OBJECT RECOGNITION

Object recognition is a subproblem of the more general problem of perception, and can be defined as follows. Given a scene consisting of one or more objects, can we identify and localize those objects that are sufficiently visible to the sensory system? It is generally assumed that a description of each object to be recognized is available to the computer and can be used to facilitate the task of identification and localization. These descriptions can either be model-based or appearance-based, or a combination of both. Model-based object representation is based on geometric features, whereas appearance-based representation uses a large set of images for training but does not require any insight into the geometric structure of the objects. Object recognition is a key component of many intelligent vision systems, such as those used in hand-eye coordination for bin picking, inspection, and mobile robotics.

Various types of object recognition problems can be stated based on the dimensionality of their spatial description: (1) recognition of a 2-D object from a single 2-D image; (2) recognition of a 3-D object from a single 2-D image; (3) recognition of a 3-D object from a 3-D image (a range map); (4) recognition of a 2-D or 3-D object from multiple 2-D images taken from different viewpoints; and so on. About 40 years ago, research in computer vision began with attempts at solving the problem of how to recognize a general 3-D object using a single 2-D image. Since humans can perform this task effortlessly, it was believed then that designing a computer-based system for accomplishing the same would be easy. However, forty years later this problem remains largely unsolved. In contrast, much progress has been made in recognizing 2-D objects in single 2-D images and in recognizing 3-D objects in range maps. Although not as impressive, considerable progress has also been made in the recognition of 2-D or 3-D objects using multiple 2-D images, as in binocular or multiple-camera stereo.

The earliest successful system for the recognition of 2-D objects, such as gaskets used in industrial products, using single camera images was the VS-100 Vision Module fielded by SRI [1]. We believe that it was this system that launched industrial interest in computer vision. Another early industrial vision system that also became well known and that is of historical importance is the CONSIGHT system [2]. The HYPER system [3] was used for identifying overlapping flat electromechanical components, and used heuristic tree pruning to speed up the search for a scene-to-model match. Two early systems for the recognition of 3-D objects from single 2-D images are the ACRONYM system [4] and the SCERPO system [5], which used perceptual organization ideas to cope with the lack of depth information from a single image. Some studies on the errors associated with the recognition of 3-D objects from 2-D images include Refs. [6–8]. An exemplar on the more recent avenue of automatic learning of relevant features for object recognition is in Ref. [9].

One of the first successful systems that recognized 3-D objects in range maps was the 3DPO system for object orientation computation using graph matching [10]. Later contributions include the 3D-POLY system for object recognition in occluded environments [11], the INGEN system for generic object recognition [12], the MULTI-HASH system for fast 3D object recognition [13], and the BONSAI system for object recognition using constrained search [14]. Other relevant work on 3-D object recognition from range data includes Refs. [15–19]. Recent work on 3D object recognition from range images that also uses voting mechanisms on hash tables is in Ref. [20]; and one on the use of graph representations for model-based recognition is in Ref. [21].

Systems that have been demonstrated to recognize 3-D objects using principles of binocular stereo and other multicamera systems with various degrees of effectiveness include Refs. [22–24]. A variation on the idea of using multiple 2-D images for recognizing a 3-D object consists of projecting the image of an object into a space of lower dimensionality in order to facilitate the search for a match in a database of known object models. Examples of such systems include Refs. [25] and [26] for the recognition of human faces, and a real-time appearance-based recognition system that identifies 3-D objects [27]. These systems are sensitive to unrestricted illumination conditions and can only analyze scenes with one object at a time. A recent alternative to the use of PCAs for appearance-based recognition is with a novel sparse multiscale representation based on Gaussian differential basis functions, that simplify the image matching problem into a problem of polynomial evaluation [28].

Traditionally, a model-based recognition system includes the following sequence of tasks: sensory data acquisition, low-level processing of the sensory input, feature extraction, perceptual organization (e.g., grouping of features), scene-to-model hypothesis generation, and model matching. However, it is believed now that the interpretation of a complex scene cannot proceed in a purely bottom-up manner; instead, some of these tasks must cooperate with each other. For example, successful grouping of features could be guided by general constraints associated with the object classes. The recognition of a large database of objects cannot be efficiently achieved without the ability to represent an object in terms of its components, but there is no universally accepted formal definition of what constitutes a part and no general approach for decomposing an object into parts.

SENSORY DATA ACQUISITION

Light Intensity Images

For 2-D object recognition one snapshot of the scene to be analyzed is usually sufficient. In this case, the goal is to identify and locate one or more nearly flat objects in an image, often from a viewpoint that is perpendicular to the objects. An example is shown on Fig. 1(a). Here, the goal is to identify and accurately estimate the position and orientation of the keys. Gray-level or color digital images can be used for this purpose, and they can be captured with a digital camera or obtained by digitizing the signal of an analog camera using specialized hardware. When the objects in the scene have highly reflecting surfaces, the images obtained with a color or gray-scale camera may not be

