 7], image recognition using two-dimensional PCA [8], and image recognition using high-order tensors [9].

Application

The eigenspace method is used as a fundamental technique in arbitrary computer vision applications, such as object recognition, detection, and tracking, because the method works effectively despite its algorithm being quite simple.

References

1. Sirovich L, Kirby M (1987) Low-dimensional procedure for the characterization of human faces. J Opt Soc Am A 4(3):519–524
2. Turk M, Pentland A (1991) Eigenfaces for recognition. J Cognit Neurosci 3(1):71–86
3. Murase H, Nayar S (1995) Visual learning and recognition of 3-d objects from appearance. Int J Comput Vis 14:5–24
4. Ohba K, Ikeuchi K (1997) Detectability, uniqueness, and reliability of eigen window for stable verification of partially occluded objects. IEEE Trans Pattern Anal Mach Intell 19(9):1043–1048
5. Moghaddam B, Pentland A (1997) Probabilistic visual learning for object representation. IEEE Trans Pattern Anal Mach Intell 19(7):696–710
6. Schölkopf B, Smola A, Müller K (2006) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10(5):1299–1319

7. Belkin M, Niyogi P (2006) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396
8. Yang J, Zhang D, Frangi A, Yang J (2004) Two-dimensional pca: a new approach to appearance-based face representation and recognition. IEEE Trans Pattern Anal Mach Intell 26(1):131–137
9. Vasilescu M, Terzopoulos D (2002) Multilinear analysis of image ensembles: tensorfaces. Proc Eur Conf Comput Vis (ECCV) 1:447–460

Eight-Point Algorithm

Zhengyou Zhang
Microsoft Research, Redmond, WA, USA

Related Concepts

▸Eight-Point Algorithm; ▸Epipolar Geometry; ▸Essential Matrix; ▸Fundamental Matrix

Definition

The 8-point algorithm is a linear technique to estimate the essential matrix or the fundamental matrix from eight or more point correspondences.

Background

When dealing with multiple images, it is essential to determine the relative geometry between them, which is known as the epipolar geometry. Between two images of a scene, given a pair of corresponding image points $(\mathbf{m}_i, \mathbf{m}'_i)$, the following epipolar constraint must be satisfied:

$$\widetilde{\mathbf{m}}_i'^T \mathbf{M} \widetilde{\mathbf{m}}_i = 0 \,, \tag{1}$$

where $\widetilde{\mathbf{m}}_i = \begin{bmatrix} \mathbf{m}_i \\ 1 \end{bmatrix}$ is point $\mathbf{m}_i$ in homogeneous coordinates. Similarly, $\widetilde{\mathbf{m}}'_i$ is point $\mathbf{m}'_i$ in homogeneous coordinates. Matrix $\mathbf{M}$ is a 3×3 matrix. If the images are calibrated with known intrinsic camera parameters and the image points are expressed in the normalized image coordinate system, then matrix $\mathbf{M}$ is known as the *essential matrix*, and is usually denoted by $\mathbf{E}$; otherwise, matrix $\mathbf{M}$ is known as the *fundamental matrix* and is usually denoted by $\mathbf{F}$. Both essential matrix and fundamental matrix must satisfy certain properties, but the common property is that it is a rank-2 matrix, i.e., the determinant of matrix $\mathbf{M}$ is equal to zero.

Theory

The 8-point algorithm ignores the constraints on the elements of matrix $\mathbf{M}$ and treats them independently. Let us define a 9-D vector $\mathbf{x}$ using the elements of matrix $\mathbf{M}$ such that

$$\mathbf{x} = [M_{11}, M_{21}, M_{31}, M_{12}, M_{22}, M_{32}, M_{13}, M_{22}, M_{33}]^T \,, \tag{2}$$

where M_{ij} is the (i, j) element of matrix $\mathbf{M}$. Let $\mathbf{m} = [u, v]^T$. Then, the epipolar constraint (Eq. 1) can be rewritten as

$$\mathbf{a}_i^T \mathbf{x} = 0 \,, \tag{3}$$

where

$$\mathbf{a}_i = [u_i \widetilde{\mathbf{m}}_i'^T, v_i \widetilde{\mathbf{m}}_i'^T, \widetilde{\mathbf{m}}_i'^T]^T \,. \tag{4}$$

It is clear that $\mathbf{x}$ can only be determined up to a scale factor.

Given eight or more point correspondences, $\mathbf{x}$ can be determined as follows. Each point correspondence yields one epipolar equation (Eq. 1). With N $(N \geq 8)$ point correspondences, we can stack them together into a vector equation as follows:

$$\mathbf{A}^T \mathbf{x} = \mathbf{0} \,, \tag{5}$$

with $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_N]$. The solution to $\mathbf{x}$ can be obtained by minimizing the least squares, subject to $\|\mathbf{x}\| = 1$. With Lagrange multiplier, this is equivalent to minimizing

$$\|\mathbf{A}^T \mathbf{x}\|^2 + \lambda(1 - \|\mathbf{x}\|^2) \,. \tag{6}$$

With simple algebra, it can be found that the solution to $\mathbf{x}$ is the eigenvector of the 9×9 matrix $\mathbf{A}\mathbf{A}^T$ associated with the smallest eigenvalue.

When the image coordinates (u_i, v_i) are in pixels, the elements of matrix $\mathbf{A}$ have orders of difference in value, and matrix $\mathbf{A}\mathbf{A}^T$ may not be well conditioned. One remedy is to pre-normalize the image

points, and several solutions are examined in [1]. One simplest approach is to perform a scaling and translation such that all image coordinates are within $[-1, 1]$. Compared with using directly the pixel coordinates, significant improvement in accuracy has been observed.

The matrix $\mathbf{M}$ estimated above is obtained by ignoring its property. For example, the estimated $\mathbf{M}$ is usually not rank-2. To obtain the closest rank-2 matrix, "closest" in terms of Fronobius norm, we perform a singular value decomposition on $\mathbf{M}$, i.e.,

$$\mathbf{M} = \mathbf{USV}\,, \tag{7}$$

where $\mathbf{S} = \text{diag}\,(s_1, s_2, s_3)$ with $s_1 \geq s_2 \geq s_3 \geq 0$. Replacing the smallest singular value by zero, i.e., $\hat{\mathbf{S}} = \text{diag}\,(s_1, s_2, 0)$, then

$$\mathbf{U}\hat{\mathbf{S}}\mathbf{V} \equiv \hat{\mathbf{M}} \tag{8}$$

is the optimal rank-2 matrix.

For more details about the epipolar geometry, the reader is referred to [2, 3]. The reader is referred to [4] for a review of various methods for determining the epipolar geometry (the essential matrix and the fundamental matrix), to [5] for a study of the relationship between various optimization criteria, and to [6] for how to obtain a more robust Euclidean motion and structure estimation via the estimation of fundamental matrix.

References

1. Hartley R (1997) In defense of the eight-point algorithm. IEEE Trans Pattern Anal Mach Intell 19(6):580–593
2. Faugeras O, Luong QT, Papadopoulo T (2001) The geometry of multiple images. MIT, Cambridge
3. Hartley R, Zisserman A (2000) Multiple view geometry in computer vision. Cambridge University Press, Cambridge/New York
4. Zhang Z (1998) Determining the epipolar geometry and its uncertainty: a review. Int J Comput Vis 27(2):161–195
5. Zhang Z (1998) On the optimization criteria used in two-view motion analysis. IEEE Trans Pattern Anal Mach Intell 20(7):717–729
6. Zhang Z (1997) Motion and structure from two perspective views: from essential parameters to euclidean motion via fundamental matrix. J Opt Soc Am A 14(11): 2938–2950

Ellipse Fitting

Zhi-Yong Liu
Institute of Automation, Chinese Academy of Sciences, Beijing, P. R. China

Synonyms

Ellipse matching

Definition

Fit one or more ellipses to a set of image points.

Background

Fitting geometric primitives to image data is a basic task in pattern recognition and computer vision. The fitting allows reduction and simplification of image data to a higher level with certain physical meanings. One of the most important primitive models is ellipse, which, being a projective projection of a circle, is of great importance for a variety of computer vision-related applications.

Ellipse fitting methods can be roughly divided into two categories: least square fitting and voting/clustering. Least square fitting, though usually fast to implement, requires the image data presegmented and is sensitive to outliers. On the other hand, voting techniques can detect multiple ellipses at once and exhibit some robustness against noise but suffer from a heavier computational and memory load. Furthermore, most of standard ellipse-fitting methods cannot be directly used in real-world applications involving a noisy environment. Thus, the arc finding-based techniques will also be described in the entry, with emphasis on ellipse fitting in complicated images, though, strictly speaking, they cannot be taken as a counterpart of the above two categories.

Due to space limitation, there are some other techniques that cannot be covered by the entry but with some references listed in the recommended reading, such as the moment [16] and genetic algorithm [12]-based methods.

Theory

Least Square Fitting

Least square fitting is realized by minimizing some distance measure between ellipse and image data as follows:

$$\mathbf{\Theta} = \arg\min_{\mathbf{\Theta}} \sum_{i=1}^{N} \mathcal{D}(\mathbf{\Theta}, \mathbf{x}_i)^2, \tag{1}$$

where $\mathcal{D}(\mathbf{\Theta}, \mathbf{x}_i)$ denotes some distance between pixel $\mathbf{x}_i = [x_i, y_i]^T$ and the ellipse specified by $\mathbf{\Theta}$.

There are two main distance measures for ellipse fitting: geometric distance and algebraic distance. In efficiency, the algebraic distance-based techniques outperform its counterpart because a direct solution of Eq. (1) is obtainable by using algebraic distance. However, the algebraic fitting suffers from the drawback of bias estimation and unclear physical interpretations on the estimated errors and fitting parameters. For instance, some algebraic fitting results are not invariant to the coordinates transformation of image data.

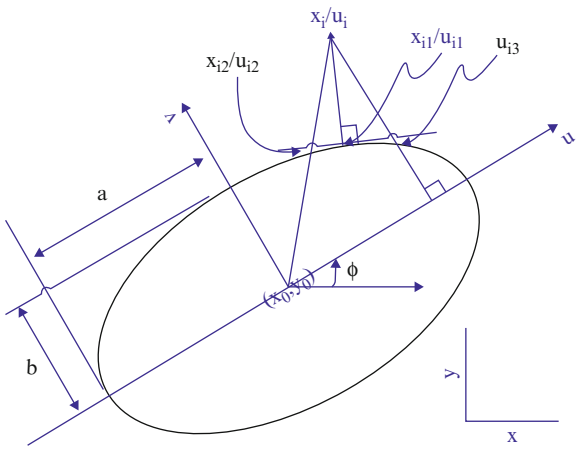
Geometric Fitting

An ellipse takes the following equation:

$$\frac{[(x - x_0)\cos\phi + (y - y_0)\sin\phi]^2}{a^2} + \frac{[(x - x_0)\sin\phi + (y - y_0)\cos\phi]^2}{b^2} = 1, \tag{2}$$

where x_0, y_0 are coordinates of center, ϕ is the orientation, and a, b are the semiaxis, as shown in Fig. 1. Geometric distance, also known as shortest or orthogonal distance, is defined by the distance between one image point and its orthogonal projection upon the ellipse, as illustrated by $D_1(\mathbf{x}_i) = \|\mathbf{x}_i - \mathbf{x}_{i1}\|$ in Fig. 1. In such a setting, the ellipse fitted becomes actually a principal curve, which is best in the sense of mean square reconstruction error minimization. However, the distance D_1 is analytically intractable. There exist some techniques to tackle the problem, unavoidably involving some iterative numerical algorithm.

Ahn et al. [1] use a Gaussian-Newton algorithm for geometric fitting, by implicitly describing the orthogonal projection point $\mathbf{x}_1$. A temporary coordinate system



Ellipse Fitting, Fig. 1 An ellipse with center at (x_0, y_0), orientation ϕ, and semiaxis a and b. The projection of point $\mathbf{x}_i$, or $\mathbf{u}_i$ in coordinate system uv, upon the ellipse is the point with the shortest distance on the ellipse, denoted by $\mathbf{x}_{i1}$ ($\mathbf{u}_{i1}$)

uv is introduced as follows:

$$\mathbf{u} = \mathbf{R}(\mathbf{x} - \mathbf{x}_0), \tag{3}$$

where $\mathbf{R} = \begin{pmatrix} \cos\phi, & \sin\phi \\ -\sin\phi, & \cos\phi \end{pmatrix}$ is a rotation matrix. In coordinate system uv, the ellipse is transformed with center at the origin and without rotation, i.e., with the simple formulation as

$$f_1(u, v) = \frac{u^2}{a^2} + \frac{v^2}{b^2} - 1 = 0. \tag{4}$$

For a data point $\mathbf{u}_i$, the tangent line through its projection point $\mathbf{u}_{i1}$ on the ellipse is perpendicular to the line connecting the two points $\mathbf{u}_i$ and $\mathbf{u}_{i1}$:

$$f_2(u, v) = b^2 u(v_i - v) - a^2 v(u_i - u) = 0. \tag{5}$$

Then, a generalized Newton method is employed to find $\mathbf{u}_{i1}$ as follows:

$$\begin{aligned} \mathbf{Q}_n \Delta\mathbf{u} &= -\mathbf{f}(\mathbf{u}_n) \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \Delta\mathbf{u} \end{aligned} \tag{6}$$

where $\mathbf{f} \triangleq (f_1, f_2)^T$, and the Jacobian matrix

$$\mathbf{Q} = \begin{pmatrix} b^2 u, & a^2 v \\ (a^2 - b^2)v + b^2 v_i, & (a^2 - b^2)u - a^2 u_i \end{pmatrix}. \tag{7}$$

It is noted that several solutions (maximum to 4) could be found by the iteration above. In order to get the right one, one can choose a proper initial value as

$$\mathbf{u}_0 = 0.5(\mathbf{u}_{i2} + \mathbf{u}_{i3}),$$

where $\mathbf{u}_{i2} = \mathbf{u}_i \frac{ab}{\sqrt{b^2 u_i^2 + a^2 v_i^2}}$ and $\mathbf{u}_{i3} = \begin{cases} (u_i, \text{sign}(v_i)\frac{b}{a}\sqrt{a^2 - u_i^2})^T & \text{if } |u_i| < a, \\ (\text{sign}(u_i)a, 0)^T & \text{else.} \end{cases}$, as illustrated in Fig. 1.

Once $\mathbf{u}_1$ has been found, through a backward transformation of Eq. (3), the error distance vector becomes

$$\mathbf{D}_1(\mathbf{x}_i) = \mathbf{x}_i - \mathbf{x}_{i1} = \mathbf{R}^{-1}(\mathbf{u}_i - \mathbf{u}_{i1}). \tag{8}$$

By implicitly describing the orthogonal projection $\mathbf{u}_1$ through f_1 and f_2 defined by Eqs. 4 and 5, the Jacobian matrix with respect to the five parameters $\theta = (x_0, y_0, a, b, \phi)^T$ is as follows:

$$\mathbf{J}_{\mathbf{x}_{i1},\theta} = (\mathbf{R}^{-1}\mathbf{Q}^{-1}\mathbf{B})|_{\mathbf{u}=\mathbf{u}_1}, \tag{9}$$

where $\mathbf{Q}$ is given by Eq. 7 and $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4, \mathbf{B}_5)$ with

$$\begin{aligned}
\mathbf{B}_1 &= (b^2 u \cos\phi - a^2 v \sin\phi, b^2(v_i - v)\cos\phi \\
&\quad + a^2(u_i - u)\sin\phi)^T, \\
\mathbf{B}_2 &= (b^2 u \sin\phi + a^2 v \cos\phi, b^2(v_i - v)\sin\phi \\
&\quad - a^2(u_i - u)\cos\phi)^T, \\
\mathbf{B}_3 &= (a(b^2 - v^2), 2av(u_i - u))^T, \\
\mathbf{B}_4 &= (b(a^2 - u^2), -2bu(v_i - b))^T, \\
\mathbf{B}_5 &= ((a^2 - b^2)uv, (a^2 - b^2)(u^2 - v^2 - uu_i + vv_i))^T.
\end{aligned}$$

Finally, based on Eqs. (8) and (9), the fitting can be accomplished by a Gaussian-Newton iteration as

$$\theta_{n+1} = \theta_n + \lambda \mathbf{J}^{-1}_{\mathbf{x}_{i1},\theta} \mathbf{D}_1(\mathbf{x}_i), \tag{10}$$

where λ denotes a step-size.

Instead of the shortest distance, an alternative approach was proposed to use radial distance, as illustrated by $D_2(\mathbf{x}_i) = \|\mathbf{x}_i - \mathbf{x}_{i2}\|$ in Fig. 1, where $\mathbf{x}_{i2}$ is the intersection of the ellipse and the radial line passing through $\mathbf{x}_i$. Distance D_2 is analytically obtained as [9]

$$\mathcal{D}_2(\boldsymbol{\Theta}, \mathbf{x}) = |\frac{ab}{\sqrt{(\kappa)}} - 1|\sqrt{(x - x_0)^2 + (y - y_0)^2}, \tag{11}$$

with $\kappa \triangleq a^2(\cos\phi(y - y_0) - \sin\phi(x - x_0))^2 + b^2(\cos\phi(x - x_0) + \sin\phi(y - y_0))^2$. However, solution of Eq. 1 can still not be directly reached. Usually, some iterative technique is still required to get the fitting results, such as stochastic gradient algorithm.

Some other developments on geometric fitting are referred to [2, 13].

Algebraic Fitting

In fact, a more commonly used distance measure for ellipse fitting is algebraic distance, which defines the distance by the second-order polynomial

$$\mathcal{D}(\boldsymbol{\Theta}, \mathbf{u}) = ax^2 + bxy + cy^2 + dx + ey + f, \tag{12}$$

with the constraint

$$b^2 - 4ac < 0 \tag{13}$$

and with $\boldsymbol{\Theta} = [a, b, c, d, e, f]^T$. The problem of Eq. (1) with the distance measure given by Eq. (12) has been extensively studied in the name of conic fitting. In order to fit an ellipse, though the general problem of conic fitting could be solved directly, the constraint of Eq. (13) generally makes the methods iterative. By constraining $b^2 - 4ac = -1$ and by transferring the problem to solving a generalized eigensystem, Fitzgibbon et al. [3] propose a fast and direct ellipse fitting method. Specifically, the problem is reformulated as

$$\min_{\boldsymbol{\Theta}} \|\mathbf{D}\boldsymbol{\Theta}\|^2 \text{ subject to } \boldsymbol{\Theta}^T \mathbf{C} \boldsymbol{\Theta} = 1, \tag{14}$$

where $\mathbf{D} = [\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_N]^T$ with $\mathbf{u}_i = [x^2, xy, y^2, x, y, 1]$, and the constraint matrix $\mathbf{C}$ is defined as

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ & & \mathbf{0} & & & \end{pmatrix}, \tag{15}$$

where $\mathbf{0} \in \mathbb{R}^{3\times 6}$ is a null matrix. By introducing Lagrange multipliers λ and differentiating, the conditions for the optimal $\boldsymbol{\Theta}$ become

$$\begin{aligned} \mathbf{S}\Theta &= \lambda \mathbf{C}\Theta, \\ \Theta^T \mathbf{C}\boldsymbol{\Theta} &= 1, \end{aligned} \tag{16}$$

where $\mathbf{S} = \mathbf{D}^T\mathbf{D}$. Thus, problem (14) can be solved by finding the eigenvector Θ_i of the generalized eigensystem defined by Eq. (16). It is also noted that there are six solution pairs (λ_i, Θ_i). By definition, the one corresponding to the minimal positive eigenvalue λ_i is chosen as the solution because

$$\|\mathbf{D}\boldsymbol{\Theta}\|^2 = \boldsymbol{\Theta}^T\mathbf{D}^T\mathbf{D}\boldsymbol{\Theta} = \boldsymbol{\Theta}^T\mathbf{S}\boldsymbol{\Theta} = \lambda\boldsymbol{\Theta}^T\mathbf{C}\boldsymbol{\Theta} = \lambda. \tag{17}$$

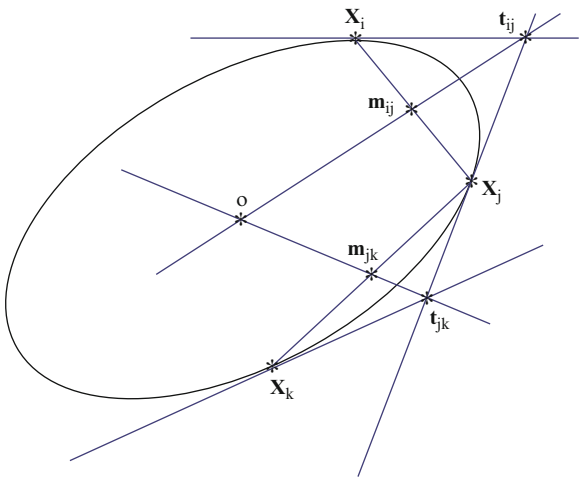
Some other developments on algebraic fitting are referred to [5, 14].

Voting-Based Techniques

Different from the least square fitting techniques which can fit only one primitive, the voting-based methods, consisting of mainly the Hough transform, can detect multiple primitives at once. Even for single ellipse fitting, the Hough transform-based techniques are more robust against outliers. On the other hand, however, Hough transform usually suffers from a larger computational and memory load, due to the fact that an ellipse involves five parameters. To alleviate the problem, there exist some modifications of the original method.

One popular way to tackle the problem is to decompose the five-dimensional space into several subspace with lower dimensions, thanks to some special geometric features of ellipse. A common decomposition method is first to locate the center of the ellipse, and then to specify the remaining three parameters. The center of an ellipse can be located in several ways, based on different geometric features of ellipse.