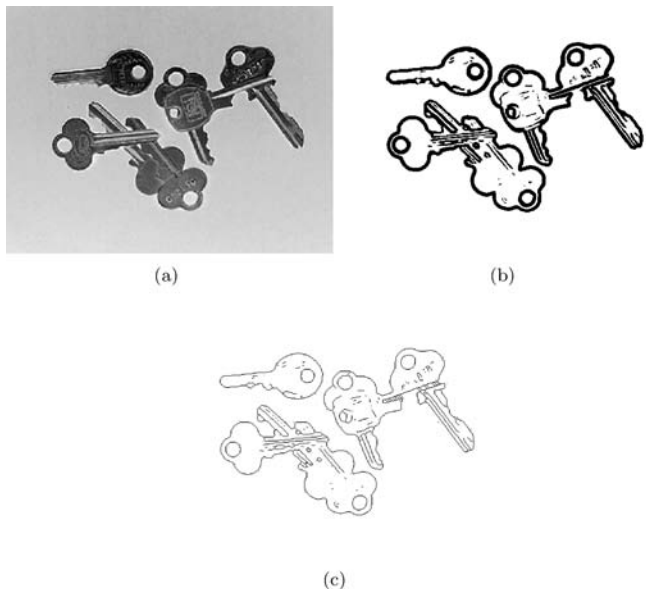


Figure 1. In 2-D object recognition the scene is usually observed from a viewpoint that is perpendicular to the objects. a) Scene image, b) output of Sobel operator for edge detection, c) one-pixel-wide edges obtained using morphological operators.

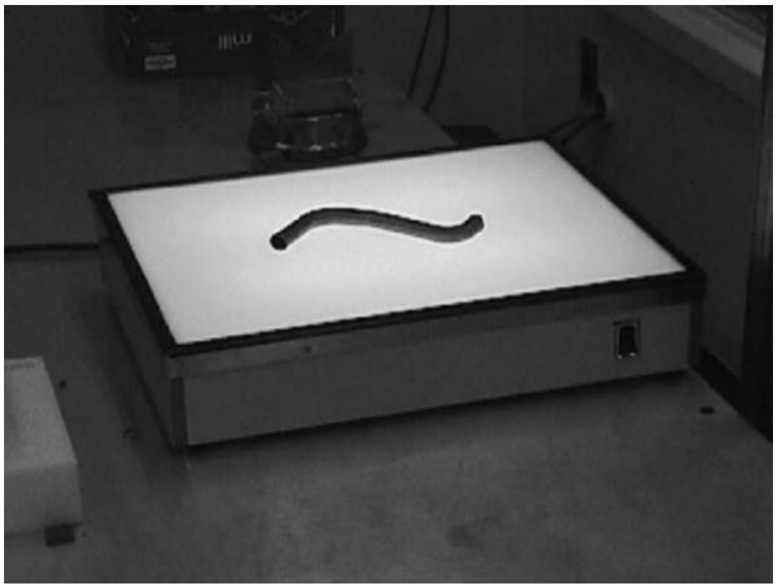


Figure 2. When the objects in a scene have highly reflecting surfaces, the images obtained with a color or gray-scale camera may not be acceptable. For these cases the use of a backlit table is more effective.

acceptable. For these cases the use of a backlit table will produce superior results. See Fig. 2. If high resolution is necessary, a high-density solid-state linear camera can be used. The objects are placed on a conveyor belt and scanned as the belt traverses linearly under the camera.

Various approaches are possible for acquiring data for 3-D vision. For example, with binocular stereo, two slightly shifted images of the same scene are taken and, when feasible, object points are located in 3-D space by triangu-lation. For stereo vision to work robustly, one must solve the correspondence problem: the matching of pixel pairs in the left and right images that correspond to the same point in 3-D space. See Fig. 3. Several geometric constraints can be used to alleviate the correspondence problem. One of these constraints is termed the feature constraint, and refers to the fact that what is probably an edge in the left image will most likely correspond to an edge in the right image also. Generally speaking, the characteristics of the neighboring pixels for the matching of the left and right image points should be consistent. For Lambertian (completely matte) surfaces, the reflected light is the same in all directions, and as a result, the intensities at two corresponding points in the left and right images should be the same. In practice, few surfaces tend to be purely Lambertian. By the same token, few surfaces tend to be completely glossy. In reality, for most surfaces the reflected light will vary slowly with the direction of viewing. Another restriction is the epipolar constraint which states that for any point in the left im-