One geometric feature of ellipse frequently used is described as follows [18]. Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be two points on an ellipse, and find their midpoint $\mathbf{m}_{ij} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)$



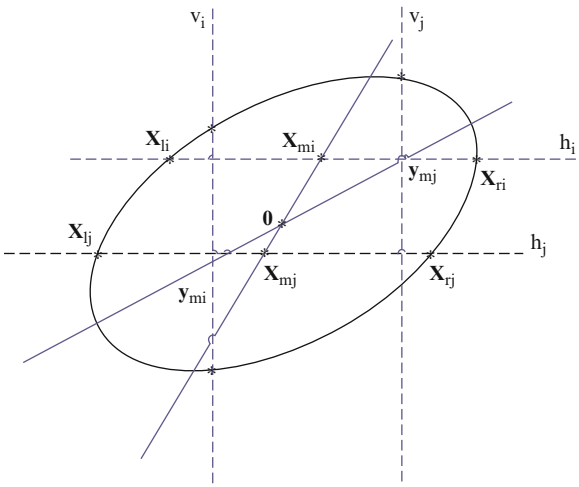
Ellipse Fitting, Fig. 2 Intersection of lines $\overline{t_{ij}m_{ij}}$ and $\overline{t_{jk}m_{jk}}$ is center of the ellipse

and the intersection point $\mathbf{t}_{ij}$ of their tangent lines, as illustrated in Fig. 2. Then, the line connecting the two points $\mathbf{m}_{ij}$ and $\mathbf{t}_{ij}$ is a radial line that passes through center of the ellipse. That is, any point pairs on an ellipse with unparallel tangents can produce a radial line. Consequently, the center candidates can be located in the following way:

1. Generate the radial lines based on every point pairs in the image.
2. Define a 2-dimensional histogram on $x - y$ plane of the image, and record each radial line by incrementing the histogram bin through which the line passes.
3. Find all local maxima in the 2-dimensional histogram as the center candidates.

Another interesting feature of ellipse arises from its symmetry [6]. For two horizontal lines h_i and h_j, their intersections with an ellipse are denoted by $\mathbf{x}_{li}, \mathbf{x}_{ri}$ and $\mathbf{x}_{lj}, \mathbf{x}_{rj}$, respectively, as illustrated in Fig. 3. The line $\overline{\mathbf{x}_{mi}\mathbf{x}_{mj}}$ will pass through center of the ellipse, where the midpoint $\mathbf{x}_{mi} \triangleq \frac{1}{2}(\mathbf{x}_{li} + \mathbf{x}_{ri})$ and $\mathbf{x}_{mj} \triangleq \frac{1}{2}(\mathbf{x}_{lj} + \mathbf{x}_{rj})$. Similarly, for two vertical lines v_i and v_j, the line $\overline{\mathbf{y}_{mi}\mathbf{y}_{mj}}$ also passes through center of the ellipse, where line $\overline{\mathbf{y}_{mi}\mathbf{y}_{mj}}$ is gotten in a similar way as $\overline{\mathbf{x}_{mi}\mathbf{x}_{mj}}$. Thus, intersection of the two lines $\overline{\mathbf{x}_{mi}\mathbf{x}_{mj}}$ and $\overline{\mathbf{y}_{mi}\mathbf{y}_{mj}}$ gives center of the ellipse. Based on the geometric feature, a center detection method is given as follows:

1. Fully horizontally scan the image, and find the midpoints $\mathbf{m}_{hi}$ of every possible point pairs on each scanning line.



Ellipse Fitting, Fig. 3 Intersection of lines $\overline{x_{mi}x_{mj}}$ and $\overline{y_{mi}y_{mj}}$ is center of the ellipse

2. A 2-dimensional Hough transform is applied to find all of the lines l_h formed by the midpoints $\mathbf{m}_{hi}$ $(i = 1, 2, \ldots)$.
3. All lines l_v are found in a similar way as above on vertical scanning.
4. Every intersection of l_h and l_v gives a center candidate.

Once the center candidates have been extracted, detection of the three remaining parameters can be reached through a standard voting process in an accumulator space or can be further decomposed into detection of orientation and semiaxis, respectively. A typical decomposition strategy for the three parameters is to find first axis ratio $\frac{a}{b}$ and orientation ϕ and then axis length [4]. Axis ratio and orientation satisfy the following relationship:

$$\frac{a^2}{b^2} = \frac{(x_i - x_0)\cos\phi + (y_i - y_0)\sin\phi}{(x_i - x_0)\sin\phi - (y_i - y_0)\cos\phi} \tan(\theta_i - \phi + \frac{\pi}{2}), \tag{18}$$

where θ_i denotes the angle of the tangent line through $\mathbf{x}_i$ and x_0, y_0 have been extracted in the previous stage. Thus, a two-dimensional plane can be built based on Eq. (18) for each point on the ellipse, and consequently the axis ratio and orientation can be estimated by finding the maxima in the plane. It is noted that, for each one center candidate, only the data points that participate in the extraction of center are used in the above voting process.

Finally, ellipse fitting can be accomplished by specifying a or b. Taking a as an example as follows, which satisfies

$$a = \sqrt{(x_i - x_0)^2 + \gamma(y_i - y_0)^2}, \tag{19}$$

where $\gamma \triangleq \frac{a}{b}$ is gotten in the previous step. Hence, voting on a one-dimensional accumulator by the data points pertaining each candidate will retrieve the axis length of the target.

There are also some improvements of the Hough transform that are used to alleviate the computational load, such as the randomized Hough transform. Because discussion on the variants of Hough transform is out of the scope of current entry, readers are referred to some other related entries or some recommended readings such as [11].

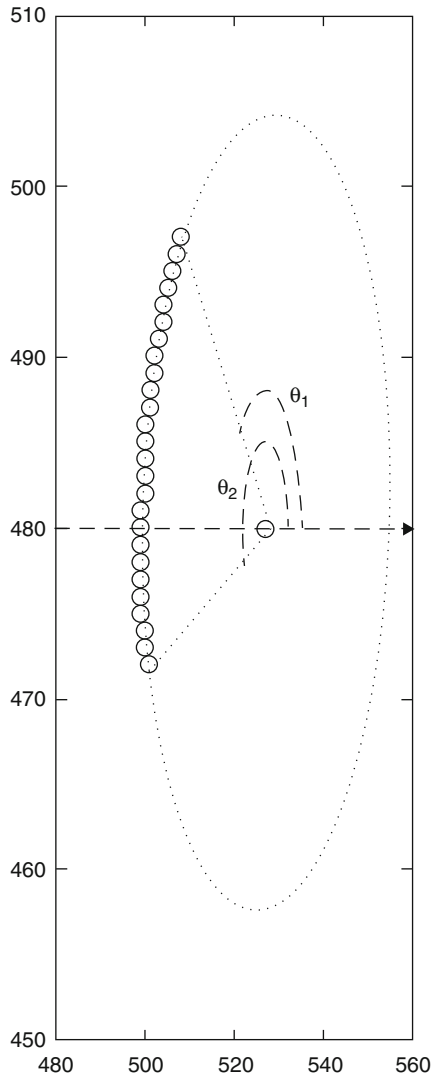
Some other developments on voting-based fitting are referred to [15, 17].

Arc Finding-Based Techniques

Arc finding-based techniques accomplish ellipse fitting by finding elliptic arcs in the edge map. Thus, they in general require edge detection as a preprocessing. Arc finding is of particular interest for ellipse fitting, thanks to the fact that any fragment/elliptic arc (longer than five pixels) of an ellipse can retrieve the whole one. This makes such techniques suitable for ellipse fitting in complicated image because it is usually easier to extract arcs than a whole ellipse from an edge map. Arc finding-based ellipse fitting generally takes the following three steps: arc finding, arc fitting and grouping, and ellipse fitting.

In order to extract ellipse, the arc found should be a *neat* curve without any branches. A simple approach to get neat curve is described as follows [9]: first, a thinning operator is applied to make the edge with one pixel width; second, delete those pixels that have more than two pixels in its 8-neighbor field; finally, a standard chain finding algorithm is used to find neat curves.

Each curve is further fitted by an ellipse, by using a geometric fitting technique mentioned above. The reason for choosing geometric fitting lies in that, first, geometric fitting provides a proper criterion



Ellipse Fitting, Fig. 4 The beginning and ending angles of an elliptic arc

(error measure) to evaluate the curve to be a qualified elliptic arc or not and, second, geometric fitting gives a more accurate result than algebraic fitting and Hough transform on a small elliptic fragment, which is frequently encountered since the neat curve finding tends to break the curves with branches into several shorter ones. Finally, each curve is characterized by a 8-dimensional vector

$$[x_0, y_0, a, b, \phi, e, \theta_1, \theta_2]^T, \tag{20}$$

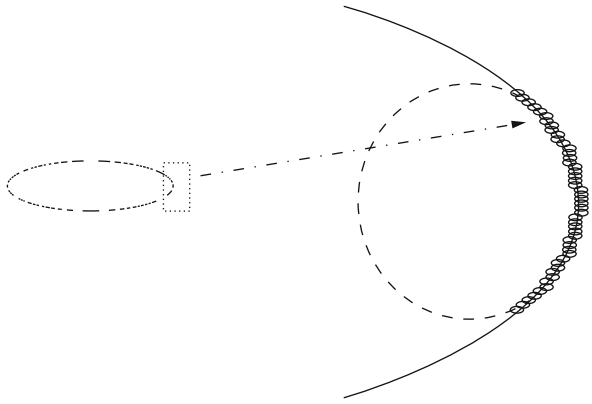
where $e = \frac{1}{N}\sum_{i=1}^{N} \mathcal{D}(\mathbf{\Theta}, \mathbf{x}_i)^2$ denotes the mean square fitting error and θ_1 and θ_2 denote its beginning and ending angles, i.e., its direction, as illustrated in Fig. 4. The curves whose fitting error is smaller than a predefined threshold are chosen as qualified elliptic arcs. The elliptic arcs belonging to one same ellipse are then grouped together according to the following two rules:

- Their positions and shapes are close to each other, by the first five parameters.
- They are mutually complementary in direction to form an entire ellipse, by the last two parameters.

During the arc grouping, the data points belonging to each single ellipse are also segmented. Consequently, each ellipse can be finally fitted by least square fitting on these segmented point sets individually.

It is noticed that the arc finding methods employ an integrated framework since its fitting technique comes from least square fitting. However, such integration seems to be unavoidable in real applications, especially in unstructured environments that usually involve heavy noises.

Some other developments on arc finding-based techniques are referred to [8, 10].

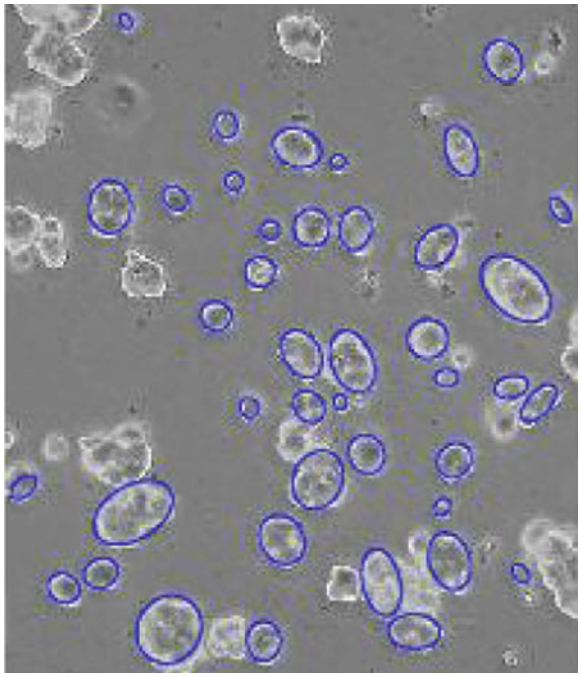


Ellipse Fitting, Fig. 5 Fitting results of a small elliptic fragment: in the image on *right* side, dash ellipse is gotten by algebraic fitting and solid ellipse by geometric fitting

Open Problems

Statistical bias of algebraic fitting make the fitting results tend to shrink, especially on small fragments or on data points with heavier noises. Although there are some works on general conic fitting to remove

Ellipse Fitting, Fig. 6 Real-world application: ellipse fitting results on gear image



Ellipse Fitting, Fig. 7 Real-world application: ellipse fitting results on blood cell image

the bias [7], discussion on direct ellipse fitting is lacked. Since direct algebraic fitting seems to be the fastest algorithm for ellipse fitting, it is of theoretical and practical significance to make some progress on removing bias in the fitting process.

On the other hand, although there are some methods proposed for ellipse fitting in complicated images, it still remains a big challenge to fit ellipse in uncontrolled real-world environments.

Experimental Results

It is not intended to demonstrate all of the algorithms described above. Only two experimental results are given, with the first one to demonstrate the bias of algebraic fitting on a small elliptic arc and the second one to give one glimpse of state-of-art real-world application in noisy environments of ellipse fitting.

Image points used in the first experiments are fetched from a small elliptic fragment, as shown in Fig. 5. The dash ellipse is the fitting result of algebraic fitting, which shrinks to be smaller than the right one as denoted by the bigger solid ellipse that is resulted from geometric fitting.

Two real-world applications of ellipse fitting on holes of gear and blood cells are shown in Figs. 6 and 7, respectively.

References

1. Ahn SJ, Rauh W, Warnecke HJ (2001) Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola. Pattern Recognit 34:2283–2302
2. Cui Y, Weng J, Reynolds H (1996) Estimation of ellipse parameters using optimal minimum variance estimator. Pattern Recognit Lett 17:309–316
3. Fitzgibbon AW, Pilu M, Fisher RB (1999) Direct least-squares fitting of ellipses. IEEE Trans Pattern Anal Mach Intell 21(5):476–480
4. Guil N, Zapata EL (1997) Lower order circle and ellipse hough transform. Pattern Recognit 30:1729–1744
5. Halir R, Flusser V (1998) Numerically stable direct least squares fitting of ellipses. In: WSCG'98 conference proceedings, Plzen-Bory
6. Ho CT, Chen LH (1995) A fast ellipse/circle detector using geometric symmetry. Pattern Recognit 28:117–124
7. Kanatani K (1994) Statistical bias of conic fitting and renormalization. IEEE Trans Pattern Anal Mach Intell 16(3): 320–326
8. Kim E, Haseyama V, Kitajima H (2002) Fast and robust ellipse extraction from complicated images. In: Proceedings of IEEE international conference on information technology and applications, Bathurst, NSW, Australia
9. Liu ZY, Qiao H (2009) Multiple ellipses detection in noisy environments: a hierarchical approach. Pattern Recognit 42:2421–2433

10. Mai F, Hung YS, Zhong H, Sze WF (2008) A hierarchical approach for fast and robust ellipse extraction. Pattern Recognit 8(41):2512–2524
11. McLaughlin RA (1998) Randomized hough transform: improved ellipse detection with comparison. Pattern Recognit Lett 19:299–305
12. Roth G, Levine MD (1994) Geometric primitive extraction using a genetic algorithm. IEEE Trans Pattern Anal Mach Intell 16(9):901–905
13. Spath H (1997) Orthogonal distance fitting by circles and ellipses with given data. Comput Stat 12:343–354
14. Taubin G (1991) Estimation of planar curves, surfaces and non-planar space curves defined by implicit equations with applications to edge and range image segmentation. IEEE Trans Pattern Anal Mach Intell 13(11):1115–1138
15. Tsuji S, Matsumoto F (1978) Detection of ellipses by a modified hough transformation. IEEE Trans Comput 25: 777–781
16. Voss K, Suesse H (1997) Invariant fitting of planar objects by primitives. IEEE Trans Pattern Anal Mach Intell 19(1):80–84
17. Yipa KK, Tama KS, Leung NK (1992) Modification of hough transform for circles and ellipses detection using a 2-dimensional array. Pattern Recognit 25:1007–1022
18. Yuen HK, Illingworth J, Kittler J (1989) Detecting partially occluded ellipses using the hough transform. Image Vis Comput 7:31–37

Ellipse Matching

▸Ellipse Fitting

EM-Algorithm

▸Expectation Maximization Algorithm

Environment Mapping

▸Image-Based Lighting

Epipolar Constraint

Zhengyou Zhang
Microsoft Research, Redmond, WA, USA

Synonyms

Coplanarity constraint

Related Concepts

▸Epipolar Geometry

Definition

Epipolar constraint states that in stereovision with two cameras, given a point in one image, its corresponding point in the other image must lie on a line, known as the *epipolar line*. This constraint arises from the fact that the pair of corresponding image points and the optical centers of the two cameras must lie on a plane (known as *coplanarity constraint*), and the intersection of this plane with the image plane is the epipolar line.

Background

See entry ▸Epipolar Geometry for details.

Epipolar Geometry

Zhengyou Zhang
Microsoft Research, Redmond, WA, USA

Synonyms

Multiple view geometry; Multiview geometry

Related Concepts

▸Epipolar Constraint; ▸Essential Matrix; ▸Fundamental Matrix

Definition

Epipolar geometry describes the geometric relationship between two camera systems. It is captured by a 3×3 matrix known as *essential matrix* for calibrated cameras and as *fundamental matrix* for uncalibrated cameras. It states that for a point observed in one camera, its corresponding point in the other camera must lie on a line. This is known as the *epipolar constraint.*

It reduces the search space of correspondences from two dimensions to one dimension. In motion and structure from motion, this constraint is also known as *coplanarity constraint* because the optical centers of the cameras and a pair of corresponding image points must lie in a single plane.

Background

The epipolar geometry exists between any two camera systems. Consider the case of two cameras as shown in Fig. 1. Let C and C' be the optical centers of the first and second cameras, respectively. Given a point $\mathbf{m}$ in the first image, its corresponding point in the second image is constrained to lie on a line called the *epipolar line* of $\mathbf{m}$, denoted by $\mathbf{l}'_{\mathbf{m}}$. The line $\mathbf{l}'_{\mathbf{m}}$ is the intersection of the plane $\boldsymbol{\Pi}$, defined by $\mathbf{m}$, C, and C' (known as the *epipolar plane*), with the second image plane $\mathcal{I}'$. This is because image point $\mathbf{m}$ may correspond to an arbitrary point on the semi-line $C\mathtt{M}$ ($\mathtt{M}$ may be at infinity) and that the projection of $C\mathtt{M}$ on $\mathcal{I}'$ is the line $\mathbf{l}'_{\mathbf{m}}$. Furthermore, one observes that all epipolar lines of the points in the first image pass through a common point $\mathbf{e}'$, which is called the *epipole*. Epipole $\mathbf{e}'$ is the intersection of the line CC' with the image plane $\mathcal{I}'$. This can be easily understood as follows. For each point $\mathbf{m}_k$ in the first image $\mathcal{I}$, its epipolar line $\mathbf{l}'_{\mathbf{m}_k}$ in $\mathcal{I}'$ is the intersection of the plane $\boldsymbol{\Pi}^k$, defined by $\mathbf{m}_k$, C, and C', with image plane $\mathcal{I}'$. All epipolar planes $\boldsymbol{\Pi}^k$ thus form a pencil of planes containing the line CC'. They must intersect $\mathcal{I}'$ at a common point, which is $\mathbf{e}'$. Finally, one can easily see the symmetry of the epipolar geometry. The corresponding point in the first image of each point $\mathbf{m}'_k$ lying on $\mathbf{l}'_{\mathbf{m}_k}$ must lie on the epipolar line $\mathbf{l}_{\mathbf{m}'_k}$, which is the intersection of the same plane $\boldsymbol{\Pi}^k$ with the first image plane $\mathcal{I}$. All epipolar lines form a pencil containing the epipole $\mathbf{e}$, which is the intersection of the line CC' with the image plane $\mathcal{I}$. The symmetry leads to the following observation. If $\mathbf{m}$ (a point in $\mathcal{I}$) and $\mathbf{m}'$ (a point in $\mathcal{I}'$) correspond to a single physical point $\mathtt{M}$ in space, then $\mathbf{m}$, $\mathbf{m}'$, C, and C' must lie in a single plane. This is the well-known *coplanarity constraint* in solving motion and structure from motion problems when the intrinsic parameters of the cameras are known [1].

The computational significance in matching different views is that for a point in the first image, its correspondence in the second image must lie on the epipolar line in the second image, and then the search space for a correspondence is reduced from 2 dimensions to 1 dimension. This is called the *epipolar constraint*.

If the line linking the two optical centers is parallel to one or both of the image planes, then the epipole in one or both of the images goes to infinity, and the epipolar lines are parallel to each other. Additionally, if the line linking the two optical centers is parallel with the horizontal scanlines of the cameras, then the epipolar lines become horizontal, too. This is the assumption of many stereo algorithms which have horizontal epipolar lines.

Theory

Before proceeding further, the reader is referred to the entry ▸Camera Parameters (Intrinsic, Extrinsic) for the description of camera perspective projection matrix and intrinsic and extrinsic parameters. It is assumed that the reader is familiar with the notation used in that entry.

Assume that the second camera is brought from the position of the first camera through a rotation $\mathbf{R}$ followed by a translation $\mathbf{t}$. Thus, any point (X, Y, Z) in the first camera coordinate system has coordinates (X', Y', Z') in the second camera coordinate system such that

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} + \mathbf{t} \tag{1}$$

where

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{and} \quad \mathbf{t} = \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix}.$$

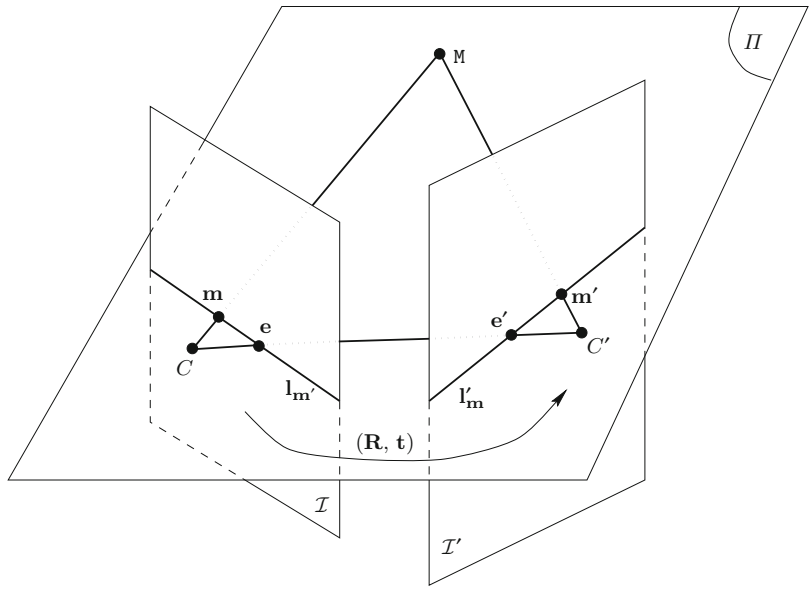
$\mathbf{R}$ has nine components but there are only three degrees of freedom. There are six constraints on $\mathbf{R}$. Indeed, a rotation matrix $\mathbf{R}$ must satisfy

$$\mathbf{R}\mathbf{R}^T = \mathbf{I}, \tag{2}$$

and

$$\det(\mathbf{R}) = 1. \tag{3}$$

Epipolar Geometry, Fig. 1
The epipolar geometry



See, for example [2], for more details on the different representations of the rotation and its properties.

In the following, we first derive the epipolar equation with the normalized image coordinates, then extend it to include the pixel image coordinates, and finally formulate in terms of camera perspective projection matrices.

Working with Normalized Image Coordinates

The two images of a space point $\mathtt{X} = [X, Y, Z]^T$ are $[x, y, 1]^T$ and $[x', y', 1]^T$ in the first and second normalized images, respectively. They are denoted by $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$. Let $\mathtt{X}' = [X', Y', Z']^T$ be the coordinates of the same space point in the second camera coordinate system. From the pinhole model, we have

$$\tilde{\mathbf{x}} = \mathtt{X}/Z,$$
$$\tilde{\mathbf{x}}' = \mathtt{X}'/Z'.$$

Eliminating the structure parameters $\mathtt{X}$ and $\mathtt{X}'$ using Eq. (1), we obtain

$$\tilde{\mathbf{x}} = \frac{1}{Z}(Z'\mathbf{R}\tilde{\mathbf{x}}' + \mathbf{t}),$$

which contains still two unknown structure parameters Z and Z'. The cross product of the above equation with vector $\mathbf{t}$ yields

$$\mathbf{t} \times \tilde{\mathbf{x}} = \frac{Z'}{Z}\mathbf{t} \times \mathbf{R}\tilde{\mathbf{x}}'.$$

Its dot product (or inner product) with $\tilde{\mathbf{x}}$ gives

$$\tilde{\mathbf{x}}^T \mathbf{t} \times (\mathbf{R}\tilde{\mathbf{x}}') = 0. \tag{4}$$

Here, the quantity Z'/Z has been removed.

Equation (4) is very important in solving motion and structure from motion. Geometrically, it is very clear. The three vectors CC', $C\tilde{\mathbf{x}}$, and $C'\tilde{\mathbf{x}}'$ are coplanar. When expressed in the first camera coordinate system, they are equal to $\mathbf{t}$, $\tilde{\mathbf{x}}$, and $\mathbf{R}\tilde{\mathbf{x}}'$, respectively. The coplanarity of the three vectors implies that their mixed product should be equal to 0, which gives Eq. (4).

Let us define a mapping $[\cdot]_\times$ from a 3D vector to a 3×3 *antisymmetric matrix* (also called *skew symmetric matrix*):

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{5}$$

It is clear that

$$[\mathbf{t}]_\times = -[\mathbf{t}]_\times^T. \tag{6}$$

Using this mapping, we can express the cross product of two vectors by the matrix multiplication of a 3×3 matrix and a three-vector: $\mathbf{t} \times \tilde{\mathbf{x}} = [\mathbf{t}]_\times \tilde{\mathbf{x}}$, $\forall \tilde{\mathbf{x}}$. Equation (4) can then be rewritten as

$$\tilde{\mathbf{x}}^T \mathbf{E} \tilde{\mathbf{x}}' = 0, \tag{7}$$

where

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}. \tag{8}$$

We call this equation the *epipolar equation*.

Matrix $\mathbf{E}$ is known under the name of the *essential matrix*. It was first proposed by Longuet-Higgins [1] for structure from motion. The essential matrix is determined completely by the rotation and translation between the two cameras. Because $[\mathbf{t}]_\times$ is anti-symmetric, we have $\det([\mathbf{t}]_\times) = 0$. Thus, we have

$$\det(\mathbf{E}) = \det([\mathbf{t}]_\times) \det(\mathbf{R}) = 0. \tag{9}$$

For more properties of the essential matrix, see [3, 4].

Before gaining an insight of Eq. (7), we recall how to represent a line in a plane. Any line can be described by an equation of the form

$$ax + by + c = 0. \tag{10}$$

Thus, the line can be represented by a three-vector $\mathbf{l} = [a, b, c]^T$ such that a point $\tilde{\mathbf{x}} = [x, y, 1]^T$ on it must satisfy

$$\tilde{\mathbf{x}}^T \mathbf{l} = 0. \tag{11}$$

Of course, the three-vector $\mathbf{l}$ is only defined up to a scale factor. Multiplying $\mathbf{l}$ by any nonzero scalar λ gives $\lambda\mathbf{l}$, which describes exactly the same line. If a line goes through two given points $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$, we have

$$\tilde{\mathbf{x}}_1^T \mathbf{l} = 0 \quad \text{and} \quad \tilde{\mathbf{x}}_2^T \mathbf{l} = 0,$$

and it is easy to see that the line is represented by

$$\mathbf{l} = \tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2, \tag{12}$$

that is, the cross product of the two point vectors.

For point $\tilde{\mathbf{x}}' = [x', y', 1]^T$ in the second image, its corresponding point $\mathtt{X}'$ in space must be on the semi-line $C'\mathtt{X}'_\infty$ passing through $\tilde{\mathbf{x}}'$, where $\mathtt{X}'_\infty$ is a point at infinity. From the pinhole model, point $\mathtt{X}'$ can be represented as

$$\mathtt{X}' = \lambda \tilde{\mathbf{x}}' = \lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \quad \lambda \in (0, \infty).$$

This is in fact the parametric representation of the semi-line $C'\mathtt{X}'_\infty$. If we express this point in the coordinate system of the first camera, we have

$$\mathtt{X} = \mathbf{R}\mathtt{X}' + \mathbf{t} = \lambda \mathbf{R} \tilde{\mathbf{x}}' + \mathbf{t}, \quad \lambda \in (0, \infty).$$

The projection of the semi-line $C'\mathtt{X}'_\infty$ on the first camera is still a line, denoted by $\mathbf{l}_{\mathbf{x}'}$, on which the corresponding point in the first image of point $\mathbf{x}$ must lie. The line $\mathbf{l}_{\mathbf{x}'}$ is known as the *epipolar line* of $\mathbf{x}'$. The epipolar line can be defined by two points. The first point can be obtained by projecting $\mathtt{X}$ with $\lambda = 0$, which gives $\tilde{\mathbf{e}} = \frac{1}{t_Z}\mathbf{t}$, where t_Z is the Z-component of the translation vector $\mathbf{t}$. This is in fact the projection of the optical center C of the second camera on the first camera and is called the *epipole* in the first image. The second point can be obtained by projecting $\mathtt{X}$ with $\lambda = \infty$, which gives $\tilde{\mathbf{x}}_\infty = \frac{1}{\mathbf{r}_3^T \tilde{\mathbf{x}}'} \mathbf{R} \tilde{\mathbf{x}}'$, where $\mathbf{r}_3$ is the third row of the rotation matrix $\mathbf{R}$. As described in the last paragraph, the epipolar line $\mathbf{l}_{\mathbf{x}'}$ is represented by

$$\mathbf{l}_{\mathbf{x}'} = \tilde{\mathbf{e}} \times \tilde{\mathbf{x}}_\infty = \mathbf{t} \times \mathbf{R} \tilde{\mathbf{x}}' = \mathbf{E} \tilde{\mathbf{x}}'. \tag{13}$$

Here we have multiplied the original vector by t_Z and $\mathbf{r}_3^T \tilde{\mathbf{x}}'$ because, as we said, a three-vector for a line is only defined up to a scalar factor.

If now we reverse the role of the two camera, we find that the epipolar geometry is symmetric for the two cameras. For a given point $\mathbf{x}$ in the first image, its corresponding epipolar line in the second image is

$$\mathbf{l}'_{\mathbf{x}} = \mathbf{E}^T \tilde{\mathbf{x}}.$$

It is seen that the transpose of matrix $\mathbf{E}$, $\mathbf{E}^T$, defines the epipolar lines in the second image.

From the above discussion, Eq. (7) says nothing more than that point $\mathbf{x}$ is on the epipolar line $\mathbf{l}_{\mathbf{x}'}$, that is,

$$\tilde{\mathbf{x}}^T \mathbf{l}_{\mathbf{x}'} = 0 \quad \text{with } \mathbf{l}_{\mathbf{x}'} = \mathbf{E} \tilde{\mathbf{x}}',$$

or that point $\tilde{\mathbf{x}}'$ is on the epipolar line $\mathbf{l}'_{\mathbf{x}}$, that is,

$$\mathbf{l}'^T_{\mathbf{x}} \tilde{\mathbf{x}}' = 0 \quad \text{with } \mathbf{l}'_{\mathbf{x}} = \mathbf{E}^T \tilde{\mathbf{x}}.$$

The epipoles are intersections of all epipolar lines. That is, epipoles satisfy all the epipolar line equations.

Let the normalized coordinates of the epipole in the first image be $\mathbf{e}$. Then, from Eq. (7), $\mathbf{e}$ satisfies

$$\tilde{\mathbf{e}}^T \mathbf{E} \tilde{\mathbf{x}}' = 0 \tag{14}$$

regardless of $\mathbf{x}'$. This means,

$$\tilde{\mathbf{e}}^T \mathbf{E} = \mathbf{0}^T \tag{15}$$

at anytime. That is,

$$\tilde{\mathbf{e}}^T \mathbf{E} = \tilde{\mathbf{e}}^T [\mathbf{t}]_\times \mathbf{R} = \mathbf{0}^T. \tag{16}$$

Since $\mathbf{R}$ is an orthonomal matrix, we have

$$\tilde{\mathbf{e}}^T [\mathbf{t}]_\times = \mathbf{0}. \tag{17}$$

The solution is

$$\tilde{\mathbf{e}} = \left[\frac{t_X}{t_Z}, \frac{t_Y}{t_Z}, 1 \right]^T. \tag{18}$$

This is exactly the projection of the optical center of the second camera onto the first image plane, as we have already explained geometrically. For the second image, we have

$$\mathbf{E} \tilde{\mathbf{e}}' = [\mathbf{t}]_\times \mathbf{R} \tilde{\mathbf{e}}' = \mathbf{0}. \tag{19}$$

Thus,

$$\mathbf{R} \tilde{\mathbf{e}}' = \mathbf{t}. \tag{20}$$

The position of the epipole can then be determined as

$$\tilde{\mathbf{e}}' = \frac{1}{\mathbf{r}_3' \cdot \mathbf{t}} \mathbf{R}^T \mathbf{t} = \left[\frac{\mathbf{r}_1' \cdot \mathbf{t}}{\mathbf{r}_3' \cdot \mathbf{t}}, \frac{\mathbf{r}_1' \cdot \mathbf{t}}{\mathbf{r}_3' \cdot \mathbf{t}}, 1 \right]^T, \tag{21}$$

where $\mathbf{r}_i' = [r_{1i}, r_{2i}, r_{3i}]^T$, and $i = 1, 2, 3$ are the column vectors of $\mathbf{R}$.

For the epipole in the first image to go to infinity, we must have

$$t_Z = 0. \tag{22}$$

This means that the translation of the camera has to be within the focal plane of the first camera. For both epipoles in the two images to go to infinity, then

$$\mathbf{r}_3' \cdot \mathbf{t} = 0. \tag{23}$$

This implies that the optical center of the first camera lies in the focal plane of the second camera. Furthermore, if we require the two focal planes to be a single one, then besides $t_Z = 0$, r_{13} and r_{23} have to be 0. Since $\|\mathbf{r}_i'\| = 1$, we have

$$r_{33} = 1. \tag{24}$$

Thus $\mathbf{R}$ is in the form of

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{25}$$

This means that the rotation can be only around the optical axis of the cameras.

Substituting Eq. (22) and (25) for (8), we have

$$\begin{aligned} \mathbf{E} &= [\mathbf{t}]_\times \mathbf{R} \\ &= \begin{bmatrix} 0 & 0 & t_Y \\ 0 & 0 & -t_X \\ -t_Y \cos\theta - t_X \sin\theta & -t_Y \sin\theta + t_X \cos\theta & 0 \end{bmatrix}. \end{aligned} \tag{26}$$

If we expand the above equation, it is clear that there is only linear terms of the image coordinates, rather than quadric terms in the original form. That means the epipolar lines are parallel in both images and the orientations are independent of the image points.

Working with Pixel Image Coordinates

If two points $\mathbf{m}$ and $\mathbf{m}'$, expressed in pixel image coordinates in the first and second camera, are in correspondence, they must satisfy the following equation

$$\tilde{\mathbf{m}}^T \mathbf{F} \tilde{\mathbf{m}}' = 0, \tag{27}$$

where

$$\mathbf{F} = \mathbf{A}^{-T} \mathbf{E} \mathbf{A}'^{-1}, \tag{28}$$

and $\mathbf{A}$ and $\mathbf{A}'$ are respectively the of the first and second camera. Equation (27) is easily verified. From the pinhole camera model, the $\mathbf{x}$ are related to the pixel coordinates $\mathbf{m}$ by $\tilde{\mathbf{x}} = \mathbf{A}^{-1} \tilde{\mathbf{m}}$. Plunging it into Eq. (7) yields Eq. (27). This is a fundamental constraint for two pixels to be in correspondence between two images.

As with the normalized image coordinates, the above Eq. (27) can also be derived from the pinhole model. Without loss of generality, we assume

that the world coordinate system coincides with the second camera coordinate system. From the camera perspective projection model, we have

$$s\tilde{\mathbf{m}} = \mathbf{A}\,[\mathbf{R}\ \mathbf{t}]\begin{bmatrix}\mathtt{M}'\\1\end{bmatrix}$$
$$s'\tilde{\mathbf{m}}' = \mathbf{A}'\,[\mathbf{I}\ \mathbf{0}]\begin{bmatrix}\mathtt{M}'\\1\end{bmatrix}.$$

Eliminating $\mathtt{M}'$, s, and s' in the above two equations, we obtain, not at all surprising, Eq. (27).

The 3×3 matrix $\mathbf{F}$ is called the *fundamental matrix*. With this fundamental matrix, we can express the epipolar equation for two unnormalized images in the same form as for the normalized images. Since $\det(\mathbf{E}) = 0$,

$$\det(\mathbf{F}) = 0. \tag{29}$$

$\mathbf{F}$ is of rank 2. Besides, it is only defined up to a scalar factor. If $\mathbf{F}$ is multiplied by an arbitray scalar, Eq. (27) still holds. Therefore, a fundamental matrix has only seven degrees of freedom. There are only seven independent parameters among the nine elements of the fundamental matrix.

We now derive the expression of the epipoles. The epipole $\mathbf{e}$ in the first image is the projection of the optical center C' of the second camera. Since $C' = \mathbf{0}$, from the pinhole model, we have

$$s_e\tilde{\mathbf{e}} = \mathbf{A}\,[\mathbf{R}\ \mathbf{t}]\begin{bmatrix}0\\1\end{bmatrix} = \mathbf{A}\mathbf{t}, \tag{30}$$

where s_e is a scale factor. Thus, the epipole $\tilde{\mathbf{e}}$ is equal to $\mathbf{A}\mathbf{t}$ divided by its third element. Similarly, the epipole $\mathbf{e}'$ in the second image is the projection of the optical center C of the first camera. The optical center is determined by

$$\mathbf{A}\,[\mathbf{R}\ \mathbf{t}]\begin{bmatrix}C\\1\end{bmatrix} = \mathbf{0},$$

which gives

$$C = -\mathbf{R}^{-1}\mathbf{t}.$$

Therefore, the epipole $\mathbf{e}'$ is given by

$$s'_e\tilde{\mathbf{e}}' = \mathbf{A}'\,[\mathbf{I}\ \mathbf{0}]\begin{bmatrix}C\\1\end{bmatrix} = -\mathbf{A}'\mathbf{R}^{-1}\mathbf{t}, \tag{31}$$

that is, it is equal to $-\mathbf{A}'\mathbf{R}^{-1}\mathbf{t}$ divided by the third element of the vector.

Now we show, for a given point $\mathbf{m}'$ in the second image, how to compute the corresponding epipolar line $\mathbf{l}_{\mathbf{m}'}$ in the first image. It is determined by two points. Besides the epipole $\mathbf{e}$, we need another point. This point can be the projection of any point $\widehat{\mathtt{M}}'$ on the optical ray $\langle C', \tilde{\mathbf{m}}'\rangle$. In particular, we can choose $\widehat{\mathtt{M}}'$ such that

$$\tilde{\mathbf{m}}' = \mathbf{A}'\ [\mathbf{I}\ \mathbf{0}]\begin{bmatrix}\widehat{\mathtt{M}}'\\1\end{bmatrix} = \mathbf{A}'\widehat{\mathtt{M}}',$$

that is, the scale factor is equal to 1. This gives $\widehat{\mathtt{M}}' = \mathbf{A}'^{-1}\tilde{\mathbf{m}}'$. The projection of this point in the first camera, $\widehat{\mathbf{m}}$, is given by

$$s_m\tilde{\widehat{\mathbf{m}}} = \mathbf{A}\ [\mathbf{R}\ \mathbf{t}]\begin{bmatrix}\widehat{\mathtt{M}}'\\1\end{bmatrix} = \mathbf{A}(\mathbf{R}\mathbf{A}'^{-1}\tilde{\mathbf{m}}' + \mathbf{t}),$$

where s_m is the scale factor. As already described in Eq. (10) on page 250, a line can be represented by a three-vector defined *up to a scale factor*. According to Eq. (12), the epipolar line $\mathbf{l}_{\mathbf{m}'}$ is given by

$$\begin{aligned}\mathbf{l}_{\mathbf{m}'} &= s_e s_m \tilde{\mathbf{e}} \times \tilde{\widehat{\mathbf{m}}}\\ &= (\mathbf{A}\mathbf{t}) \times [\mathbf{A}(\mathbf{R}\mathbf{A}'^{-1}\tilde{\mathbf{m}}' + \mathbf{t})]\\ &= (\mathbf{A}\mathbf{t}) \times (\mathbf{A}\mathbf{R}\mathbf{A}'^{-1}\tilde{\mathbf{m}}').\end{aligned}$$

It can be shown that $(\mathbf{A}\mathbf{x}) \times (\mathbf{A}\mathbf{y}) = \det(\mathbf{A})\,\mathbf{A}^{-T}(\mathbf{x} \times \mathbf{y})$ for all vectors $\mathbf{x}$ and $\mathbf{y}$ if matrix $\mathbf{A}$ is invertible. Therefore, we have

$$\mathbf{l}_{\mathbf{m}'} = \mathbf{A}^{-T}[\mathbf{t} \times (\mathbf{R}\mathbf{A}'^{-1}\mathbf{m}')] = \mathbf{F}\tilde{\mathbf{m}}'$$

with $\mathbf{F}$ given by Eq. (28). Then any point $\mathbf{m}$ on the epipolar line of $\mathbf{m}'$ satisfies $\tilde{\mathbf{m}}^T\mathbf{F}\tilde{\mathbf{m}}' = 0$, and this is exactly Eq. (34). Therefore, we obtain geometrically the same equation.

Now we reverse the role of the two images and consider the epipolar line $\mathbf{l}'_{\mathbf{m}}$ in the second image for a given point $\mathbf{m}$ in the first image. Line $\mathbf{l}'_{\mathbf{m}}$ goes through the epipole $\mathbf{e}'$. We choose the projection of a point $\widehat{\mathtt{M}}'$ on the optical ray $\langle C, \tilde{\mathbf{m}}\rangle$ such that

$$\tilde{\mathbf{m}} = \mathbf{A}\,[\mathbf{R}\ \mathbf{t}]\begin{bmatrix}\widehat{\mathtt{M}}'\\1\end{bmatrix},$$

that is, the scale factor is chosen to be 1. This gives

$$\widehat{\mathtt{M}}' = (\mathbf{AR})^{-1}(\tilde{\mathbf{m}} - \mathbf{At}).$$

Its projection in the second camera gives

$$\begin{aligned} s'_m \tilde{\hat{\mathbf{m}}}' &= \mathbf{A}' [\mathbf{I}\ \mathbf{0}] \begin{bmatrix} \widehat{\mathtt{M}}' \\ 1 \end{bmatrix} \\ &= \mathbf{A}'(\mathbf{AR})^{-1}(\tilde{\mathbf{m}} - \mathbf{At}) \\ &= \mathbf{A}'\mathbf{R}^{-1}\mathbf{A}^{-1}\tilde{\mathbf{m}} - \mathbf{A}'\mathbf{R}^{-1}\mathbf{t}. \end{aligned}$$

The epipolar line $\mathbf{l}'_{\mathbf{m}}$ is thus represented by

$$\begin{aligned} \mathbf{l}'_{\mathbf{m}} &= s'_e s'_m \tilde{\mathbf{e}}' \times \tilde{\hat{\mathbf{m}}}' \\ &= -(\mathbf{A}'\mathbf{R}^{-1}\mathbf{t}) \times (\mathbf{A}'\mathbf{R}^{-1}\mathbf{A}^{-1}\tilde{\mathbf{m}}) \\ &= -(\mathbf{A}'\mathbf{R}^{-1})^{-T}(\mathbf{t} \times \mathbf{A}^{-1}\tilde{\mathbf{m}}) \\ &= -\mathbf{A}'^{-T}\mathbf{R}^T[\mathbf{t}]_\times \mathbf{A}^{-1}\tilde{\mathbf{m}} \\ &= \mathbf{F}^T\tilde{\mathbf{m}}. \end{aligned}$$

In the above, we have used the following properties:
- $(\mathbf{Ax}) \times (\mathbf{Ay}) = \det(\mathbf{A})\mathbf{A}^{-T}(\mathbf{x} \times \mathbf{y}),\ \forall \mathbf{x}, \mathbf{y}$ if matrix $\mathbf{A}$ is invertible.
- $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ if matrices $\mathbf{A}$ and $\mathbf{B}$ are invertible.
- $\mathbf{R}^T = \mathbf{R}^{-1}$ if $\mathbf{R}$ is a rotation matrix.
- $[\mathbf{t}]_\times^T = -[\mathbf{t}]_\times$ if $[\mathbf{t}]_\times$ is an antisymmetric matrix.