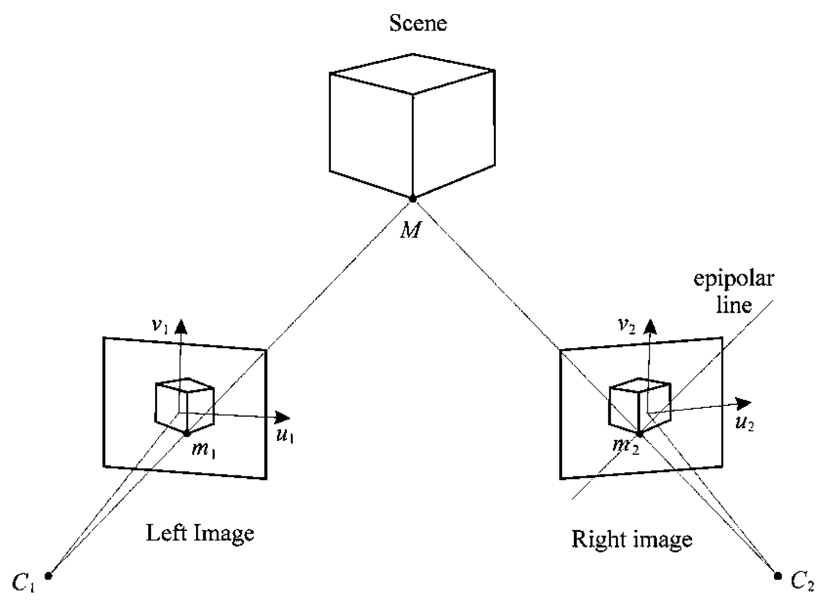


Figure 3. Correspondence problem in stereo vision: the points m_1 and m_2 in the left and right images correspond to the same point in 3-D space.

age its possible matches in the right image all lie on the epipolar line, therefore reducing the dimensionality of the search space from two dimensions to one. An epipolar line is the projection on (say) the right image of the line that passes through the center of projection and the pixel point on the left image. Other constraints include uniqueness, continuity, and ordering of points [22].

The use of multicamera stereo makes fuller use of the epipolar and other geometric constraints, thus simplifying the correspondence problem. Relative motion can also be used to determine 3-D location of points. If the objects in the scene are rigid and the configuration of the scene does not change while the camera is moving, the various images obtained can be treated as separated only in space and fixed in time. This process is referred to as baseline stereo. To benefit from the geometry of a baseline stereo system, the motion of the camera is generally made linear and perpendicular to its optical axis.

A robust approach to stereo vision in an industrial setting may use object-level knowledge in the stereo fusion process [29]. That is, the stereo system will use object model knowledge to extract from each of the images higher-level pixel groupings that correspond to discernible features on one or more surfaces of the object, and will try to solve the correspondence problem using these features. Because these features tend to be distinctive, the problem of contention in establishing correspondence is minimized.

Structured Light Sensing

A more robust approach for collecting 3-D data is by the use of structured light sensing in which a scene is illuminated with a single laser light stripe that scans across the scene. For each position of the light stripe, a camera registers the scene points illuminated by that stripe. The pixel coordinates of the illuminated points are subsequently converted into the *xyz* coordinates of the corresponding object points. For every row in the camera image, the column index of the brightest point represents the variation in depth for that particular scan line. The 3-D coordinates of every illuminated point are computed using appropriate triangulation formulas. Several configurations exist for structured-light imaging, the most common being obtained by single-slit projections and by bar code parallel projections. Furthermore, a single-slit projection system can be implemented using fan scanning, linear scanning, or fixed-camera linear scanning [30]. See Fig. 4.

In a single-slit fan-scanning projection system, a laser light is projected on a computer-controlled mirror and reflected onto the scene at different orientations. The scene is scanned step by step by the laser light stripe, and for each stripe, the position of the pixel with maximum brightness in each row on the camera image is obtained. The *xyz* coordinates of an object point p are computed from the intersection of the plane defined by the center of the mirror and the laser light stripe with the line defined by all the points in the space that corresponds to the point with coordinates u, v on the image plane. The equation of the plane illuminated by the laser light stripe varies with respect to θ, the mirror orientation angle.

In a linear scanning system, the mirror does not rotate. Instead, the laser light stripe projector and the camera are attached to a platform that moves at small intervals along a straight line perpendicular to the plane defined by the mirror and the laser light stripe. The coordinates of the illuminated points in the scene are computed in the same way as in the fan scanning projection system. The sensor, consisting of the projector and the camera, is usually located above the scene to be scanned, but it can also be attached to the gripper of a robotic arm. Linear scanning is ideal for integrating the range mapping process with manipulation. In a fixed-camera linear scanning projection system the camera is kept stationary while the slit projector is moved along an axis perpendicular to the plane illuminated by the laser light stripe.

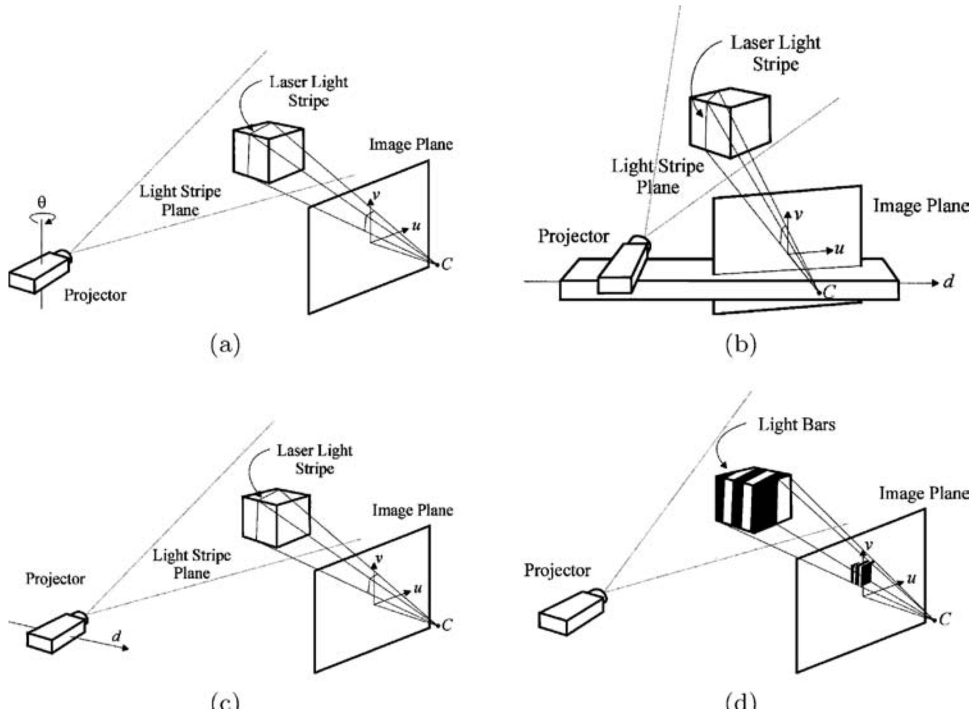


Figure 4. Structured light imaging configurations: a) single-slit fan scanning, b) single-slit linear scanning, c) single-slit fixed-camera linear scanning, and d) bar-code projection.

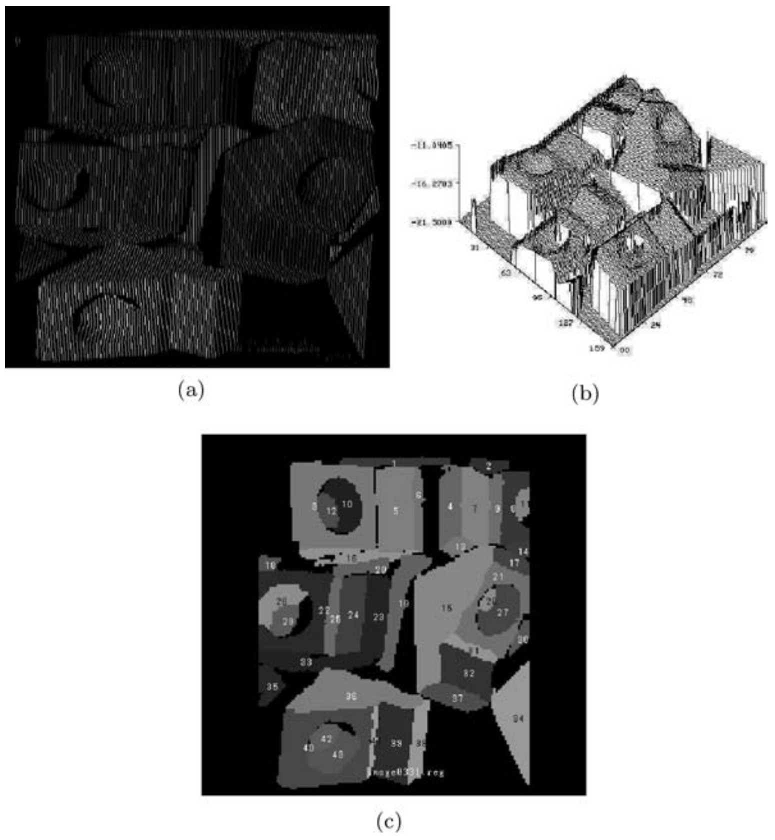


Figure 5. The set of images acquired in a structured-light system can be used to create a composite light stripe image (a). The computed *xyz* coordinates for the illuminated points along each stripe light are combined to produce a range map (b). Planar, cylindrical, and other second order surfaces (c) are extracted from the range map using low-level segmentation techniques.

The computed *xyz* coordinates for the illuminated points along each light stripe are combined to produce a range map. See Fig. 5(b). Structured-light scene reconstruction exhibits a few drawbacks. It is desirable for the laser light stripe to be the only illumination source for the scene. For this reason, structured-light projection systems cannot be used outdoors. Another disadvantage is the time it takes to scan a scene for acquiring its range map. Bar-code parallel projection can be used to alleviate this problem.

In a bar-code projection system, instead of illuminating the scene with a single laser light stripe, the scene is illuminated with a number of bar-code patterns like the ones shown in Fig. 6. Each stripe in the highest resolution bar-code pattern is equivalent to a single laser light stripe in a single-slit fixed-camera linear scanning projection system. The individual stripes corresponding to an equivalent single-slit scan are obtained by a simple decoding algorithm, which consists of examining for a point in the scene the on-off sequence of illuminations obtained for all the projected grid patterns, and then placing there a stripe corresponding to the resulting binary code word.

For N stripes in a single-slit system, $\log_2 N$ patterns are sufficient for the bar-code projection system, thus reducing the acquisition time by a logarithmic factor. The main disadvantage of the bar-code projection system is that when highly reflective surfaces are present in the scene, the camera may register those reflections, leading to errors in stripe decoding [30].