It is thus clear that if $\mathbf{F}$ describes epipolar lines in the first image for points given in the second image, then $\mathbf{F}^T$ describes epipolar lines in the second image for points given in the first image. The two images play a symmetric role in the epipolar geometry.

We now compute the epipoles from a different point of view. By definition, all epipolar lines in the first image go through the epipole $\mathbf{e}$. This implies

$$\tilde{\mathbf{e}}^T\mathbf{F}\tilde{\mathbf{m}}' = 0, \quad \forall \mathbf{m}',$$

or in vector equation form

$$\mathbf{F}^T\tilde{\mathbf{e}} = \mathbf{0}. \tag{32}$$

Pludging Eq. (28) into the above equation gives

$$\mathbf{A}'^{-T}\mathbf{R}^T[\mathbf{t}]_\times \mathbf{A}^{-1}\tilde{\mathbf{e}} = \mathbf{0}.$$

Because $[\mathbf{t}]_\times \mathbf{t} = \mathbf{0}$, up to a scale factor, we have $s_e\mathbf{A}^{-1}\tilde{\mathbf{e}} = \mathbf{t}$, and thus $s_e\tilde{\mathbf{e}} = \mathbf{At}$. This is exactly Eq. (30). Similarly, for the epipole $\mathbf{e}'$ in the second image, we have

$$\mathbf{F}\tilde{\mathbf{e}}' = \mathbf{0} \tag{33}$$

or

$$\mathbf{A}^{-T}[\mathbf{t}]_\times \mathbf{R}\mathbf{A}'^{-1}\tilde{\mathbf{e}}' = \mathbf{0}.$$

This gives $s'_e\mathbf{R}\mathbf{A}'^{-1}\tilde{\mathbf{e}}' = -\mathbf{t}$, or $s'_e\tilde{\mathbf{e}}' = -\mathbf{A}'\mathbf{R}^{-1}\mathbf{t}$. This is exactly Eq. (31).

Working with Camera Perspective Projection Matrices

In several applications, for example, in the case of calibrated stereo, the camera perspective projection matrices are given, and we want to compute the epipolar geometry. Let $\mathbf{P}$ and $\mathbf{P}'$ be the projection matrices of the first and second camera. Furthermore, the 3×4 matrix $\mathbf{P}$ is decomposed as the concatenation of a 3×3 submatrix $\mathbf{B}$ and a three-vector $\mathbf{b}$, that is, $\mathbf{P} = [\mathbf{B}\ \mathbf{b}]$. Similarly, $\mathbf{P}' = [\mathbf{B}'\ \mathbf{b}']$.

From the pinhole model, we have

$$\begin{aligned} s\tilde{\mathbf{m}} &= [\mathbf{B}\ \mathbf{b}] \begin{bmatrix} \mathtt{M}' \\ 1 \end{bmatrix} \\ s'\tilde{\mathbf{m}}' &= [\mathbf{B}'\ \mathbf{b}'] \begin{bmatrix} \mathtt{M}' \\ 1 \end{bmatrix}. \end{aligned}$$

Assume that $\mathbf{B}$ and $\mathbf{B}'$ are invertible, we can compute $\mathtt{M}'$ from each of the above equations:

$$\begin{aligned} \mathtt{M}' &= s\mathbf{B}^{-1}\tilde{\mathbf{m}} - \mathbf{B}^{-1}\mathbf{b} \\ \mathtt{M}' &= s'\mathbf{B}'^{-1}\tilde{\mathbf{m}}' - \mathbf{B}'^{-1}\mathbf{b}'. \end{aligned}$$

The right sides of the above equations must be equal, which gives

$$s\mathbf{B}^{-1}\tilde{\mathbf{m}} = s'\mathbf{B}'^{-1}\tilde{\mathbf{m}}' + \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}'^{-1}\mathbf{b}'.$$

Multiplying both sides by $\mathbf{B}$ gives

$$s\tilde{\mathbf{m}} = s'\mathbf{B}\mathbf{B}'^{-1}\tilde{\mathbf{m}}' + \mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}'.$$

Performing a cross product with $\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}'$ yields

$$s(\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}') \times \tilde{\mathbf{m}} = s'(\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}') \times \mathbf{B}\mathbf{B}'^{-1}\tilde{\mathbf{m}}'.$$

Eliminating the arbitrary scalars s and s' by multiplying $\tilde{\mathbf{m}}^T$ from the left (i.e., dot product) gives

$$\tilde{\mathbf{m}}^T \mathbf{F} \tilde{\mathbf{m}}' = 0, \tag{34}$$

where

$$\mathbf{F} = [\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}']_\times \mathbf{B}\mathbf{B}'^{-1}. \tag{35}$$

We thus obtain the epipolar equation in terms of the perspective projection matrices. Again, it is clear that the roles of $\mathbf{m}$ and $\mathbf{m}'$ are symmetric, and we have $\tilde{\mathbf{m}}'^T \mathbf{F}^T \tilde{\mathbf{m}} = 0$.

Now let us show how to compute the epipoles. The epipole $\mathbf{e}$ in the first image is the projection of the optical center C' of the second camera, and the optical center C' is given by

$$C' = -\mathbf{B}'^{-1}\mathbf{b}'.$$

We thus have

$$s_e \tilde{\mathbf{e}} = \mathbf{P} \begin{bmatrix} C' \\ 1 \end{bmatrix} = \mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}', \tag{36}$$

where s_e is a scale factor. Thus, epipole $\mathbf{e}$ is equal to $(\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}')$ divided by its third element. Similarly, the epipole in the second image, $\mathbf{e}'$, is equal to $(\mathbf{b}' - \mathbf{B}'\mathbf{B}^{-1}\mathbf{b})$ divided by its third element.

Next, we show how, for a given point $\mathbf{m}'$ in the second image, to compute the corresponding epipolar line $\mathbf{l}_{\mathbf{m}'}$ in the first image. The epipolar line must go through the epipole $\mathbf{e}$. We thus need another point to determine it. This point can be the projection of any point $\widehat{\mathbb{M}}'$ on the optical ray $\langle C', \tilde{\mathbf{m}}' \rangle$. In particular, we can choose $\widehat{\mathbb{M}}'$ such that

$$\tilde{\mathbf{m}}' = \mathbf{P}' \begin{bmatrix} \widehat{\mathbb{M}}' \\ 1 \end{bmatrix} = \mathbf{B}'\widehat{\mathbb{M}}' + \mathbf{b}',$$

that is, the scale factor is equal to 1. This gives $\widehat{\mathbb{M}}' = \mathbf{B}'^{-1}(\tilde{\mathbf{m}}' - \mathbf{b}')$. According to the pinhole model, the image $\hat{\mathbf{m}}$ of this point is given by

$$s_m \tilde{\hat{\mathbf{m}}} = \mathbf{P} \begin{bmatrix} \widehat{\mathbb{M}}' \\ 1 \end{bmatrix} = \mathbf{B}\mathbf{B}'^{-1}\tilde{\mathbf{m}}' + (\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}'),$$

where s_m is the scale factor. As already described in Eq. (10) on page 250, a line can be represented by a three-vector defined *up to a scale factor*. According to Eq. (12), the epipolar line $\mathbf{l}_{\mathbf{m}'}$ is given by

$$\begin{aligned} \mathbf{l}_{\mathbf{m}'} &= s_e s_m \tilde{\mathbf{e}} \times \tilde{\hat{\mathbf{m}}} \\ &= (\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}') \times [\mathbf{B}\mathbf{B}'^{-1}\tilde{\mathbf{m}}' + (\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}')] \\ &= (\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}') \times (\mathbf{B}\mathbf{B}'^{-1}\tilde{\mathbf{m}}'), \end{aligned}$$

or

$$\mathbf{l}_{\mathbf{m}'} = \mathbf{F}\tilde{\mathbf{m}}', \tag{37}$$

where $\mathbf{F}$ is given by Eq. (35). Then any point $\mathbf{m}$ on the epipolar line of $\mathbf{m}'$ satisfies $\tilde{\mathbf{m}}^T \mathbf{F} \tilde{\mathbf{m}}' = 0$, and this is exactly Eq. (34). Therefore, we obtain geometrically the same equation. Because of symmetry, for a given point $\mathbf{m}$ in the first image, its corresponding epipolar line in the second image is represented by the vector $\mathbf{F}^T \tilde{\mathbf{m}}$.

Now, we show that if the images are calibrated, then the $\mathbf{F}$ is reduced to the $\mathbf{E}$. Since the images are calibrated, the points $\mathbf{m}$ can be expressed in normalized coordinates, that is, $\mathbf{m} = \mathbf{x}$. Without loss of generality, the world coordinate system is assumed to coincide with the second camera coordinate system. From the perspective projection model, we have the following camera projection matrices:

$$\mathbf{P} = [\mathbf{R}\ \mathbf{t}] \quad \text{and} \quad \mathbf{P}' = [\mathbf{I}\ \mathbf{0}].$$

This implies that $\mathbf{B} = \mathbf{R}$, $\mathbf{b} = \mathbf{t}$, $\mathbf{B}' = \mathbf{I}$, and $\mathbf{b}' = \mathbf{0}$. Pludging them into Eq. (35) gives $\mathbf{F} = [\mathbf{t}]_\times \mathbf{R}$, which is exactly the essential matrix Eq. (8).

In the above derivation of the fundamental matrix, a camera perspective projection matrix $\mathbf{P}$ is decomposed into a 3×3 matrix $\mathbf{B}$ and a three-vector $\mathbf{b}$, and $\mathbf{B}$ must be invertible. Later, we provide a more general derivation directly in terms of the camera projection matrices $\mathbf{P}$ and $\mathbf{P}'$.

Fundamental Matrix and Epipolar Transformation

We examine the relationship between the fundamental matrix and the (i.e., the transformation of the epipoles and the epipolar lines between the two images).

For any point $\mathbf{m}'$ in the second image, its epipolar line $\mathbf{l}_{\mathbf{m}'}$ in the first image is given by $\mathbf{l}'_{\mathbf{m}} = \mathbf{F}\tilde{\mathbf{m}}'$. It must go through the $\tilde{\mathbf{e}} = [e_1, e_2, e_3]^T$ and a point $\tilde{\mathbf{m}} = [u, v, s]^T$, that is, $\mathbf{l}_{\mathbf{m}'} = \tilde{\mathbf{e}} \times \tilde{\mathbf{m}} = \mathbf{F}\tilde{\mathbf{m}}'$. Here, we use the for the image points. Symmetrically, the epipolar line in the second image $\mathbf{l}'_{\mathbf{m}}$ of point $\mathbf{m}$ is given by $\mathbf{l}'_{\mathbf{m}} = \mathbf{F}^T\tilde{\mathbf{m}}$ and must go through the epipole $\tilde{\mathbf{e}}' = [e'_1, e'_2, e'_3]^T$ and a point $\tilde{\mathbf{m}}' = [u', v', s']^T$, that is, $\mathbf{l}'_{\mathbf{m}} = \tilde{\mathbf{e}}' \times \tilde{\mathbf{m}}' = \mathbf{F}^T\tilde{\mathbf{m}}$. In other words, the epipole $\mathbf{e}'$ is *on* the epipolar line $\mathbf{l}'_{\mathbf{m}}$ for any point $\mathbf{m}$; that is,

$$\tilde{\mathbf{e}}'^T \mathbf{l}'_{\mathbf{m}} = \tilde{\mathbf{e}}'^T \mathbf{F}^T \tilde{\mathbf{m}} = 0 \quad \forall \mathbf{m},$$

which yields

$$\mathbf{F}\tilde{\mathbf{e}}' = \mathbf{0}. \tag{38}$$

Let $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$ be the column vectors of $\mathbf{F}$, and we have $e'_1\mathbf{c}_1 + e'_2\mathbf{c}_2 + e'_3\mathbf{c}_3 = \mathbf{0}$; thus the rank of $\mathbf{F}$ is at most two. The solution to the epipole $\tilde{\mathbf{e}}'$ is given by

$$\begin{aligned} e'_1 &= F_{23}F_{12} - F_{22}F_{13} \\ e'_2 &= F_{13}F_{21} - F_{11}F_{23} \\ e'_3 &= F_{22}F_{11} - F_{21}F_{12}, \end{aligned} \tag{39}$$

up to, of course, a scale factor. Similarly, for the epipole in the first image, we have

$$\mathbf{F}^T\tilde{\mathbf{e}} = \mathbf{0}, \tag{40}$$

which gives

$$\begin{aligned} e_1 &= F_{32}F_{21} - F_{22}F_{31} \\ e_2 &= F_{31}F_{12} - F_{11}F_{32} \\ e_3 &= F_{22}F_{11} - F_{21}F_{12}, \end{aligned} \tag{41}$$

also up to a scale factor.

Now let us examine the relationship between the epipolar lines. Once the epipole is known, an epipolar line can be parameterized by its direction vector. Consider $\mathbf{l}'_{\mathbf{m}} = \tilde{\mathbf{e}}' \times \tilde{\mathbf{m}}'$; its direction vector $\mathbf{u}'$ can be parameterized by one parameter τ' such that $\mathbf{u}' = [1, \tau', 0]^T$. A particular point on $\mathbf{l}'_{\mathbf{m}}$ is then given by $\tilde{\mathbf{m}}' = \tilde{\mathbf{e}}' + \lambda'\mathbf{u}'$, where λ' is a scalar. Its epipolar line in the first image is given by

$$\begin{aligned} \mathbf{l}_{\mathbf{m}'} &= \mathbf{F}\tilde{\mathbf{m}}' = \mathbf{F}\tilde{\mathbf{e}}' + \lambda'\mathbf{F}\mathbf{u}' = \lambda'\mathbf{F}\mathbf{u}' \\ &= \lambda' \begin{bmatrix} F_{11} + F_{12}\tau' \\ F_{21} + F_{22}\tau' \\ F_{31} + F_{32}\tau' \end{bmatrix}. \end{aligned} \tag{42}$$

This line can also be parameterized by its direction vector $\mathbf{u} = [1, \tau, 0]^T$ in the first image, which implies that

$$\begin{aligned} \mathbf{l}_{\mathbf{m}'} &\cong \tilde{\mathbf{e}} \times (\tilde{\mathbf{e}} + \lambda\mathbf{u}) = \lambda\tilde{\mathbf{e}} \times \mathbf{u} \\ &= \lambda \begin{bmatrix} -(F_{11}F_{22} - F_{21}F_{12})\tau \\ F_{11}F_{22} - F_{21}F_{12} \\ (F_{32}F_{21} - F_{22}F_{31})\tau - F_{31}F_{12} + F_{11}F_{32} \end{bmatrix}, \end{aligned} \tag{43}$$

where $\cong$ means "equal" up to a scale factor and λ is a scalar. By requiring that Eq. (42) and Eq. (43) represent the same line, we have

$$\tau = \frac{a\tau' + b}{c\tau' + d}, \tag{44}$$

where

$$\begin{aligned} a &= F_{12} \\ b &= F_{11} \\ c &= -F_{22} \\ d &= -F_{21}. \end{aligned} \tag{45}$$

Writing in matrix form gives

$$\rho \begin{bmatrix} \tau \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \tau' \\ 1 \end{bmatrix},$$

where ρ is a scale factor. This relation is known as the *homography* between τ and τ', and we say that *there is a homography between the epipolar lines in the first image and those in the second image*. The above is, of course, only valid for the epipolar lines having the nonzero first element in the direction vector. If the first element is equal to zero, we should parameterize the direction vector as $[\tau, 1, 0]^T$, and similar results can be obtained.

At this point, we can see that the epipolar transformation is defined by the coordinates $\tilde{\mathbf{e}} = [e_1, e_2, e_3]^T$ and $\tilde{\mathbf{e}}' = [e'_1, e'_2, e'_3]^T$ of the epipoles and the four parameters a, b, c, and d of the homography between the two pencils of the epipolar lines. The coordinates of each epipole are defined up to a scale factor, and the parameters of the homography, as can be seen in Eq. (44), are also defined up to a scale factor. Thus, we have in total seven free parameters. This is exactly the number of parameters of the fundamental matrix.

If we have identified the parameters of the epipolar transformation, that is, the coordinates of the two epipoles and the coefficients of the homography, then we can construct the fundamental matrix, from Eq. (39), (41), and (45), as

$$
\begin{aligned}
F_{11} &= be_3e'_3 \\
F_{12} &= ae_3e'_3 \\
F_{13} &= -(ae'_2 + be'_1)e_3 \\
F_{21} &= -de_3e'_3 \\
F_{22} &= -ce_3e'_3 \\
F_{23} &= (ce'_2 + de'_1)e_3 \\
F_{31} &= (de_2 - be_1)e'_3 \\
F_{32} &= (ce_2 - ae_1)e'_3 \\
F_{33} &= -(ce'_2 + de'_1)e_2 + (ae'_2 + be'_1)e_1.
\end{aligned} \tag{46}
$$

The determinant $ad - bc$ of the homography is equal to the determinant of the first 2×2 submatrix of $\mathbf{F}$, $F_{11}F_{22} - F_{12}F_{21}$, which is zero when the epipoles are at infinity.

General Form of Epipolar Equation for Any Projection Model

In this section, we will derive a which does not assume any particular projection model.

Intersecting Two Optical Rays

The projections for the first and second cameras are represented respectively as

$$s\tilde{\mathbf{m}} = \mathbf{P}\tilde{\mathtt{M}}, \tag{47}$$

and

$$s'\tilde{\mathbf{m}}' = \mathbf{P}'\tilde{\mathtt{M}}', \tag{48}$$

where $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{m}}'$ are augmented image coordinates and $\tilde{\mathtt{M}}$ and $\tilde{\mathtt{M}}'$ are augmented space coordinates of a single point in the two camera coordinate systems. Here both projection matrices *do not include the extrinsic parameters*.

The same point in the two camera coordinate systems can be related by

$$\tilde{\mathtt{M}} = \mathbf{D}\tilde{\mathtt{M}}', \tag{49}$$

where

$$\mathbf{D} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix}$$

is the Euclidean transformation matrix compactly representing both rotation and translation. Now substituting Eq. (49) for (47), we have

$$s\tilde{\mathbf{m}} = \mathbf{P}\mathbf{D}\tilde{\mathtt{M}}'. \tag{50}$$

For an image point $\tilde{\mathbf{m}}'$, Eq. (48) actually defines an optical ray on which every space point $\tilde{\mathtt{M}}'$ projects onto the second image at $\tilde{\mathbf{m}}'$. This optical ray can be written in pamametric form as

$$\tilde{\mathtt{M}}' = s'\mathbf{P}'^{+}\tilde{\mathbf{m}}' + \mathbf{p}'^{\perp}, \tag{51}$$

where $\mathbf{P}'^{+}$ is the pseudoinverse matrix of $\mathbf{P}'$:

$$\mathbf{P}'^{+} = \mathbf{P}'^{T}(\mathbf{P}'\mathbf{P}'^{T})^{-1}, \tag{52}$$

and $\mathbf{p}'^{\perp}$ is a four-vector that is perpendicular to all the row vectors of $\mathbf{P}'$, that is,

$$\mathbf{P}'\mathbf{p}'^{\perp} = \mathbf{0}. \tag{53}$$

There are an infinite number of matrices that satisfy $\mathbf{P}'\mathbf{P}'^{+} = \mathbf{I}$. Thus, $\mathbf{P}'^{+}$ is not unique. See [5, 6] for how to derive this particular pseudoinverse matrix.

It remains to determine $\mathbf{p}'^{\perp}$. First note that such a vector does exist because the difference between the row dimension and column dimension is one, and the row vectors are generally independent of each other. Actually, one way to obtain $\mathbf{p}'^{\perp}$ is

$$\mathbf{p}'^{\perp} = (\mathbf{I} - \mathbf{P}'^{+}\mathbf{P}')\boldsymbol{\omega}, \tag{54}$$

where $\boldsymbol{\omega}$ is an arbitrary four-vector. To show that it is perpendicular to every row vector of $\mathbf{P}'$, we multiply $\mathbf{P}'$ and $\mathbf{p}'^{\perp}$:

$$\mathbf{P}'(\mathbf{I} - \mathbf{P}'^{+}\mathbf{P}')\boldsymbol{\omega} = (\mathbf{P}' - \mathbf{P}'\mathbf{P}'^{T}(\mathbf{P}'\mathbf{P}'^{T})^{-1}\mathbf{P}')\boldsymbol{\omega} = \mathbf{0}$$

which is indeed a zero vector.

Actually, the following equation always stands, as long as the of the 3×4 matrix $\mathbf{P}'$ is 3:

$$\mathbf{I} - \mathbf{P}'^{+}\mathbf{P}' = \mathbf{I} - \mathbf{P}'^{T}(\mathbf{P}'\mathbf{P}'^{T})^{-1}\mathbf{P}' = \frac{\mathbf{p}'^{\perp}\mathbf{p}'^{\perp T}}{\|\mathbf{p}'^{\perp}\|^2}. \tag{55}$$

The effect of matrix $\mathbf{I} - \mathbf{P}'^{+}\mathbf{P}'$ is to transform an arbitrary vector to a vector that is perpendicular to every row vector of $\mathbf{P}'$. If $\mathbf{P}'$ is of rank 3 (which is usually the case), then $\mathbf{p}^{\perp}$ is unique up to a scale factor.

Equation (51) is easily justified by projecting $\mathtt{M}'$ onto the image using Eq. (48), which indeed gives $\tilde{\mathbf{m}}'$. If we look closely at the equation, we can find that $\mathbf{p}'^{\perp}$ actually defines the optical center, which always

projects onto the origin, and $\mathbf{P'}^{+}\tilde{\mathbf{m}}'$ defines the direction of the optical ray corresponding to image point $\tilde{\mathbf{m}}'$. For a particular value s', Eq. (51) corresponds to a point on the optical ray defined by $\mathbf{m}'$.