More novel structured-light sensors include the use of parabolic or elliptical mirrors. These sensors allow for dynamic reconfiguration of the triangulation geometry, and permit the acquisition of depth maps of a scene with varying levels of occlusion and depth resolution [31]. Color information can also be incorporated in a structured-light imaging system [13]. Color information is obtained by alternately illuminating a scene with a laser stripe and a white-light stripe, the latter being sampled at exactly those points that were illuminated by the laser stripe. While the laser stripe yields the *xyz* coordinates, the white-light stripe provides the color content at those points. Of course, one must use a color camera for such systems. Using such a system, one can not only recognize objects on the basis of their shapes, but also discriminate among similarly shaped objects on the basis of their color properties.

It is also possible to use line-of-sight laser sensors without cameras for acquiring 3-D information. A laser beam is transmitted toward the scene, and part of the light is reflected back the sensor a fraction of a second later. The sensor calculates the distance to the object point in the scene using the time of flight of the pulsed light. A rotating mirror deflects the light beam in such a way that the entire scene can be scanned in a raster fashion. The distance to an object point can also be computed by comparing the phase of a low-frequency power modulation of the outgoing and returning laser beams. A major advantage of such sensors is that they do not suffer from the occlusion problems that can sometimes reduce the effectiveness of structured-light sensors. For a structured-light sensor to work, an object point must be visible to both the illuminating source and the camera. On the other hand, for a line-of-sight sensor to work, it is sufficient for the object point to be visible to just the illuminating source, since no camera is involved. See Fig. 7.

LOW LEVEL PROCESSING

Low-level Processing of 2-D Intensity Images

The processing of 2-D intensity images for recognition of, say, flat objects usually begins with the detection or extraction of simple features characterized by singularities, such as edges or points. An edge can be defined as a discontinuity of intensity levels. Associated with every pixel at coordinates *u, v* in the image, there is an associated intensity value *I*. For gray-scale images, *I* is a scalar, whereas for color images, *I* will consist of three-color components. The detection of discontinuities in image intensity can be achieved mathematically by computing derivatives of the image intensity function *I(u,v)*.

The gradient of an image at each point is represented by the largest rate of intensity change at that point, and the direction of the gradient is along the direction of steepest change. Given an image *I(u, v)* the gradient is given by

$$\Delta I(u, v) = \hat{\mathrm{u}}\frac{\partial I}{\partial u} + \hat{\mathrm{v}}\frac{\partial I}{\partial v}$$

and its magnitude can be computed by the norm

$$\|\nabla I(u, v)\| = \sqrt{(\frac{\partial I}{\partial u})^2 + (\frac{\partial I}{\partial v})^2}$$

For digital implementations, the magnitude calculated by using the norm

$$\|\nabla I(u, v)\| = \max[\frac{\partial I}{\partial u}, \frac{\partial I}{\partial v}]$$

gives more uniform results with respect to the direction of the edge. The former is called the L_2 norm, whereas the latter is called the L_∞ form.

The direction of the gradient at the image point with coordinates (u, v) is given by

$$\tan^{-1}(\frac{\partial I}{\partial v}/\frac{\partial I}{\partial u})$$

and the direction of the edge at that point is perpendicular to that of the gradient. In order to find the derivatives of the image in the u and v directions, several operators can be used. See Fig. 8. The 2-D intensity function I has to be convolved with these operators, each of them possessing different characteristics regarding the sensitivity of edge detection to the direction of an edge. The most popular of these being the Sobel operator due to its smoothing property for noise reduction. The cumulative output of the convolution with the Sobel operators for the image in Fig. 1(a) is shown in Fig. 1(b). The Roberts operators on the contrary, not only do not provide smoothing, but are also of even size, with the consequence that their response cannot be assigned to a central point. It is important to realize that the sum of all entries in a filter kernel must add up to zero, indicating that for regions of homogeneous intensity values no features can be detected.

Image derivatives are sensitive to image noise, and it is sometimes desirable to first smooth out the noise prior

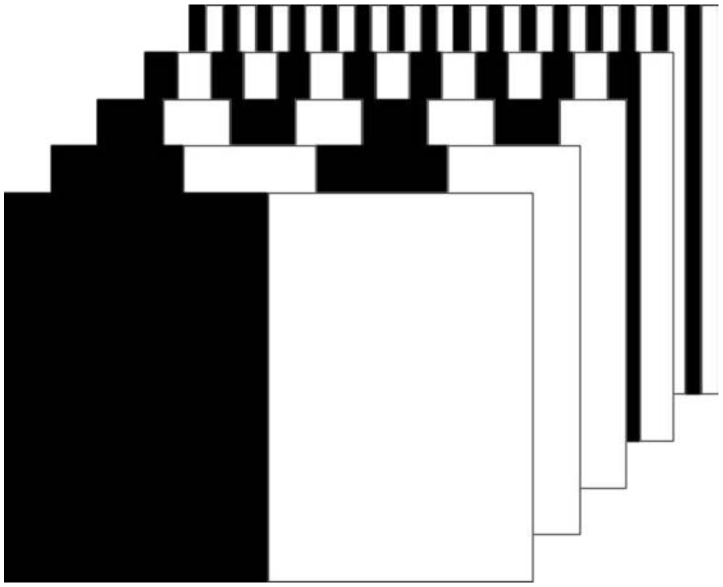


Figure 6. In a bar-code projection system the scene is illuminated with a number of bar-code patterns.