Similarly, an image point $\tilde{\mathbf{m}}$ in the first image also defines an optical ray. Requiring the two rays to intersect in space means that a point $\tilde{\mathtt{M}}'$ corresponding to a particular s' in Eq. (51) must project onto the first image at $\tilde{\mathbf{m}}$. That is,

$$s\tilde{\mathbf{m}} = s'\mathbf{P}\mathbf{D}\mathbf{P'}^{+}\tilde{\mathbf{m}}' + \mathbf{P}\mathbf{D}\mathbf{p'}^{\perp}, \tag{56}$$

where $\mathbf{P}\mathbf{D}\mathbf{p'}^{\perp}$ is the $\mathbf{e}$ in the first image.

Performing a cross product with $\mathbf{P}\mathbf{D}\mathbf{p'}^{\perp}$ yields

$$s(\mathbf{P}\mathbf{D}\mathbf{p'}^{\perp}) \times \tilde{\mathbf{m}} = (\mathbf{P}\mathbf{D}\mathbf{p'}^{\perp}) \times (s'\mathbf{P}\mathbf{D}\mathbf{P'}^{+}\tilde{\mathbf{m}}').$$

Eliminating s and s' by multiplying $\tilde{\mathbf{m}}^T$ from the left (equivalent to an inner product), we have

$$\tilde{\mathbf{m}}^T\mathbf{F}\tilde{\mathbf{m}}' = 0, \tag{57}$$

where

$$\mathbf{F} = \left[\mathbf{P}\mathbf{D}\mathbf{p'}^{\perp}\right]_{\times}\mathbf{P}\mathbf{D}\mathbf{P'}^{+} \tag{58}$$

is the general form of fundamental matrix. It is evident that the roles that the two images play are symmetrical.

Note that Eq. (58) will be the essential matrix $\mathbf{E}$ if $\mathbf{P}$ and $\mathbf{P}'$ do not include the intrinsic parameters, that is, if we work with normalized cameras.

We can also include all the intrinsic and extrinsic parameters in the two projection matrices $\mathbf{P}$ and $\mathbf{P}'$, so that for a 3D point $\tilde{\mathtt{M}}'$ in a world coordinate system, we have

$$s\tilde{\mathbf{m}} = \mathbf{P}\tilde{\mathtt{M}}', \tag{59}$$
$$s'\tilde{\mathbf{m}}' = \mathbf{P}'\tilde{\mathtt{M}}'. \tag{60}$$

Similarly we get

$$s\tilde{\mathbf{m}} = s'\mathbf{P}\mathbf{P'}^{+}\tilde{\mathbf{m}}' + \mathbf{P}\mathbf{p'}^{\perp}, \tag{61}$$

The same line of reasoning will lead to the general form of epipolar equation

$$\tilde{\mathbf{m}}^T\mathbf{F}\tilde{\mathbf{m}}' = 0,$$

where

$$\mathbf{F} = \left[\mathbf{P}\mathbf{p'}^{\perp}\right]_{\times}\mathbf{P}\mathbf{P'}^{+}. \tag{62}$$

It can also be shown that this expression is equivalent to Eq. (35) for the full perspective projection (see next subsection), but it is more general. Indeed, Eq. (35) assumes that the 3×3 matrix $\mathbf{B}'$ is invertible, which is the case for full perspective projection but not for affine cameras, while Eq. (62) makes use of the pseudoinverse of the projection matrix, which is valid for both full perspective projection as well as affine cameras. Therefore the equation does not depend on any specific knowledge of projection model. Replacing the projection matrix in the equation by specific projection matrix for each specific projection model produces the epipolar equation for that specific projection model.

The Full Perspective Projection Case

Here we work with normalized cameras. Under the full perspective projection, the projection matrices for the two cameras are the same:

$$\mathbf{P}_p = \mathbf{P}'_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{63}$$

It is not difficult to obtain

$$\mathbf{P}'^{+}_p = \mathbf{P}^T_p,$$

and

$$\mathbf{p}'^{\perp}_p = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Now substituting the above equations for Eq. (58), we have obtained the essential matrix

$$\mathbf{E}_p = \left[\mathbf{P}_p\mathbf{D}\mathbf{p}'^{\perp}_p\right]_{\times}\mathbf{P}_p\mathbf{D}\mathbf{P}'^{+}_p = [\mathbf{t}]_{\times}\mathbf{R}, \tag{64}$$

which is exactly the same as what we derived in the last section.

For the full perspective projection, we can prove that Eq. (62) is equivalent to Eq. (35). From definitions, we have

$$\begin{bmatrix}\mathbf{B}' & \mathbf{b}'\end{bmatrix}\mathbf{p}'^{\perp}_p = \mathbf{0},$$

$$\begin{bmatrix}\mathbf{B}' & \mathbf{b}'\end{bmatrix}\mathbf{P}_p'^{+} = \mathbf{I}.$$

It is easy to show

$$\mathbf{P}\mathbf{p}_p'^{\perp} = \lambda(\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}'),$$
$$\mathbf{P}_p'^{+} = \begin{bmatrix}\mathbf{B}'^{-1} - \mathbf{B}'^{-1}\mathbf{b}'\mathbf{q}^T \\ \mathbf{q}^T\end{bmatrix},$$

where λ is a nonzero scalar and $\mathbf{q}$ is a nonzero arbitrary three vector. Substituting them for Eq. (62) yields

$$\begin{aligned}\mathbf{F} &= \lambda[\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}']_\times \left(\mathbf{B}\mathbf{B}'^{-1} - (\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}')\mathbf{q}^T\right) \\ &= \lambda[\mathbf{b} - \mathbf{B}\mathbf{B}'^{-1}\mathbf{b}']_\times \mathbf{B}\mathbf{B}'^{-1}, \end{aligned} \quad (65)$$

which completes the proof as $\mathbf{F}$ is defined up to a scale factor.

The reader is referred to [5, 6] for the epipolar geometry between affine cameras and for a general expression of the fundamental matrix for both perspective and affine cameras. The reader is referred to [7, 8] for various algorithms of determining essential matrix and fundamental matrix from point correspondences. The reader is referred to [9, 10] for a general treatment of geometry across multiple cameras.

References

1. Longuet-Higgins H (1981) A computer algorithm for reconstructing a scene from two projections. Nature 293:133–135
2. Zhang Z, Faugeras OD (1992) 3D dynamic scene analysis: a stereo based approach. Springer, Berlin/Heidelberg
3. Maybank S (1992) Theory of reconstruction from image motion. Springer, Berlin/New York
4. Faugeras O (1993) Three-dimensional computer vision: a geometric viewpoint. MIT, Cambridge
5. Xu G, Zhang Z (1996) Epipolar geometry in stereo, motion and object recognition. Kluwer Academic, Dordrecht/Boston
6. Zhang Z, Xu G (1998) A unified theory of uncalibrated stereo for both perspective and affine cameras. J Math Imaging Vis 9:213–229
7. Zhang Z (1997) Motion and structure from two perspective views: from essential parameters to euclidean motion via fundamental matrix. J Opt Soc Am A 14(11):2938–2950
8. Zhang Z (1998) Determining the epipolar geometry and its uncertainty: a review. Int J Comput Vis 27(2):161–195
9. Hartley R, Zisserman A (2000) Multiple view geometry in computer vision. Cambridge University Press, Cambridge/New York
10. Faugeras O, Luong QT (2001) In: Papadopoulo T (ed) The geometry of multiple images. MIT, Cambridge/London

Error Concealment

▸Inpainting

Error-Correcting Graph Matching

▸Many-to-Many Graph Matching

Error-Tolerant Graph Matching

▸Many-to-Many Graph Matching

Essential Matrix

Zhengyou Zhang
Microsoft Research, Redmond, WA, USA

Related Concepts

▸Epipolar Geometry; ▸Fundamental Matrix

Definition

Essential matrix is a special 3×3 matrix which captures the geometric relationship between two calibrated cameras or between two locations of a single moving camera.

Background

See entry ▸Epipolar Geometry for details.

Theory

Because the cameras are calibrated, we use the normalized image coordinates. The two images of a space point $\boldsymbol{X} = [X, Y, Z]^T$ are $[x, y, 1]^T$ and $[x', y', 1]^T$ in the first and second images and are denoted by $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{x}}'$. Assume that the second camera is brought from the position of the first camera through a rotation $\mathbf{R}$

followed by a translation $\boldsymbol{t}$. From the epipolar geometry, we have the following equation

$$\widetilde{\boldsymbol{x}}^T \mathbf{E} \widetilde{\boldsymbol{x}}' = 0 \,, \tag{1}$$

where

$$\mathbf{E} = [\boldsymbol{t}]_\times \mathbf{R} \,. \tag{2}$$

And $[\boldsymbol{t}]_\times$ is a 3×3 antisymmetric matrix defined by vector $\boldsymbol{t}$. This equation is called the *epipolar equation*, and matrix $\mathbf{E}$ is known as the *essential matrix*.

Essential matrix has a number of properties, including:

1. $\det \mathbf{E} = 0$, because $[\boldsymbol{t}]_\times$ is an antisymmetric matrix.
2. $\mathbf{E}^T \boldsymbol{t} = \mathbf{0}$, because $([\boldsymbol{t}]_\times \mathbf{R})^T \boldsymbol{t} = \mathbf{R}^T [\boldsymbol{t}]_\times^T \boldsymbol{t} = -\mathbf{R}^T (\boldsymbol{t} \times \boldsymbol{t}) = \mathbf{0}$.
3. $\mathbf{E}\mathbf{E}^T = (\boldsymbol{t}^t \boldsymbol{t})\mathbf{I} - \boldsymbol{t}\boldsymbol{t}^T$, because $\mathbf{E}\mathbf{E}^T = ([\boldsymbol{t}]_\times \mathbf{R})([\boldsymbol{t}]_\times \mathbf{R})^T = [\boldsymbol{t}]_\times \mathbf{R}\mathbf{R}^T [\boldsymbol{t}]_\times^T = -[\boldsymbol{t}]_\times^2$.
4. $\|\mathbf{E}\|^2 = 2\|\boldsymbol{t}\|^2$. Here, $\|\mathbf{E}\|$ is the Frobenius norm of matrix $\mathbf{E}$, i.e., $\|\mathbf{E}\|^2 = \sum_{i,j} E_{ij}^2$.

It can be shown that a real 3×3 matrix can be decomposed into the multiplication of an antisymmetric matrix $\mathbf{T}$ and a rotation matrix $\mathbf{R}$ if and only if one of $\mathbf{E}$'s singular values is 0 while the other two are equal.

Euclidean Geometry

Gunnar Sparr
Centre for Mathematical Sciences, Lund University, Lund, Sweden

Related Concepts

▸Algebraic Curve; ▸Camera Calibration

Definition

Euclidean geometry deals with properties of geometric configurations that are preserved under isometric (or *length preserving*) transformations. Alternatively, it may be characterized as a mathematical theory based on an axiom system (that can be traced back to Euclid) expressing, in modern terminology, incidence, order, congruence, continuity, and parallelity. Euclidean geometry is today a special case of many geometric theories (projective, affine, and Riemannian geometries, Hilbert spaces . . .).

Historical Background

Euclidean geometry has a long and glorious history (cf. [1–3]), having lived at the core of the development of science and culture since antiquity. It is today not an area of research per se, but still plays an important role in many contexts.

Euclidean geometry is one of the oldest manifestations of humans in science. The latter part of the word *geometry* originates from the Greek word *metri'a* for *measure*, and the subject developed in the antiquity as an empirical science for surveying. It was given a scientific formulation by the Greek mathematician Euclid in Alexandria, about 300 B.C. Starting from a small set of intuitively appealing *axioms*, in his monumental treatise, the *Elements*, he deduced a large number of *propositions* about geometrical figures (cf. [8]). In the Elements, plane geometry was presented in essentially the way it is today taught in secondary school. Also the solid geometry of three dimensions was addressed. Two other big contributors to ancient geometry, both active in the third century B.C., were Archimedes, with equations for, e.g., the circumference of the circle and the area of the sphere, and Apollonius, with investigations of conic sections.

For over 2,000 years, the attribute *Euclidean* was unnecessary, because no other kind of geometry was conceived of. Early, however, the fifth of Euclid's axioms, the *parallel axiom*, was met with challenge, and many unsuccessful efforts were made to deduce it from the other axioms. However, it took until the nineteenth-century before its independence was settled by the construction of consistent geometric models with other parallelity concepts. Some contributors to such non-Euclidean geometries were Gauss, Bolyai, and Lobachevsky.

By today's standard of rigor, the treatment of Euclid is not without objections, using some assumptions and concepts not explicitly accounted for. Beginning in the nineteenth-century, several categorical axiom systems were presented, e.g., by Hilbert in 1899 (cf. [6, 9]).

Also other geometric systems were discovered and developed, like affine geometry, beginning with Euler in the eighteenth-century, and projective geometry,

through Poncelet, von Staudt, a.o. in the nineteenth-century. While in affine geometry, parallelity plays a prominent role; in projective geometry, the concept does not exist, in that every pair of lines intersect.

In contrast to the then prevalent *synthetic* approach to geometry, based on geometric constructions, in the seventeenth century, Descartes introduced coordinate systems and founded the *analytic geometry*. Also de Moivre made significant contributions. The use of coordinates enabled the study of geometry by means of algebra and calculus. In this formalism, in a natural way, Euclidean geometry can be generalized to higher dimensions, by means of the notion of Euclidean space; see below.

From a more modern mathematical point of view, geometry in some space is the study of properties of configurations that are preserved under some group of one-to-one transformations of the space in question (cf. [1–3]). This viewpoint was first formulated by Klein 1872 in the so-called Erlanger program, cf. [6]. It has been the key to a fruitful interplay between geometry, algebra, analysis, and topology. For Euclidean geometry, the characterizing transformation group is the group of isometric transformations, reflecting, e.g., the crucial property of congruence, that two triangles are congruent if one of them can be moved rigidly onto the other one.

Theory and Applications

The topic of Euclidean geometry today has branched out in many directions. Below, we focus on a few of relevance to computer vision.

Synthetic Euclidean Geometry

The synthetic approach to Euclidean geometry (the one used by Euclid) starts from the basic concepts of *point, line, and surface*. The rules for interaction between these are described by *axioms*. Euclid used five axioms (cf. [8]):

1. Through any two points, there is exactly one line.
2. A finite line segment can be extended to a line.
3. A circle can be drawn with any center point and any radius.
4. All right angles are equal to another.
5. The parallel postulate: If a straight line falling on two straight lines make the interior angles on the same side less than two right angles, the two straight lines, if produced indefinitely, meet on that side on which are the angles less than the two right angles.

For a rigorous exposition of Euclidean geometry, see [9], built on Hilbert's system of 20 axioms in five main groups: combination, order, parallelity, congruence, and continuity.

From the beginning, Euclidean geometry mostly dealt with squares and rectangles, right-angled triangles, trapezia, and circles. Some well-known examples of results from Euclidean geometry are the theorem of Pythagoras, the theorem stating that the sum of angles in a triangle equals two right angles, and the theorem stating that the periphery angle corresponding to an arch of a circle equals half the angle at the center.

The proofs in this tradition were constructive. Questions were raised about the possibility of generally solving geometric problems by ruler and compass. Using advances in algebra, the inherent impossibility of such constructions for some famed problems was proved, e.g., the trisectioning of an angle, the doubling of the cube, and the squaring of a circle.

Analytic Geometry: The Space $\mathbf{R}^n$

In the analytic geometry, as learned at school, points in the plane are represented by coordinates as (x_1, x_2), and points in space as (x_1, x_2, x_3). This inspires to consider n-tuples of real numbers $(x_1, x_2, \ldots, x_n)$, which together build up $\mathbf{R}^n$. This space forms the scene for geometry, not only in 1, 2, and 3 dimensions, but also in higher dimensions. The elements are called points $X : (x_1, x_2, \ldots, x_n)$.

The set $\mathbf{R}^n$ can be provided with different structures:

Linear Space. Equip $\mathbf{R}^n$ with the operations of addition, $\boldsymbol{x} + \boldsymbol{y} = (x_1, \ldots, x_n) + (y_1, \ldots, y_n) = (x_1 + y_1, \ldots, x_n + y_n)$, and multiplication with a scalar $\lambda \boldsymbol{x} = (\lambda x_1, \ldots, \lambda x_n)$. Then, in a way known from introductory courses in linear algebra, the notion of linear space is introduced. The elements of $\mathbf{R}^n$ are called *vectors*. The notion of *dimension* is defined, giving $\mathbf{R}^n$ the dimension n.

Affine Space. Combining the point and vector interpretations of $\mathbf{R}^n$, affine spaces are defined in a way such that, loosely speaking, it is possible to subtract points to get vectors and add a vector to a point to get another point. (On the contrary, it is not possible to add points.) For a thorough presentation,

see [1]. For affine spaces, the notation A^n will be used below.

Euclidean Space. Known from linear algebra is also the notion of *scalar product* on $\mathbf{R}^n$, being a function $\boldsymbol{x} \cdot \boldsymbol{y}$ such that for all vectors $\boldsymbol{x}$, $\boldsymbol{y}$, and all scalars λ:

- $\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{y} \cdot \boldsymbol{x}$.
- $\boldsymbol{x} \cdot (\lambda_1 \boldsymbol{y}_1 + \lambda_2 \boldsymbol{y}_2) = \lambda_1 \boldsymbol{x} \cdot \boldsymbol{y}_1 + \lambda_2 \boldsymbol{x} \cdot \boldsymbol{y}_2$.
- $\boldsymbol{x} \cdot \boldsymbol{x} \geq 0$ with equality if and only if $\boldsymbol{x} = 0$.

Given a scalar product, a *norm* on the linear space $\mathbf{R}^n$ is defined by $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}$. Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal if $\boldsymbol{x} \cdot \boldsymbol{y} = 0$. An orthonormal basis $\boldsymbol{e}_1, \ldots \boldsymbol{e}_n$ is characterized by $\boldsymbol{e}_i \cdot \boldsymbol{e}_j = 0$ for $i \neq j$ and $= 1$ for $i = j$.

Having a scalar product, the affine space A^n can be provided with a *metric*, by which the distance between points is given by $d(X, Y) = \|\boldsymbol{u}\|$, where $\boldsymbol{u}$ is the vector from X to Y. In particular, in an orthonormal basis, the distance between the points $X : (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ is

$$|XY| = ((x_1 - y_1)^2 + \ldots (x_n - y_n)^2)^{1/2}$$

(in agreement with the theorem of Pythagoras in the planar case).

The scalar product also makes it possible to define the *angle* θ between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ as

$$\boldsymbol{x} \cdot \boldsymbol{y} = \|\boldsymbol{x}\| \|\boldsymbol{y}\| \cos\theta.$$

Provided with a scalar product for vectors, A^n becomes a Euclidean space, below denoted E^n.

Euclidean Geometry and Transformation Groups

Given a set S, consider the group $\mathrm{Bij}(S)$ consisting of all one-to-one transformations $f : S \to S$. According to the view of the Erlanger program, to impose a geometry on S is the same as to specify a subgroup G of $\mathrm{Bij}(S)$ and to say that two subsets A, B of S are equivalent if there is an $f \in G$ such that $fA = B$. See [1–3].

Two important such transformation groups on $\mathbb{R}^n$ are $\mathrm{GL}(n)$, consisting of all non-singular $n \times n$-matrices, and $\mathrm{O}(n)$, consisting of all orthogonal $n \times n$-matrices.

For planar Euclidean geometry, S is the affine space A^2 and G is the group of all *isometric transformations*, i.e., transformations such that for any points X, $Y \in \mathbb{R}^2$, $d(TX, TY) = d(X, Y)$. In fact, it can be proven that every mapping T with this property can be written as

$$T : X \to QX + b \text{ , with } Q \in \mathrm{O}(n) \text{ .}$$

In particular, T is an affine map, mapping lines onto lines. Thus, it also maps triangles onto triangles, after which the isometric property guarantees congruence. In higher dimensions holds the analogous formula for T.

In Euclidean geometry, also the notion of *similarity* plays a prominent role, which could motivate the use of similarity transformations above (where Q is replaced by $cQ, c \neq 0$). Note that the isometric transformations form a subgroup of the similarity transformations.

The example of planar Euclidean geometry is so crucial that it is worthwhile to comment further on the structure of its transformation group. The formula above shows that an isometric transformation is composed by a rotation around the origin (expressed by Q), followed by a translation (by b). Furthermore, it is possible to prove that any such transformation is equal to either the identity, or a translation, or a rotation (around some point), or a reflection in some line, or a so-called glide reflection (a translation followed by a reflection in a line parallel to the translation). Also in $\mathbb{R}^3$, it is possible to make an analogous characterization, where a type screw (composition of a rotation and a translation) is added. Note in particular that in this way, isometries on $\mathbb{R}^2$ and $\mathbb{R}^3$ are characterized by geometric constructions only, without use of any particular coordinate system.

On the Stratum of Euclidean, Affine and Projective Geometry

Euclidean geometry is one of the layers in a hierarchy of structures that can be imposed on $\mathbb{R}^n$ and which are of high relevance to computer vision. With a term introduced by Faugeras [5], these form a *geometric stratum*. For more details on the respective geometries, see [1–3, 10].

Affine Geometry. For the linear space $\mathbf{R}^n$, a particular role is played by linear subspaces of the form $a_1 x_1 + \ldots + a_n x_n = 0$, called *hyperplanes* (always passing through the origin). As a linear space, its dimension is $n-1$. Analogously, in the affine space A^n, one considers *affine hyperplanes* $a_1 x_1 + \ldots +$

$a_n x_n = a$ (which for $a \neq 0$ do not pass through the origin). In the case $n = 2$, one talks about *lines* instead of hyperplanes.

For affine geometry on $\mathbf{R}^n$, the characterizing transformations are

$$T : X \to AX + b \text{ , with } A \in \mathrm{GL(n)} .$$

By such transformations, affine hyperplanes are mapped onto affine hyperplanes. Two nonintersecting hyperplanes are mapped onto two nonintersecting hyperplanes. Hence, the property of parallelity is preserved and is thus a concept within affine geometry. On the contrary, distance is not preserved and is an alien concept for affine geometry. However, it is possible to compare distances along a line (e.g., saying that M is the midpoint of a line segment PQ), but it is not possible to measure and compare distances on nonparallel lines. Neither does the concept of angle live in affine geometry, in lack of a scalar product.

Also affine geometry can be built up from axioms. Compared to Euclidean geometry, one way of doing this is by replacing the parallel axiom by an axiom "for any point P and any line ℓ, not through P, there is at least one line through P which does not meet ℓ," plus another one, involving seven-point configurations, named after Desargues.

Projective Geometry. By a point in the so-called projective space $\mathbb{P}^n$ is meant a line through the origin in $\mathbb{R}^{n+1}$. In other words, a point is represented by any of the vectors $\lambda(x_1, x_2, \ldots x_{n+1})$, $\lambda \neq 0$. This is called *homogeneous coordinates* for the point in $\mathbb{P}^n$. By a *projective transformation* T is meant a mapping represented by $\mathrm{GL}(n + 1)$ in homogeneous coordinates, i.e., $y = \lambda T x$ for some λ.

Looking in particular at plane projective geometry, where points are represented by lines through the origin in $\mathbb{R}^3$, a *projective line* is represented by a plane through the origin in $\mathbb{R}^3$. Since every pair of such planes intersect in a line through the origin, i.e., a point in $\mathbb{P}^2$, the notion of parallelity does not exist in projective geometry.

The projective space $\mathbb{P}^n$, embedded in $\mathbb{R}^{n+1}$, can also be visualized by means of an affine hyperplane in $\mathbb{R}^{n+1}$, e.g., $\Pi : \; x_{n+1} = 1$. Every line not parallel to Π intersects Π in a unique point. Besides these, $\mathbb{P}^n$ contains points corresponding to lines parallel to Π. The latter points of $\mathbb{P}^n$ are called *points at infinity* and may be identified with directions in Π. The set of points at infinity is called the *line at infinity*. To sum up, the projective space $\mathbb{P}^n$ can be visualized by a model formed by an n-dimensional affine space extended with points at infinity.

Perspective Transformations: Multiple View Geometry

For computer vision, *perspective transformations* play a crucial role. Specializing to three dimensions, consider two affine 3-dimensional hyperplanes Π and Π' in $\mathbb{R}^4$. If C is a point outside Π and Π', a mapping $P : \Pi \longrightarrow \Pi'$ is defined by considering the intersections of lines through C with Π and Π'. Then C is called the *focal point* of the *perspective transformation* P. In particular, one notes that if Π and Π' is nonparallel, then point at infinity of Π is mapped onto *ordinary* points of Π' and vice versa. If Π and Π' are parallel, then points at infinity are mapped onto points at infinity, and the perspective transformation is an affine transformation.

To sum up, Euclidean geometry is a special case of affine geometry, as follows from the fact that the group of isometries is a subgroup of the affine group. Affine geometry is a special case of projective geometry, since the affine transformations form a subgroup of the projective ones, characterized by leaving the line at infinite invariant.

Of particular importance to computer vision is the case of perspective transformations from three dimensions to two. These are singular, i.e., not one to one, contrary to the ones discussed above. They may be visualized by letting C tend to the plane Π', which in the limit gives a mapping $\Pi \longrightarrow \Pi \cap \Pi'$, from three to two dimensions. Moreover, the ambient space $\mathbb{R}^4$ becomes superfluous and can be replaced by the three-dimensional affine space Π.

Multiple view geometry deals with the situation where a number of such two-dimensional perspective images are known of a common three-dimensional object (cf. [4, 7]). With no further information, reconstruction is possible up to projective transformations. Having more geometric structure available, e.g., Euclidean information on the image planes and focal points, more structure can be obtained for the reconstruction, ideally making it Euclidean. This situation is termed *camera calibration*; see [7].

Some Other Geometries Embracing Euclidean Geometry

Algebraic Geometry. Besides hyperplanes, crucial roles in the study of Euclidean (as well as projective and affine) geometry are played by *conics*. While hyperplanes have equations involving first-order polynomials, conics are described by second-order polynomials. In two dimensions, this leads to the analytic geometry of conic sections, representing ellipses, hyperbolas, and parabolas. Algebraic geometry is the generalization of this to higher dimensions and higher-order polynomials, where Euclidean geometry thus falls out as a special case.

Riemannian geometry. Riemann geometry lives on a so-called Riemannian manifold, which, loosely speaking, is a space constructed by deforming and patching together Euclidean spaces according to certain rules, guaranteeing, e.g., *smoothnes*. Such a space enjoys notions of distance and angle but behaves in a curved, non-Euclidean manner. The simplest Riemannian manifold consists of $\mathbb{R}^n$ with a constant scalar product, leading to classical Euclidean geometry. Riemannian geometry plays a prominent role in general relativity.

Hilbert Spaces. Considering infinite sequences $(x_1, x_2, \ldots)$ instead of finite ones, $(x_1, x_2, \ldots, x_n)$, will lead to convergence problems whenever forming sums. Restricting oneself to sequences with $\sum_1^\infty x_i^2$ finite, a prototype of so-called Hilbert spaces is obtained. In a natural way, a scalar product is defined, leading to a distance measure, by means of which it is possible to prove, e.g., the infinite-dimensional analogue of the theorem of Pythagoras and many other theorems of Euclidean geometry.

References

1. Berger M (1987) Geometry I and II. Springer Universitext. Springer, Berlin/Heidelberg/New York/London/Paris/Tokyo
2. Coxeter HSM (1989) Introduction to geometry, 2nd edn. Wiley Classics Library, Wiley/New York
3. Eves H (1972) A survey of geometry. Allyn and Bacon, Boston
4. Faugeras O (1993) Three-dimensional computer vision. A geometric viewpoint. MIT, Cambridge/London
5. Faugeras O (1995) Stratification of three-dimensional vision: projective, affine, and metric representations. J Opt Soc Am A 12:465–484
6. Greenberg MJ (2008) Euclidean and non-Euclidean geomtries: Development and history. W.H. Freeman
7. Hartley R, Zisserman A (2003) Multiple view geometry in computer vision, 2nd edn. Cambridge University Press, Cambridge
8. Heath TL (1956) The thirteen books of Euclid's elements, 3 vols. Dover, New York
9. Hilbert D, Cohn-Vossen S (1952) Geometry and the imagination. Chelsea, New York
10. Veblen O, Young JW (1910–1918) Projective geometry I and II. Ginn and Co., Boston

Evolution of Robotic Heads

Michael R. M. Jenkin
Department of Computer Science and Engineering, York University, Toronto, ON, Canada

Related Concepts

▸Camera Calibration

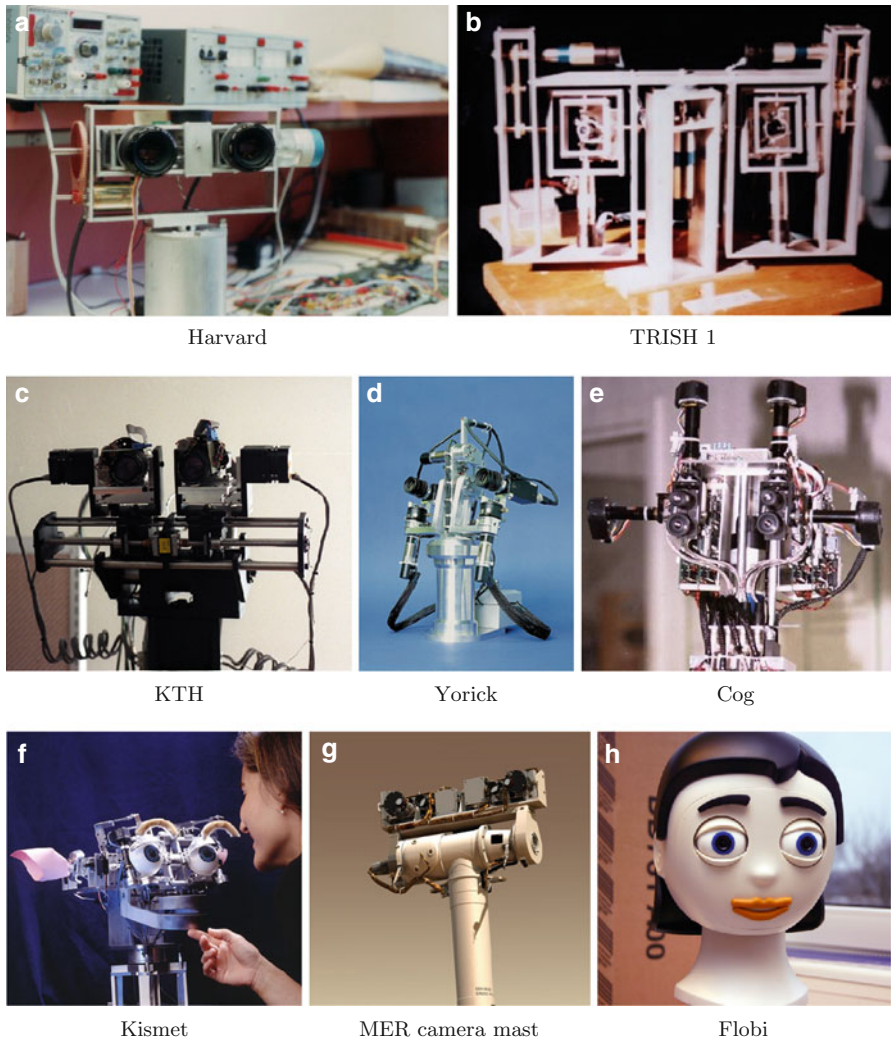
Definition

Robotic heads are actively controlled camera platforms, typically designed to mimic the head and camera (eye) motions associated with humans. Early designs were built to study the role of eye and head motion in active vision systems [1]. Later designs are also used in the study of human-robot interaction.

Background

Cameras have a finite field of view and thus must be actively controlled in order to bring out of view portions of the scene into view and to track dynamic scene events. The need for active control of camera geometry becomes even more critical when multiple cameras are involved as changes in relative camera geometry can simplify stereo image processing and can be used to bring specific scene features within the tuning range of multiple-camera scene reconstruction algorithms.

The development of single camera robotic heads can be traced back to the Stanford Cart [2]. Binocular systems began to appear in the late 1980s and early 1990s (e.g., [3–7]). By the late 1990s, head designs were beginning to be driven by research interest in human-robot interaction (e.g., [8, 9]). Current head designs are typically designed to be anthropomorphic



Evolution of Robotic Heads, Fig. 1 The evolution of robotic heads. (**a**) Appears with the kind permission of J. Clark (**b**) and (**c**) Appear courtesy of M. Jenkin. (**d**) Appears with the kind permission of P. Sharkey. (**e**) Appears courtesy NAS/JPL-Caltech. (**f**) Appears with the kind permission of Blefeld University

(e.g., [10]) or to meet specific requirements of the sensors or the application (e.g., [11]). See Fig. 1 for examples.

Theory

Although there is a wide range of different head designs, stereo robotic heads are perhaps the most common. Such systems either have fixed relative geometry between the two cameras or the relative geometry is controllable. Individual cameras may also be equipped with controllable intrinsic camera settings. Heads are typically mounted on pan and tilt "necks" that drive the entire head to look at different portions of the scene. Controllable parameters can include:

- *Head pan*. Pan angle introduced by a robotic neck.
- *Head tilt*. Tilt angle introduced by a robotic neck.
- *Baseline*. The displacement between the left and right cameras.
- *Fixation point*. The point at which the optical axes of the left and right cameras intersect. The fixation point can be defined in various ways including the individual pan directions of the left and right cameras and the vergence/version angles.

– *Cyclotorsion*. The roll of each camera about its optical axis.
– *Zoom/focus/aperture*. Intrinsic settings of each camera.

Being able to control the relative pose of the two cameras provides the sensor the ability to bring different portions of space into alignment so that objects appear at the same position in both cameras. As stereo image processing is typically performed only over a limited range of disparities (image differences), changing the relative geometry between the cameras leads to the camera system attending to different regions of space. The region of space that is brought into alignment is known as the horopter (see [12] for a review of the geometry).

Early designs were strongly constrained by the mass of available camera housings and especially the mass of controllable focus/zoom lenses. This is particularly evident in the early designs shown in Fig. 1. More recent designs have taken advantage of the decreased size and mass of camera systems.

Many modern robotic heads have been developed to help study aspects of human-robot interaction. Such "social robots" are designed to not only use multiple sensors to reconstruct 3D environments but they often also provide a range of actuators to simulate human facial responses.

References

1. Ballard DH (1991) Animate vision. Artif Intell J 48:57–86
2. Moravec HP (1983) The Stanford Cart and the CMU rover. Proc IEEE 71:872–884
3. Krotkov E (1989) Active computer vision by cooperative focus and stereo. Springer, New York
4. Ferrier NJ, Clark JJ (1993) The Harvard binocular head. Int J Pattern Recognit Artif Intell 7(1):9–32
5. Milios E, Jenkin M, Tsotsos J (1993) Design and performance of TRISH, a binocular robot head with torsional eye movements. Int J Pattern Recognit Artif Intell 7(1): 51–68
6. Madden BC, von Seelen U (1995) PennEyes: a binocular active vision system. Technical report MS-CIS-95-37, University of Pennsylvania
7. Sharkey PM, Murray DW, McLauchlan PF, Brooker JP (1998) Hardware development of the Yorick series of active vision systems. Microprocess Microsyst 21:363–375
8. Scassellati B (1998) Eye finding via face detection for a foveated, active vision system. In: Proceedings of the AAAI-98, Madison, pp 969–976
9. Breazeal C, Scassellati B (1999) How to build robots that make friends and influence people. In: IEEE/RSJ international conference on robots and autonomous systems, Kyongju
10. Lütkebohle I, Hegel F, Schulz S, Hackel M, Wrede B, Wachsmuth S, Sagerer G (2010) The Bielefeld anthropomorphic robot head "Flobi". In: IEEE international conference on robotics and automation, Anchorage
11. Goldberg SB, Maimone MW, Matthies L (2002) Stereo vision and rover navigation software for planetary exploration. In: 2002 aerospace conference, Big Sky
12. Hansard M, Horaud R (2008) Cyclopean geometry of binocular vision. J Opt Soc Am A 25(9):2357–2369

Expectation Maximization Algorithm

Boris Flach and Vaclav Hlavac
Department of Cybernetics, Czech Technical University in Prague, Faculty of Electrical Engineering, Prague 6, Czech Republic

Synonyms

EM-algorithm

Related Concepts

▸Maximum Likelihood Estimation

Definition

The Expectation Maximization algorithm iteratively maximizes the likelihood of a training sample with respect to unknown parameters of a probability model under the condition of missing information. The training sample is assumed to represent a set of independent realizations of a random variable defined on the underlying probability space.

Background

One of the main paradigms of statistical pattern recognition and Bayesian inference is to model the relation between the observable features $x \in \mathcal{X}$ of an object and its hidden state $y \in \mathcal{Y}$ by a joint probability measure $p(x, y)$. This probability measure is often known only up to some parameters $\theta \in \Theta$. It is thus necessary

to estimate these parameters from a training sample, which is assumed to represent a sequence of independent realizations of a random variable. If, ideally, these are realizations of pairs (x, y), then the corresponding estimation methods are addressed as *supervised* learning. It is, however, quite common that some of those variables describing the hidden state are latent. These latent variables are never observed in the training data. Therefore, it is necessary to marginalize over them in order to estimate the unknown parameters θ. Corresponding estimation methods are known as *unsupervised* learning. Moreover, especially in computer vision, the observation x *and* the hidden state y both may have a complex structure. The latter can be, e.g., a segmentation, a depth map, or a similar object. Consequently, it is often not feasible to provide the complete information y for the realizations in the sample. This means to estimate the parameters in the situation of missing information. The EM algorithm is a method searching for maximum likelihood estimates of the unknown parameters under such conditions.

Theory

All the situations described in the previous section can be treated in a uniform way by assuming the training sample as a set of independent realizations of a random variable.

Let Ω be a finite sample space, $\mathcal{F}$ be its power set, and $p_\theta : \mathcal{F} \to \mathbb{R}_+$ be a probability measure defined up to unknown parameters $\theta \in \Theta$. Let $X : \Omega \to \mathcal{X}$ be a random variable and $T = (x_1, x_2, \ldots, x_n)$ be a sequence of independent realizations of X (see, e.g., [1, 2] for a formal definition of independent realizations). The maximum likelihood estimator provides estimates of the unknown parameters θ by maximizing the probability of T:

$$\theta^* = \operatorname*{argmax}_\theta \prod_{i=1}^{n} p_\theta(\Omega_i), \tag{1}$$

where Ω_i denotes the pre-image $\{\omega \in \Omega \mid X(\omega) = x_i\}$. If the logarithm is taken, the task reads equivalently:

$$\theta^* = \operatorname*{argmax}_\theta L(x_1, \ldots, x_n, \theta)$$

$$= \operatorname*{argmax}_\theta \sum_{i=1}^{n} \log \sum_{\omega \in \Omega_i} p_\theta(\omega). \tag{2}$$

Remark 1. It is often assumed that Ω is a Cartesian product $\Omega = \mathcal{X} \times \mathcal{Y}$ and that X simply projects onto the first component $X(x, y) = x$. Then the probability $p_\theta(\Omega_i) = \sum_{y \in \mathcal{Y}} p_\theta(x_i, y)$ is nothing but the marginalization over all possible y. This special case will be considered in an example below. □

The optimization task (Eq. 2) is often complicated and hardly solvable by standard optimization methods – either because the objective function is not concave or because θ represents a set of parameters of different natures. Suppose, however, that the task of parameter estimation is feasible if complete information, i.e., a set of realizations of $\omega \in \Omega$, is available. This applies in particular if the corresponding simpler objective function $\sum_i \log p_\theta(\omega_i)$ is concave with respect to θ or if the task decomposes into simpler, independent optimization tasks with respect to individual components of a parameter collection.

The key idea of the Expectation Maximization algorithm is to exploit this circumstance and to solve the optimization task (Eq. 2) by iterating the following two feasible tasks:

1. Given a current estimate of θ, determine the missing information, i.e., $p_\theta(\omega|\Omega_i)$, for each element $x_i \in T$.
2. Given the complete information, solve the corresponding estimation task, resulting in an improved estimate of θ.

To further substantiate this idea of "iterative splitting" of task (Eq. 2), it is convenient to introduce nonnegative auxiliary variables $\alpha_i(\omega)$, $\omega \in \Omega_i$, for each element x_i of the learning sample T such that they fulfil:

$$\sum_{\omega \in \Omega_i} \alpha_i(\omega) = 1, \quad \forall i = 1, 2, \ldots, n. \tag{3}$$

These variables α_i can be seen as (so far arbitrary) posterior probabilities $p(\omega|\Omega_i)$ for $\omega \in \Omega_i$, given a realisation x_i. The log-likelihood of a realization x_i can be written by their use as:

$$\log p_\theta(\Omega_i) = \sum_{\omega \in \Omega_i} \alpha_i(\omega) \log p_\theta(\Omega_i)$$

$$= \sum_{\omega\in\Omega_i} \alpha_i(\omega) \log p_\theta(\omega) - \sum_{\omega\in\Omega_i} \alpha_i(\omega) \log \frac{p_\theta(\omega)}{p_\theta(\Omega_i)}, \quad (4)$$

where the first equality follows directly from (Eq. 3). The log-likelihood of the training sample can be therefore expressed equivalently as:

$$\begin{aligned} L(x_1,\ldots,x_n,\theta) &= \sum_{i=1}^{n} \sum_{\omega\in\Omega_i} \alpha_i(\omega) \log p_\theta(\Omega_i) \\ &= \sum_{i=1}^{n} \sum_{\omega\in\Omega_i} \alpha_i(\omega) \log p_\theta(\omega) \\ &\quad - \sum_{i=1}^{n} \sum_{\omega\in\Omega_i} \alpha_i(\omega) \log p_\theta(\omega|\Omega_i). \end{aligned} \quad (5)$$

The expression as a whole does not depend on the specific choice of the auxiliary variables α, whereas the minuend and subtrahend do. Moreover, note that the minuend is nothing but the likelihood of a sample of complete data, if the α are interpreted as the missing information, i.e., posterior probabilities for $\omega \in \Omega_i$ given the observation x_i.

Starting with some reasonable choice for the initial $\theta^{(0)}$ the likelihood is iteratively increased by alternating the following two steps. The (E)xpectation step calculates new α such that whatever new θ will be chosen subsequently, the subtrahend will not increase. The (M)aximization step relies on this and maximizes the minuend only, avoiding to deal with the subtrahend:

$$\text{E-step} \quad \alpha_i^{(t)}(\omega) = p_{\theta^{(t)}}\big(\omega|\Omega_i\big) \quad (6)$$

$$\text{M-step} \quad \theta^{(t+1)} = \operatorname*{argmax}_\theta \sum_{i=1}^{n} \sum_{\omega\in\Omega_i} \alpha_i^{(t)}(\omega) \log p_\theta(\omega). \quad (7)$$

From the conceptual point of view the E-step can be seen as inference – it calculates the missing data, i.e., the posterior probabilities $p_{\theta^{(t)}}\big(\omega|\Omega_i\big)$ for each element x_i in the training sample. The M-step utilizes these posterior probabilities for a supervised learning step. The names themselves stem from a rather formal view: the E-step calculates the α and therefore the objective function in (Eq. 7) which has the form of an expectation of $\log p_\theta(\omega)$. The computation of this objective function is sometimes considered to be a part of the E-step. The name for the M-step is obvious.

It is easy to see that the likelihood is monotonically increasing: The choice (Eq. 6) for α guarantees that the subtrahend in (Eq. 5) can only decrease whatever new θ will be chosen in the subsequent M-step. This follows from the inequality:

$$\sum_{\omega\in\Omega_i} p_\theta(\omega|\Omega_i) \log p_{\theta'}(\omega|\Omega_i) \leqslant \sum_{\omega\in\Omega_i} p_\theta(\omega|\Omega_i) \log p_\theta(\omega|\Omega_i) \;\; \forall\theta' \neq \theta. \quad (8)$$

Since the M-step chooses the new θ so as to maximize the minuend, the likelihood can only increase (or stay constant). Another convenient way to prove monotonicity of the EM algorithm can be found in [3, 4]. These tutorials consider the EM algorithm as the maximization of a lower bound of the likelihood.