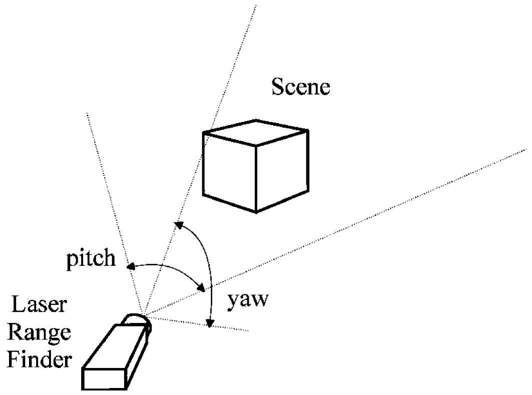


Figure 7. With a line-of-sight laser range finder, a laser beam is transmitted toward the scene, and part of the light is reflected to the sensor. The sensor computes the distance to the object point in the scene using the time of flight of the pulsed light.

$$\frac{\partial}{\partial u} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \frac{\partial}{\partial \theta_1} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \frac{\partial}{\partial u} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial}{\partial v} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \frac{\partial}{\partial \theta_2} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \frac{\partial}{\partial v} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel operators Roberts operators Prewitt operators

Figure 8. Several differentiation masks can be used to approximate the derivatives of an image.

to feature extraction. This can be done by smoothing or blurring by convolution with a Gaussian filter

$$\begin{aligned} I_G(u, v) &= I(u, v) * G(u, v) \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma^2} e^{-\frac{\alpha^2+\beta^2}{2\sigma^2}} I(u-\alpha, v-\beta) d\alpha\, d\beta \end{aligned}$$

where the σ parameter is the standard deviation or scale.

Given that differentiation commutes with convolution, $\partial(I * G) = \partial I * G = I * \partial G$, noise reduction can be obtained by differentiating not the image, but the Gaussian kernel itself. In this way, Gaussian derivative operators are suitable both to feature extraction and noise reduction at the same time. First order derivatives of Gaussian filters resemble local features such as edges or stripes, making them a good alternative to the filters in Fig. 8 for detecting such features in images from convolution. See Fig. 9. Other advantageous properties of Gaussian filters are steerability [32] and separability [33].

Steerability means that the filter can be computed at any orientation by a linear combination of basis filters [32]. The only constraints for a function to be steerable is that it

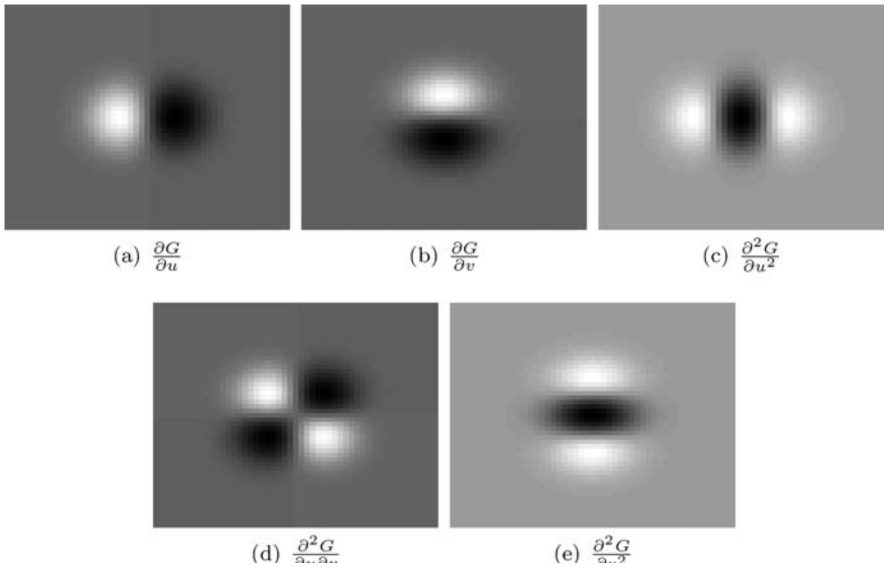

(a) $\frac{\partial G}{\partial u}$
(b) $\frac{\partial G}{\partial v}$
(c) $\frac{\partial^2 G}{\partial u^2}$
(d) $\frac{\partial^2 G}{}$
(e) $\frac{\partial^2 G}{}$

Figure 9. First and second order Gaussian derivatives

must be polar-separable, i.e., to be conformed as the product of a radial component and an angular component; and to be expressed as a linear combination of basis filters.