It remains unclear whether the global optimum of the likelihood is reached in a fix-point of the algorithm. Moreover, it happens quite often that the M-step is infeasible for complex models p_θ. Then a weaker form of the EM algorithm is used by choosing $\theta^{(t+1)}$ so as to guarantee an increase of the objective function of the M-step.

The derivation of the concept of the EM algorithm was given here for a discrete probability space and discrete random variables. It can be however generalized for uncountable probability spaces and random variables X with continuous probability density.

Example 1. The EM algorithm is often considered for the following special case. The sampling space Ω is a Cartesian product $\Omega = \mathcal{X} \times \mathcal{Y}$ and the random variable X simply projects onto the first component $X(x, y) = x$. The parameters $\theta \in \Theta$ of the probability $p_\theta(x, y)$ are to be estimated given a sequence of independent realizations of x. In this special case, the log-likelihood has the form:

$$L = \sum_{i=1}^{n} \log \sum_{y\in\mathcal{Y}} p_\theta(x_i, y). \quad (9)$$

Its decomposition (Eq. 5) is:

$$L = \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \alpha_i(y) \log p_\theta(x_i, y) - \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \alpha_i(y) \log p_\theta(y|x_i). \quad (10)$$

The EM algorithm itself then reads:

$$\text{E-step} \quad \alpha_i^{(t)}(y) = p_{\theta^{(t)}}(y|x_i) \quad (11)$$

$$\text{M-step} \quad \theta^{(t+1)} = \operatorname*{argmax}_\theta \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} \alpha_i^{(t)}(y) \log p_\theta(x_i, y). \quad (12)$$

History and Applications

The classic paper [5] is often cited as the first one introducing the EM algorithm in its general form. It should be noted, however, that the method was introduced and analyzed substantially earlier for a broad class of pattern recognition tasks in [6] and for exponential families in [7].

A comprehensive discussion of the EM algorithm can be found in [8] and in the context of pattern recognition in [9, 10]. Standard application examples are parameter estimation for mixtures of Gaussians [8] and the Mean Shift algorithm [11]. Another important application is parameter estimation for Hidden Markov Models. This model class is extensively used for automated speech recognition. The corresponding EM algorithm is known as Baum-Welch algorithm in this context [12]. Rather complex applications of the EM algorithm arise in the context of parameter estimation for Markov Random Fields [13].

References

1. Meintrup D, Schäffler S (2005) Stochastik. Springer, Berlin
2. Papoulis A (1990) Probability and Statistics. Prentice-Hall, Englewood Cliffs
3. Minka T (1998) Expectation-maximization as lower bound maximization. Tutorial, MIT, Cambridge, MA
4. Dellaert F (2002) The expectation maximization algorithm. Technical report GIT-GVU-02-20, Georgia Institute of Technology
5. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soci Ser B 39(1):1–38
6. Schlesinger MI (1968) The interaction of learning and self-organization in pattern recognition. Kibernetika 4(2):81–88
7. Sundberg R (1974) Maximum likelihood theory for incomplete data from an exponential family. Scand J Stat 1(2): 49–58
8. McLachlan GJ, Krishnan T (1997) The EM algorithm and extensions. Wiley, New York
9. Schlesinger MI, Hlavac V (2002) Ten lectures on statistical and structural pattern recognition. Kluwer Academic Publishers, Dordrecht
10. Bishop CM (2006) Pattern recognition and machine learning. Springer, New York
11. Cheng Y (1995) Mean shift, mode seeking, and clustering. IEEE Trans Pattern Anal Mach Intell 17(8):790–799
12. Jelinek F (1998) Statistical methods for speech recognition. MIT, Cambridge, MA
13. Li SZ (2009) Markov random field modeling in image analysis. 3rd edn. Advances in pattern recognition. Springer, London

Exploration: Simultaneous Localization and Mapping (SLAM)

Samunda Perera[1], Nick Barnes[2] and Alexander Zelinsky[3]
[1]Canberra Research Laboratory, NICTA, Canberra, Australia
[2]Australian National University, Canberra, Australia
[3]CSIRO, Information Sciences, Canberra, Australia

Synonyms

Concurrent mapping and localization (CML); Visual SLAM

Related Concepts

▸Structure-from-Motion (SfM)

Definition

Exploration refers to gathering data about an environment through sensors in order to discover its structure. A fundamental technique for exploration of an unknown environment is Simultaneous Localization and Mapping (SLAM). SLAM is a technique that supports the incremental building of a 3-D map representation of an environment while also using the same incremental map to accurately localize the observer in order to minimize errors in the map building.

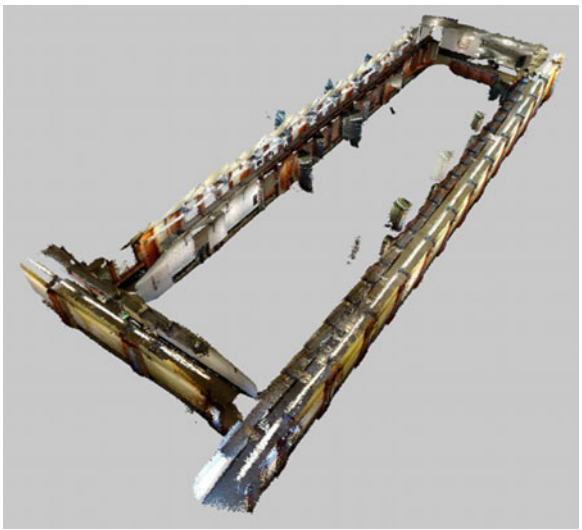
SLAM techniques can be implemented with laser range finders, monocular vision, stereo vision, and RGB-D cameras.

Background

An accurate representation of an environment is highly useful for autonomous and human assistive applications. Such a representation can be a high level minimalistic map derived from a sparse set of landmarks or a detailed dense 3-D model (e.g., Fig. 1). An accurate map can be used to perform useful tasks such as answering the question, "Where am I ?" and enables navigation and guiding tasks such as "How to get there?." When dealing with unknown environments, the map representation must be acquired through an exploration phase. With SLAM the map can be built through a data gathering or map building phase, by either online or off-line methods. In online methods, the map is incrementally built, and location within the map is simultaneously calculated. With off-line methods the data is collected and then post-processed to yield an accurate map representation.

The key process of SLAM is an optimization algorithm. Noisy observations are made about the landmarks from different viewpoints (poses), and the task is to solve the optimization problem of estimating the position of each landmark and the set of camera viewpoints which best explain the observations. In off-line mapping this is a global optimization problem where all observations are available (e.g., Bundle Adjustment (BA) [2] applied to a full image dataset). In online mapping this is performed sequentially, using observations available up to and including the current observation. Thus, in online mapping, an incremental version of the map is available as the map is being constructed.

Two different methods are used in online mapping, namely, filtering-based and keyframe-based methods. Filtering-based methods are based on the following nature of sequential mapping. As camera observations/measurements are noisy, there is induced error in the newly identified landmarks. Later when one localizes with respect to the current map, again there is induced error in this new pose estimate due to error in observation and error in these new landmarks. So in subsequent new landmark additions to the map, there will be error in landmarks due to both measurement errors and the camera viewpoint estimation error.



Exploration: Simultaneous Localization and Mapping (SLAM), Fig. 1 A 3-D map generated by RGB-D Mapping by Henry et al. [1] (Copyright © 2012 SAGE. Reprinted by Permission of SAGE)

Due to the common error associated with camera viewpoint, estimates of the landmarks are all necessarily correlated with each other [3]. Therefore a consistent full solution to the combined localization and mapping problem requires a joint state composed of the camera pose and every landmark position, to be updated following each landmark observation and hence the name Simultaneous Localization And Mapping.

In contrast keyframe-based methods do not propagate a probability distribution over time but employ a subset of image frames, called keyframes, in the map to perform efficient global optimization of the map and keyframe poses within a practical time limit. Although non keyframes are discarded in the optimization, it permits a large number of image observations per keyframe to be efficiently processed.

Visual SLAM is possible with both monocular and stereo cameras, and these systems are termed monoSLAM and stereoSLAM, respectively. In addition, RGB-D Mapping [1] which generates a dense 3-D map with RGB-D cameras has recently been introduced. It should be noted that, despite the computational effort, the use of cameras provides several unique advantages over other sensor types such as laser rangefinders and sonars. They are inexpensive, low power, compact, and capture scene information up to very large distances.

Theory

A complete SLAM system performs following operations in a cycle. First a set of new landmarks are identified in the environment, initialized (*new landmark/feature initialization*), and inserted into the map. Upon moving to a new location, *observations* are made and matched with the map landmarks, and this is referred to as *data association*. Here, *relocalization* methods are used to recover from possible tracking failures. Next, a prior estimate of the observer motion is obtained (*motion estimation*). Then the estimates of landmark positions and sensor pose that best explains the data are sought (*optimization*). If reobserving a previously mapped portion of the environment, care should be taken to ensure correct closure of the trajectory loop and consistency of the map (*loop closure*). Algorithm 1 illustrates this process (Note that as online mapping is an incremental process, Algorithm 1 assumes for generality some initial map is available. This may be the result of a single observation and new landmark initialization from the starting position).

Input: Initial set of landmarks and initial state of observer
Output: SLAM Map and observer state at each time instant
repeat
(1) Observation
(2) Data association
if tracking failure detected **then**
(3) Relocalization
(4) Motion estimation
(5) Optimization
if loop closure detected **then**
(6) Loop closure correction
(7) New landmark initialization

Algorithm 1: SLAM operation

An example situation is illustrated by Fig. 2. At time k, the true state of the observer is given by $\mathbf{x}_k$. The observer makes observations $\mathbf{z}_{k,i}, \mathbf{z}_{k,j}, \mathbf{z}_{k,p}$ of static landmarks whose true states are given by $\mathbf{m}_i, \mathbf{m}_j$, and $\mathbf{m}_p$, respectively. In the landmark selection process, it also discovers a new landmark whose true state is given by $\mathbf{m}_q$. Based on the corresponding measurement $\mathbf{z}_{k,q}$, the state of this new landmark is then initialized and inserted into the map. At time $k+1$, the observer has moved to a state $\mathbf{x}_{k+1}$ under the control input $\mathbf{u}_{k+1}$. The observer has been able to reobserve two existing landmarks in the map, namely, $\mathbf{m}_p$ and $\mathbf{m}_q$. Both the corresponding measurements $\mathbf{z}_{k+1,p}, \mathbf{z}_{k+1,q}$ and control input $\mathbf{u}_{k+1}$ can be utilized to obtain a prior estimate of the incremental motion of the observer between time k and $k+1$.

At time $k+n$, the observer has travelled along a long trajectory loop and is revisiting a previously mapped part of the environment, namely, landmarks $\mathbf{m}_i$ and $\mathbf{m}_j$. However, as can be seen from Fig. 2, the estimated state of the observer might not show a closed loop. Loop closure methods detect this situation and correct the state of the observer and the states of landmarks in the map.

The following subsections details the key stages of the SLAM system operation as given in Algorithm 1.

Observation

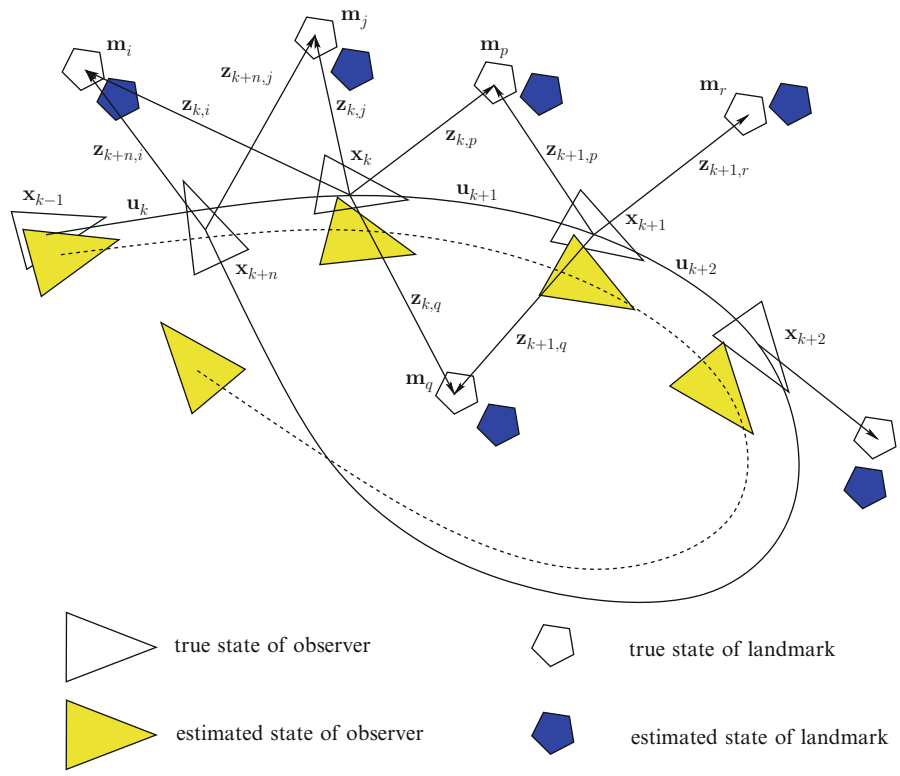
A new image frame is fetched from the camera and features (e.g., interest points, lines) which can be potential landmarks are extracted.

Data Association

Data association refers to finding correspondences between observations and landmarks. There are a number of methods in this regard. The active search/covariance-driven gating approach projects individual covariance predictions of landmarks into the observation image and limits the observation area based on suitable Mahalanobis distance. The Joint Compatibility Branch and Bound (JCBB) method [4] uses the fact that observation prediction errors are correlated with each other and hence calculates the joint probability of any set of correspondences. Active matching [5], on the other hand, considers the joint distribution and uses information theory to guide which landmarks to measure and when to measure them.

Relocalization

In case of rapid (unmodelled) motion, blur, and occlusion, camera pose tracking can fail and will result in subsequent corruption of the map. Relocalization refers to automatically detecting and recovering from such tracking failures to preserve the integrity of the map. Tracking failure can be detected by considering the percentage of unsuccessful observations and large uncertainty in the camera pose. The pose of the camera is recovered by establishing correspondence between the image and the map and solving the resulting perspective pose estimation problem [6].



Exploration: Simultaneous Localization and Mapping (SLAM), Fig. 2 SLAM problem

Motion Estimation

The observer's movement can be predicted based on odometry information or by making smooth motion assumptions (e.g., a constant velocity motion model). This information stands as a prior estimate of the observer state and covariance which is used later in the optimization stage. An example of such observer-state covariance computation is given by Eq. 5.

Optimization: Filtering-Based SLAM

Probabilistically the SLAM problem requires the computation of the joint posterior density of the landmark locations and observer (camera) state at time k, given the available observations and control inputs up to and including time k, together with the initial state of the observer [7–9].

This probability distribution can be stated as $P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0)$ where the quantities are defined as below (see Fig. 2).

$\mathbf{x}_k$ = observer state vector

$\mathbf{m}$ = vector describing the set of all landmark locations, i.e., landmark states $\mathbf{m}_i$

$\mathbf{Z}_{0:k}$ = set of all landmark observations $\mathbf{z}_{i,k}$

$\mathbf{U}_{0:k}$ = history of control inputs $\mathbf{u}_k$

Given an estimate for the problem at time $k-1$ (i.e., $P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1})$) a recursive solution to the problem can be obtained by employing a state transition model and an observation model which characterize the effect of the control input and observation, respectively.

Motion model: $P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$
Observation model: $P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$

Therefore, the SLAM algorithm can now be expressed as a standard two-step recursive prediction (time update) correction (measurement update) form as below:

Time Update

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \times P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0) d\mathbf{x}_{k-1} \quad (1)$$

Measurement Update

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \frac{P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{P(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \quad (2)$$

Two main solution methods to the probabilistic SLAM problem include Extended Kalman Filter (EKF)-based SLAM [3] and Rao-Blackwellized Filter (RBF)-based SLAM [10, 11].

Extended Kalman Filter (EKF)-Based SLAM

As noted, the solution to the SLAM problem requires a joint state composed of the observer state and the state of every landmark, to be updated following each landmark observation. This motivates to use the Kalman filter which provides a recursive solution to the discrete data linear filtering problem. When the motion model and measurement errors/noise are independent white Gaussian, the Kalman filter computes an optimal gain which minimizes mean-square error of the posterior estimate of the system. The Gaussian distribution is completely characterized by its mean and covariance. Here correlations between observer state and different landmarks can be properly represented in the total system covariance matrix of the Kalman filter. As the system would generally be nonlinear, the Extended Kalman Filter (EKF) is used.

Propagation of Uncertainty A fundamental problem in estimating covariance matrices is how to propagate uncertainty through a function. Consider a nonlinear function f which acts on the vector $\mathbf{a}$ of input variables to produce the output vector $\mathbf{b}$:

$$\mathbf{b} = f(\mathbf{a}) \quad (3)$$

Under affine approximation, the error covariance of the output vector $P_{\mathbf{bb}}$ is obtained in terms of the error covariance of the input vector $P_{\mathbf{aa}}$ by Eq. 4 (The covariance between two arbitrary vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as $P_{\mathbf{ab}} = E[(\mathbf{a} - E[\mathbf{a}])(\mathbf{b} - E[\mathbf{b}])^T]$):

$$P_{\mathbf{bb}} = \frac{\partial f}{\partial \mathbf{a}} P_{\mathbf{aa}} \frac{\partial f}{\partial \mathbf{a}}^T \quad (4)$$

State Prediction The general form of the EKF prediction steps can be customized for SLAM to yield faster computation. Note that there is no need to perform a time update for landmark states $\mathbf{m}_i : i = 1 \ldots n$ if the landmarks are stationary. Hence, only the observer's state, $\mathbf{x}_k$, is predicted ahead. The new error covariance of the observer's predicted state should also be computed. In addition, though there is no change in error covariance of landmarks, there will be change in error covariance between each landmark and observer state (since observer state changed). Given below are the new time update equations for the covariance items that require an update.

Notation:

$\{.\}^-$ = time update (prior) estimate
$\mathbf{n}$ = observer motion model noise
$P_{\mathbf{nn}}$ = observer motion model noise covariance
f_v = observer state transfer function $\mathbf{x}_{k+1} = f_v(\mathbf{x}_k, \mathbf{n})$ such that $\mathbf{x}^-_{k+1} = f_v(\mathbf{x}_k, \mathbf{0})$

$$P^-_{\mathbf{x}_{k+1}\mathbf{x}_{k+1}} = \frac{\partial f_v}{\partial \mathbf{x}_k} P_{\mathbf{x}_k\mathbf{x}_k} \frac{\partial f_v}{\partial \mathbf{x}_k}^T + \frac{\partial f_v}{\partial \mathbf{n}} P_{\mathbf{nn}} \frac{\partial f_v}{\partial \mathbf{n}}^T \quad (5)$$

$$P^-_{\mathbf{x}_{k+1}\mathbf{m}_i} = \frac{\partial f_v}{\partial \mathbf{x}_k} P_{\mathbf{x}_k\mathbf{m}_i} \quad (6)$$

The computational complexity of the EKF-SLAM solution grows quadratically with the number of landmarks. This is due to the calculations involved in the EKF update steps.

Rao-Blackwellized Filter (RBF)-Based SLAM

Rao-Blackwellized filter based SLAM methods include FastSLAM [10] and FastSLAM 2 [11] algorithms. The algorithms utilize a key point in SLAM, i.e., given the observer path/trajectory (if the observer path is assumed correct), different landmark measurements are independent of each other. So a particle filter is used to estimate the observer's path and for each particle a map is maintained. Since landmarks within this map are independent/uncorrelated, they can be represented with separate low-dimensional EKFs. Therefore, it has linear complexity rather than the quadratic complexity of EKF-SLAM.

Landmark Representation As noted, probabilistic SLAM (EKF/RBF SLAM) requires the uncertainty of landmarks to be modelled. A landmark's uncertainty

modelling depends upon the parameterization used to represent its state. The simplest way to parametrize a landmark is by the common three-dimensional Cartesian coordinates $\mathbf{m}_i^{car} = (X_i, Y_i, Z_i)^T$. This is applicable in stereo camera SLAM systems where the landmark state can be obtained by triangulation, and the uncertainty follows the common Gaussian assumption. However, in single camera SLAM systems, in which it is not possible to estimate a landmark's state from a single measurement, there is large uncertainty in the depth direction which cannot be modelled as Gaussian. Therefore, alternative parameterizations are proposed to alleviate this issue.

Civera et al. [12] represent a feature using inverse depth parameterization as

$$\mathbf{m}_i^{idp} = (x_i, y_i, z_i, \theta_i, \phi_i, \rho_i)^T, \tag{7}$$

where x_i, y_i, z_i represent the camera position from which the feature was first observed, θ_i, ϕ_i represent the azimuth and elevation of the corresponding ray, and ρ_i represents the inverse depth to the landmark along this ray. The relationship of $\mathbf{m}_i^{idp}$ to $\mathbf{m}_i^{car}$ is as follows:

$$\mathbf{m}_i^{car} = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i} \begin{pmatrix} \cos\phi_i \sin\theta_i \\ -\sin\phi_i \\ \cos\phi_i \cos\theta_i \end{pmatrix}. \tag{8}$$

Marzorati et al. [13] use the homogenous coordinates

$$\mathbf{m}_i^{isp} = (x_i, y_i, z_i, \omega_i)^T, \tag{9}$$

which relates with $\mathbf{m}_i^{car}$ as

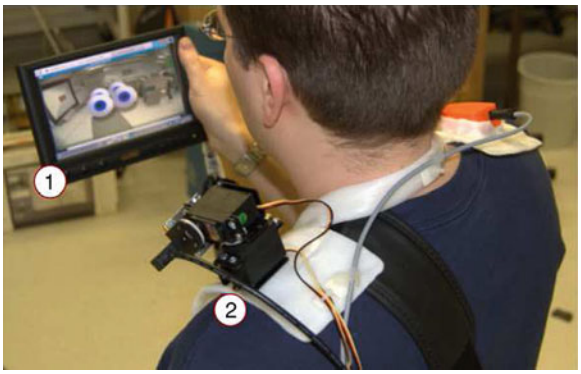
$$\mathbf{m}_i^{car} = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \frac{1}{\omega_i} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{10}$$

where ω_i is the inverse scale.

Although the number of parameters per landmark has increased from three of the minimal representation ($\mathbf{m}_i^{car}$), $\mathbf{m}_i^{idp}$ and $\mathbf{m}_i^{isp}$ are able to represent the landmark uncertainty as Gaussian and make the measurement equation more linear.

Optimization: Keyframe-Based SLAM

Keyframe-based SLAM splits camera pose tracking and mapping into separate processes. The accuracy of camera pose estimation is increased by using measurements from large numbers of landmarks per image frame. The map is usually estimated by performing bundle adjustment/optimization over only a set of frames (keyframes) since measurements from nearby image frames provides redundant and therefore less information for a given computational budget [14]. Here a new keyframe can be selected and inserted to the map based on a sliding window of most recent camera poses or by ensuring a minimum distance to the pose of the nearest keyframe already in the map.



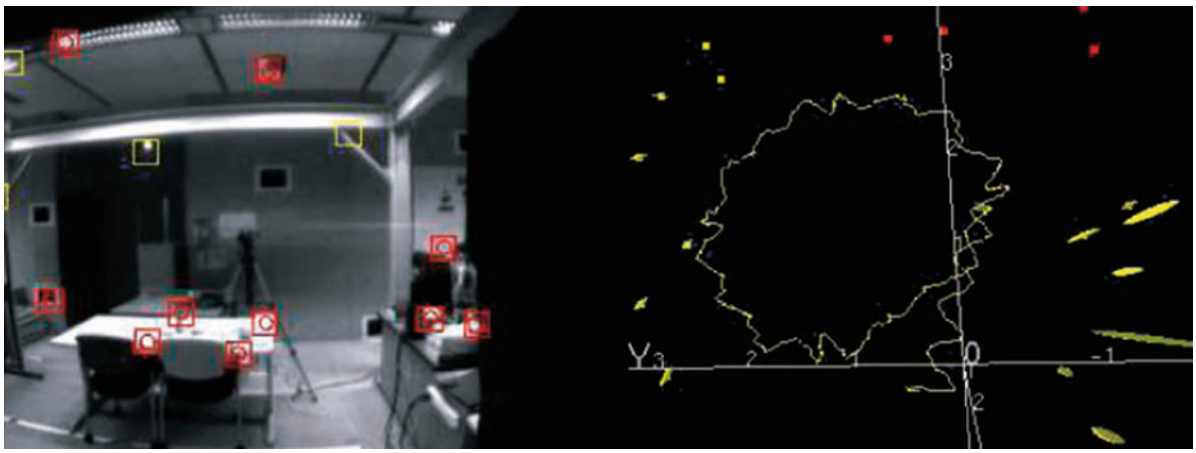
Exploration: Simultaneous Localization and Mapping (SLAM), Fig. 3 Wearable SLAM system for augmented reality by Castle et al. [18] – (1) handheld display with camera mounted on the back (2) active camera capable of pan and tilt (Reprinted from [18], with permission from Elsevier)

Loop Closure

Ideally when a SLAM system reobserves a previously mapped region of an environment, it should correctly recognize the corresponding landmarks in the map. However, particularly in long trajectory loops, landmark states in two such regions of the map can be incompatible given their uncertainty estimates. Therefore, loop closure methods are utilized to align such regions and make the map globally coherent. These include image-to-image matching [15], image-to-map matching [16], and hybrid methods [17].

New Landmark Initialization

Significant and distinguishable parts of the environment are selected as landmarks/features. In visual SLAM, these typically include interest points and edges/lines in the observed image. Upon selecting suitable landmarks, their position states and covariances are initialized and inserted into the map. Adding a new



Exploration: Simultaneous Localization and Mapping (SLAM), Fig. 4 A snapshot from MonoSLAM [20] as a humanoid robot walks in a circular trajectory. Here rectangular patches on the camera image (*left*) denote landmarks which are tracked interest points. In the 3-D map (*right*), *yellow* trace is the estimated robot trajectory and ellipsoids show landmark location uncertainties (Copyright © 2007 IEEE. All rights reserved. Reprinted, with permission, from [20])

landmark $\mathbf{m}_{new}$ to the map is a function of observer pose $\mathbf{x}_k$ and new landmark observation $\mathbf{z}$ which is given by $\mathbf{m}_{new} = f_{new}(\mathbf{x}_k, \mathbf{z})$. To add a new landmark, one needs to know the error covariance of the new landmark with other landmarks $\mathbf{m}_{oth}$ and the observer $\mathbf{x}_k$. This can be calculated using the system uncertainty at that time as below:

$$P_{\mathbf{m}_{new}\mathbf{m}_{new}} = \frac{\partial f_{new}}{\partial \mathbf{x}_k} P_{\mathbf{x}_k\mathbf{x}_k} \frac{\partial f_{new}}{\partial \mathbf{x}_k}^T + \frac{\partial f_{new}}{\partial \mathbf{z}} R \frac{\partial f_{new}}{\partial \mathbf{z}}^T \tag{11}$$

$$P_{\mathbf{x}_k\mathbf{m}_{new}} = P_{\mathbf{x}_k\mathbf{x}_k} \frac{\partial f_{new}}{\partial \mathbf{x}_k}^T \tag{12}$$

$$P_{\mathbf{m}_{oth}\mathbf{m}_{new}} = P_{\mathbf{m}_{oth}\mathbf{x}_k} \frac{\partial f_{new}}{\partial \mathbf{x}_k}^T \tag{13}$$

Note R denotes the error covariance of the observation $\mathbf{z}$.

Application

Visual SLAM approaches have been successfully used in large scale outdoor environment mappings, and due to the human-like visual sensing of the camera, it has found applications in augmented reality applications [18] (see Fig. 3).

Open Problems

Despite recent work [19], the performance of visual SLAM is not sufficiently robust in dynamic environments where multiple moving objects are present in the scene. This is similar to the motion segmentation problem in computer vision. In addition, mapping with RGB-D cameras [1] is an emerging field with the recent introduction of affordable RGB-D cameras.

Experimental Results

Figure 4 gives a snapshot of one of the first real-time MonoSLAM system in operation [20].

References

1. Henry P, Krainin M, Herbst E, Ren X, Fox D (2012) RGB-D mapping: using Kinect-style depth cameras for dense 3D modeling of indoor environments. Intern J Robot Res 31(5):647–663
2. Hartley RI, Zisserman A (2004) Multiple view geometry in computer vision, 2nd edn. Cambridge University Press, Cambridge. ISBN: 0521540518
3. Smith R, Self M, Cheeseman P (1990) Autonomous robot vehicles. In: Estimating uncertain spatial relationships in robotics. Springer, New York, pp 167–193
4. Neira J, Tardos JD (2001) Data association in stochastic mapping using the joint compatibility test. IEEE Trans Robot Autom 17(6):890–897

5. Chli M, Davison AJ (2008) Active matching. In: Proceedings of 10th European conference on computer vision (ECCV'08), Marseille
6. Williams B, Klein G, Reid I (2007) Real-time SLAM relocalisation. In: Proceedings of IEEE 11th international conference on computer vision ICCV 2007, Rio de Janeiro, pp 1–8
7. Durrant-Whyte H, Bailey T (2006) Simultaneous localization and mapping: part i. IEEE Robot Autom Mag 13(2):99–110
8. Bailey T, Durrant-Whyte H (2006) Simultaneous localization and mapping (SLAM): part ii. IEEE Robot Autom Mag 13(3):108–117
9. Thrun S, Burgard W, Fox D (2005) Probabilistic robotics. In: Intelligent robotics and autonomous agents. MIT, Cambridge
10. Montemerlo M, Thrun S, Koller D, Wegbreit B (2002) FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: Proceedings of the AAAI national conference on artificial intelligence, Edmonton. AAAI, pp 593–598
11. Montemerlo M, Thrun S, Koller D, Wegbreit B (2003) FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: Proceeding of the international conference on artificial intelligence (IJCAI), Acapulco, pp 1151–1156
12. Civera J, Davison AJ, Montiel J (2008) Inverse depth parametrization for monocular SLAM. IEEE Trans Robot 24(5):932–945
13. Marzorati D, Matteucci M, Migliore D, Sorrenti DG (2009) On the use of inverse scaling in monocular SLAM. In: Proceedings of IEEE international conference on robotics and automation ICRA'09, Kobe, pp 2030–2036
14. Strasdat H, Montiel JMM, Davison AJ (2010) Real-time monocular SLAM: why filter? In: Proceedings of IEEE international robotics and automation (ICRA) conference, Anchorage, pp 2657–2664
15. Cummins M, Newman P (2008) FAB-MAP: probabilistic localization and mapping in the space of appearance. Intern J Robot Res 27(6):647–665
16. Williams B, Cummins M, Neira J, Newman P, Reid I, Tardos J (2008) An image-to-map loop closing method for monocular SLAM. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems IROS 2008, Nice, pp 2053–2059
17. Eade E, Drummond T (2008) Unified loop closing and recovery for real time monocular SLAM. In: BMVC 2008, Leeds
18. Castle RO, Klein G, Murray DW (2010) Combining monoSLAM with object recognition for scene augmentation using a wearable camera. J Image Vis Comput 28(11): 1548–1556
19. Migliore D, Rigamonti R, Marzorati D, Matteucci M, Sorrenti DG (2009) Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments. In: Proceedings of international workshop on safe navigation in open and dynamic environments application to autonomous vehicles, Kobe
20. Davison AJ, Reid ID, Molton ND, Stasse O (2007) MonoSLAM: real-time single camera SLAM. IEEE Trans Pattern Anal Mach Intell 29(6):1052–1067

Extended Gaussian Image (EGI)

Sing Bing Kang[1] and Berthold K. P. Horn[2]
[1]Microsoft Research, Redmond, WA, USA
[2]Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA

Synonyms

Surface orientation histogram (discrete version of EGI)

Related Concepts

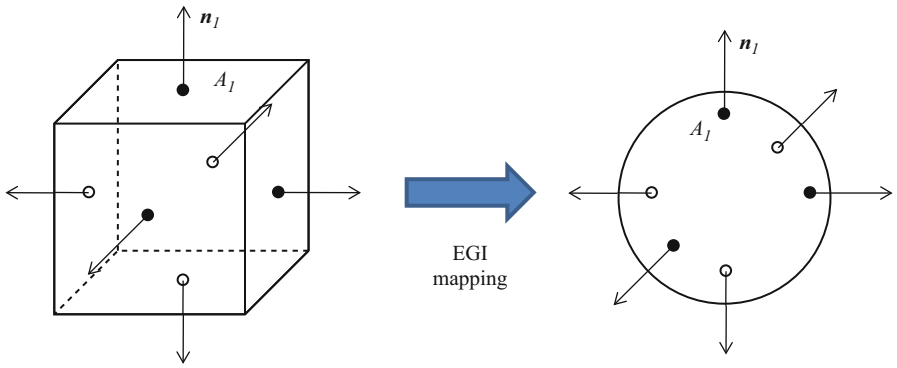
►Extended Gaussian Image (EGI)

Definition

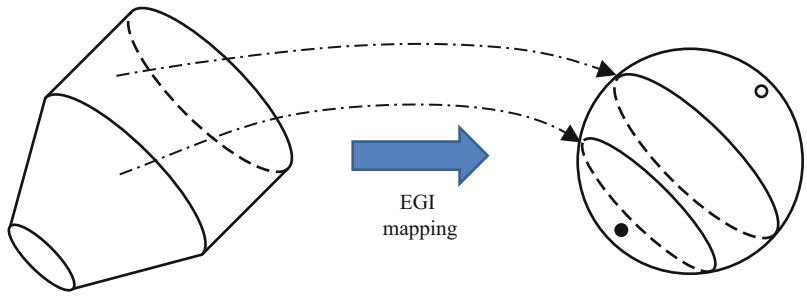
The extended Gaussian image (EGI) of a 3-D object is a function defined on a unit sphere. The value of the EGI at a point on the unit sphere is the inverse of the curvature at the corresponding point on the object – where the corresponding point is the one that has the same surface orientation. In the case of a polyhedral object, the value on the sphere is zero except for impulses at points on the sphere corresponding to faces of the polyhedron. The "size" or volume of each impulse is equal to the area of the corresponding face. The mapping from a 3-D polyhedron to the EGI is illustrated in Fig. 1. Figure 2 shows the mapping for a 3-D piecewise smooth object.

Background

Object recognition and determination of object pose (orientation and translation) are two fundamental, interlinked tasks in 3-D computer vision and robotics. The shape of an object can be measured directly using, for example, a rangefinder or binocular stereo techniques or indirectly using, say, photometric stereo. A representation for that shape is needed that makes recognition and attitude determination relatively easy. One requirement is that the representation transforms in a simple way when the object is rotated.



Extended Gaussian Image (EGI), Fig. 1 The EGI of a polyhedron. *Left*: cube with surface normals. *Right*: *arrows* on the unit sphere represent impulses corresponding to faces of the cube



Extended Gaussian Image (EGI), Fig. 2 The EGI of a piecewise smooth object. *Left*: piecewise smooth object. *Right*: the two flat ends map to two impulses while the conical surface maps to a *small circle*, and the cylindrical surface maps to a *large circle* on the sphere

Theory

EGIs can be used to represent both smoothly curved objects as well as polyhedra. For ease of explanation, the convex polyhedra are discussed first. Minkowski [11] showed in 1897 that a convex polyhedron is fully specified (up to translation) by the areas and orientations of its faces (see also [14] and [10]). The area and orientation of faces can be represented conveniently by point masses (i.e., impulses of mass density) on a sphere. The extended Gaussian image of a polyhedron is obtained by placing a mass equal to the area of each face at a point on the sphere with the same surface orientation as the face.

More specifically, the Gaussian image of an object is obtained by mapping from a point on the surface of the object to the point on the unit sphere that has the same surface normal. In the case of a convex object, this mapping is invertible – only one point on the object's surface corresponds to a point on the unit sphere. This mapping extends in a natural way to curves and surface patches. The Gaussian curvature is the limit of the ratio of the (signed) area of a patch on the unit sphere to the area of the corresponding patch on the surface of the object, as the size of the patches becomes smaller and smaller. The Gaussian curvature is everywhere nonnegative in the case of a convex object. The extended Gaussian image (EGI) associates a value with a point on the unit sphere equal to the inverse of the Gaussian curvature at the corresponding point on the object.

A.D. Alexandrov showed that a smoothly curved convex object is fully specified by curvature given as a function of surface orientation [1]. So the EGI representation is unique for both smoothly curved convex objects and convex polyhedra. Note, however, that neither of the proofs is constructive and thus they do not provide a basis for reconstructing an object from its EGI. Importantly, reconstruction is *not* required for recognition and orientation determination.

The discrete and continuous versions of the EGI can be related as follows: An object with planar faces can be thought of as the limit of an object with patches of spherical surfaces as the radius of the curvature of the patches get larger and larger. A spherical patch of radius R contributes a constant value R^2 (its Gaussian curvature) to an area of the unit sphere corresponding to all of the surface orientations at points in

that patch. As the radius tends to infinity, the value increases, at the same time that the area on the sphere shrinks. So the EGI of a polyhedron consists of a set of impulses, each of which has "volume" equal to the area of the corresponding face. It is convenient to treat these impulses as weights (or arrows of varying length) placed on the unit sphere. This is the discrete EGI. Conversely, the continuous EGI can be viewed as the limit of point mass density on the sphere as an approximation of a smoothly curved surface using a tessellation consisting of planar facets is made finer and finer.

The EGI has the following properties:

- *Translation invariance.* The EGI is not affected by object translation.
- *Rotation tracking.* Rotation of the object induces an equal rotation of the EGI since the unit surface normals rotate with the object.
- *Total mass equals total surface area.* Follows directly from the definition.
- *Center of mass at origin.* If the object is closed and bounded, the center of mass of the EGI is at the center of the Gaussian sphere.
- *Uniqueness.* There is only one closed convex object that corresponds to a particular EGI [1, 15].

If the object is not convex, more than one point on the object will contribute to a given point on the Gaussian sphere. One way of extending the definition of the EGI in the non-convex case is to use the sum of the inverses of the absolute values of the Gaussian curvature of all points on the surface that have the same orientation. While still useful in recognition and orientation determination, multiple non-convex objects may have the same EGI. For example, the EGI of all tori with the same surface area ($4\pi^2 R\rho$) and axis orientation is $2R\rho \sec \eta$, while there is but one *convex* object with that EGI. (Here R and ρ are the major axis and minor axis, respectively, while η is the angle between the surface normal and the plane perpendicular to the axis of the torus.)

In generating an EGI from an object, explicit information on shapes of faces and their adjacency relationships is not kept. Interestingly, a convex polyhedron can be recovered from its EGI [6, 9] as well as face adjacency and edge length information [12]. For a much more detailed treatment on EGIs, see [4]. The EGI has also been examined more broadly in the context of orientation-based representations [8].

The original version of the EGI is translation invariant. While it allows orientation to be extracted without regard to translation, a separate step is required to compute the translation. A variant of the EGI, the Complex EGI [7], uses complex mass to represent area (magnitude) and distance (phase). This decouples translation from rotation, allowing rotation to be determined in the usual fashion using the magnitudes while phase is subsequently used to compute translation. A support-function-based representation described in [13] encodes descriptors that are both local (tangent plane) and global (position and orientation of tangent plane).

Other extensions of the EGI address another inherent feature of not explicitly coding structural information. For example, the CSG-EESI (Constructive Solid Geometry-Enhanced Extended Spherical Image) [17] has two levels. The higher level contains the CSG tree which describes who the various subparts form the body, while the lower level describes the subparts using enhanced spherical images similar to that of [15]. Another tree-like representation is that of Hierarchical Extended Gaussian Image (HEGI) [18]. An HEGI description can be constructed as a tree where each leaf node corresponds to an Extended Gaussian Image (EGI) description of a convex part resulting from the recursive convex hull decomposition of an object. The COSMOS system described in [3] uses support functions based on Gaussian curvature and mean curvedness; connectivity information is maintained as a list.

Application

The EGI has been used for recognition and bin-picking of topmost objects (e.g., [2, 5, 6, 19]). It has also been used for symmetry detection [16] (for both reflectional and rotational symmetries).

In practice, the continuous EGI of a smoothly curved object is numerically approximated. The surface of the object may be divided into many patches, each of which is approximated by a planar facet of known area and orientation. The unit sphere may also be tessellated into cells, in each of which a total is accumulated of the areas of facets that have orientations falling within the range of orientations of that cell. Regular and semiregular tessellations of the sphere can be obtained by projecting regular

polyhedra (Platonic solids) or semiregular polyhedra (Archimedean solids). Unfortunately there are only 5 regular solids, with the icosahedron having 20 facets, which is not a fine enough tessellation. There are 13 semiregular solids, but again there are relatively few faces in these tessellations, the truncated icosahedron, for example, having only 32 faces. Finer, but less regular, tessellations can be obtained by subdividing faces of the regular and semiregular tessellations.

In typical practical applications, the object is only partially observed. Assuming the object is not occluded by other objects, the EGI of an observed convex object has information on at most one hemisphere. For a non-convex object, there is the issue of self-occlusion, so that the weight distribution for a set of tracked surface normal may change with object orientation. The most obvious solution is that of brute force search through the space of possible discrete orientations (e.g., as was done in [7]), but this is computationally expensive. Ikeuchi [6] reduces the search space by computing the ratio of the surface area to the projected area and constraining the freedom of rotation (since this ratio is independent of planar rotation for a given line of sight).

References

1. Alexandrov AD (1942) Existence and uniqueness of a convex surface with a given integral curvature. C R (Doklady) Acad Sci URSS (NS) 35(5):131–134
2. Brou P (1983) Finding objects in depth maps. PhD thesis, Department of Electrical Engineering and Computer Science, MIT
3. Dorai C, Jain AK (1997) COSMOS – a representation scheme for 3D free-form objects. PAMI 19(10):1115–1130
4. Horn BKP (1984) Extended Gaussian images. Proc IEEE 72(12):1671–1686
5. Horn BKP, Ikeuchi K (1984) The mechanical manipulation of randomly oriented parts (picking parts out of a bin of parts). Sci Am 251(2):100–111
6. Ikeuchi K (1981) Recognition of 3-D objects using the extended Gaussian image. In: IJCAI, Vancouver, pp 595–608
7. Kang SB, Ikeuchi K (1993) The complex EGI: a new representation for 3-D pose determination. PAMI 15(7): 707–721
8. Liang P, Taubes CH (1994) Orientation-based differential geometric representations for computer vision applications. PAMI 16(3):249–258
9. Little JJ (1983) An iterative method for reconstructing convex polyhedra from extended Gaussian images. In: Proceedings of the national conference on artificial intelligence, Washington, DC, pp 247–254
10. Lyusternik LA (1963) Convex figures and polyhedra. Dover, New York
11. Minkowski H (1897) Allgemeine Lehrsatze über die konvexen Polyeder. In: Nachrichten von der Königlichen Gesellschaft der Wissenschaften, mathematisch-physikalische Klasse, Göttingen, pp 198–219
12. Moni S (1990) A closed-form solution for the reconstruction of a convex polyhedron from its extended Gaussian image. In: ICPR, Atlantic City, NJ, vol 1, pp 223–226
13. Nalwa VS (1989) Representing oriented piecewise C^2 surfaces. IJCV 3(2):131–153
14. Pogorelov AV (1956) Differential geometry. Noordhoff, Groningen
15. Smith DA (1979) Using enhanced spherical images. Technical report, MIT AI Lab Memoo No. 530
16. Sun C, Sherrah J (1997) 3D symmetry detection using the extended Gaussian image. PAMI 19(2):164–168
17. Xie SE, Calvert TW (1988) CSG-EESI: a new solid representation scheme and a conversion expert system. PAMI 10(2):221–234
18. Xu JZ, Suk M, Ranka S (1996) Hierarchical EGI: a new method for object representation. In: 3rd international conference on signal processing, Beijing, vol 2
19. Yang HS, Kak AC (1986) Determination of the identification, position and orientation of the topmost object in a pile. CVGIP 36(2/3):229–255

Extrinsic Parameters

▸Camera Extrinsic Parameters