From the steerability property of Gaussian filters it is possible to detect features like edges or strips in images at any orientation. The first order Gaussian derivative is expressed in polar coordinates and then decomposed in polar series in order to assign the basis functions and their respective interpolation coefficients [33, 32]. The resulting steerable Gaussian filter is

$$\frac{\partial G}{\partial \theta} = \cos\theta \frac{\partial G}{\partial u} + \sin\theta \frac{\partial G}{\partial u}.$$

By the same procedure, the second order Gaussian derivative is

$$\frac{\partial^2 G}{\partial \theta^2} = \cos^2\theta \frac{\partial^2 G}{\partial u^2} + 2\cos\theta \sin\theta \frac{\partial^2 G}{\partial u \partial v} + \sin^2\theta \frac{\partial^2 G}{\partial v^2}$$

Fig. 10 illustrates first and second order Gaussian filters steered at an angle of $\frac{7\pi}{4}$, as well as the resulting extracted features.

Edges

Some edge detectors use second-derivative operations; the best known of these is the Laplacian of the Gaussian (LoG) edge detector [34]. In this method, the image intensity function I is smoothed with a Gaussian function and the edges are detected from the zero crossings of the second derivative

$$I_{\text{edge}}(u, v) = \Delta I_G(u, v)$$

where $\Delta = \frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2}$ is the direction-independent Laplacian operator. Recall that differentiation commutes with convolution, thus differentiation can be performed over the Laplacian operator and not over the image. The LoG is given by

$$I_{\text{edge}}(u, v) = I(u, v) * \Delta G(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(\alpha, \beta) \Delta G(u - \alpha, v - \beta) d\alpha\, d\beta$$

where

$$\Delta G(u, v) = \frac{u^2 + v^2 - 2\sigma^2}{2\pi\sigma^6} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

For edge detection, the zero crossing are taken instead of the LoG magnitude because the latter leads to double edges. As a consequence the LoG method gives one-pixel wide closed contours, but it can also produce spurious zero crossings caused by points of inflection in the first derivative, and produces biased contours in the vicinity of locations where actual edges form acute angles are present. Fig. 11 shows the LoG operator applied to the image in Fig. 10(a). Note that this filter is rotationally invariant.

Another widely used method for edge detection is the Canny edge detector. It belongs to a family of optimally designed operators based on the detection of extrema in the output of the convolution of the image with an impulse response (the operator). Other edge detectors in this family include the Deriche detector and the Spaceck detector [22].

In this method, a model of the kind of edges to be detected is defined first. Consider for the sake of simplicity the 1-D case,

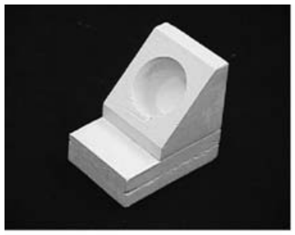
$$e(x) = AU(x)\eta(x)$$

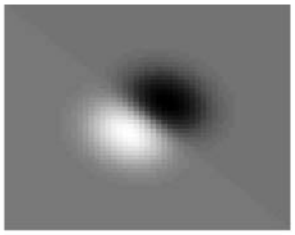
where $U(x)$ is the unit step function and $\eta(x)$ is white Gaussian noise. Then, several criteria that must be satisfied by the operator are derived, such as robustness to noise, good localization, and uniqueness of response.

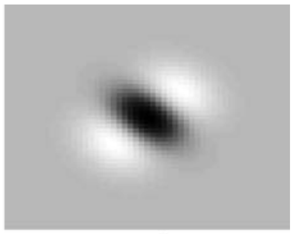
The output of our operator on an ideal edge would be

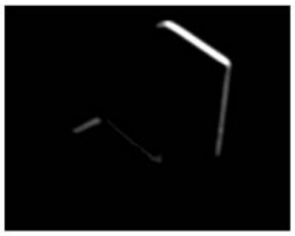
$$I_{\text{edge}}(x) = e(x) * h(x)$$

$$I_{\text{edge}}(x) = A \int_{-\infty}^{x} h(x) d\alpha + \int_{-\infty}^{\infty} \eta(x - \alpha) h(x) d\alpha$$


(a) Original image I


(b) $\frac{\partial G}{\partial \theta}$


(c) $\frac{\partial^2 G}{\partial \theta^2}$


(d) $I * \frac{\partial G}{\partial \theta}$

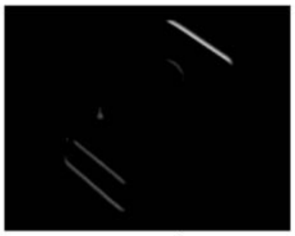
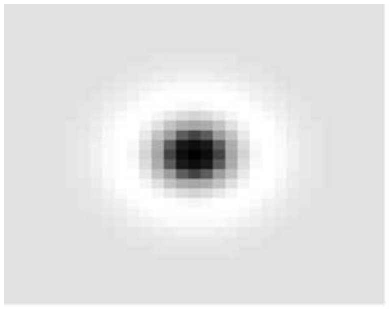
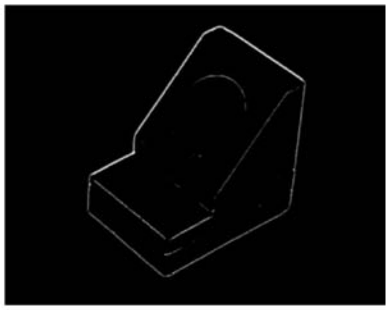

(e) $I * \frac{\partial^2 G}{\partial \theta^2}$

Figure 10. Feature extraction with a steered Gaussian filter.


(a) Laplacian filter


(b) LoG

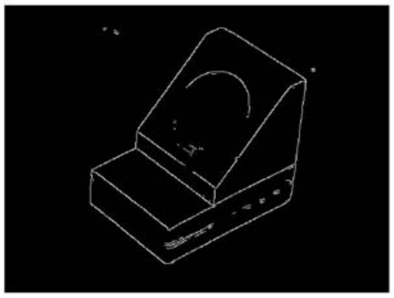

(c) Canny

Figure 11. Edge detection using the LoG and Canny operators.

and the idea is to maximize I_{edge} at $x = 0$, satisfying the following criteria: the robustness-to-noise criterion

$$\Sigma(h) = \frac{-\infty \int h(x)dx}{\sqrt{-\infty \int h^2(x)dx}}$$

the good-localization criterion

$$\Sigma(h) = \frac{\int_{-\infty}^{0} h(x)dx}{\sqrt{\int_{-\infty}^{0} h^2(x)dx}}$$

and the uniqueness-of-response criterion

$$x_{\max} = 2\pi \sqrt{\frac{\int_{-\infty}^{0} h'^2(x)dx}{\int_{-\infty}^{0} h''^2(x)dx}}$$

(For a detailed derivation of these criteria expressions see [22]). Using calculus of variations we can simultaneously find the extrema of $\Sigma(h)$, $\Lambda(h)$ and $x_{\max}$ by solving the constrained optimization problem

$$\min \int_{-\infty}^{0} h^2(x)dx$$

subject to

$$\int_{-\infty}^{0} h(x)dx = c_1,\ \int_{-\infty}^{0} h'^2(x)dx = c_2,\ \int_{-\infty}^{0} h''^2(x)dx = c_3.$$

In Canny's derivation, these criteria are satisfied with the extra constraint that $x_{\max} = kW$, which states that the average maximum distance between two local maxima has to be some fraction of the spatial extent of the operator. By the method of Lagrange multipliers, we can make $\delta C(h)/\delta h = 0$, where

$$C(h) = \int_{-W}^{0} [h^2(x) + \lambda_1 h(x) + \lambda_2 h'^2(x) + \lambda_3 h''^2(x)]dx.$$

That leads to the Euler-Lagrange equation

$$2h(x) + \lambda_1 - 2\lambda_2 h''(x) + 2\lambda_3 h^{(4)}(x) = 0$$

The solution to this differential equation gives the optimal one-dimensional operator

$$h(x) = e^{-\alpha x}(a_1 sin\ \omega x + \alpha_2 cos\ \omega x) + e^{\alpha x}(\alpha_3 sin\ \omega x + \alpha_4 cos\ \omega x) - \frac{\lambda_1}{2}$$

with conditions on α and $\omega : \alpha^2 - \omega^2 = \lambda_1/\lambda_2$ and $4a^2\omega^2 = (4\lambda_3 - \lambda_2^2)/4\lambda_3^2$.

Closed expressions can be computed for $\alpha_1, \alpha_2, \alpha_3$, and α_4 as functions of α, ω, c_3, and λ_1, resulting in $h(x)$ parameterized in terms of α, ω, c_3, and λ_1. The problem of finding the optimal operator has been reduced from an optimization problem in an infinite-dimensional space (the space of admissible functions h) to a nonlinear optimization problem with variables α, ω, c_3, and $\lambda 1$. These values are obtained using constrained numerical optimization methods. The optimal operator *h(x)* computed in this manner resembles the first derivative of a Gaussian $h(x) = -(x/\sigma^2)e^{(-x^2/2\sigma^2)}$ Fig. 11(c) shows the result of using the Canny edge detector.

Another popular edge detector that we will not discuss in detail is the Heitger detector [35]. It uses oriented energy maps, yielding good continuity of features near junctions and precise estimation of gradient orientation.

The complexity of the discrete two dimensional convolution central to the edge detection techniques described so far can be reduced from quadratic time, to $\theta(nlogn)$ by using a fast Fourier transform [36]. Lately however, another type of filters that can be computed in constant time has gained popularity for real-time applications. Such filters come from the Haar wavelets, which are a set of basis functions that encode differences in intensities between adjacent image regions. The more simple Haar filters are shown in Fig. 12, and could in a sense be seen as extended Prewitt differential operators. These filters are commonly used to represent objects via the Haar wavelet decomposition. Based on the fact that the response to image convolution with Haar filters vaguely approximates their first order Gaussian derivatives counterpart, many object recognition systems have made the swap from the former to the latter, with the great benefit of computational cost reduction [37, 38]. By using Haar filters one can compute feature or interest point detection in real time, as well as to compute local orientation values. In [39] for example, they are used in pedestrian and face detections tasks. Their fast computation is achieved using an *integral image* [40].

An integral image is a representation of the image that allows a fast computation of features because it does not work directly with the original image intensities but over an incrementally built image that adds feature values along rows and columns. Once computed this image representation, any one of the Haar features can be computed in constant time independently of its location and scale.

In its most simple form, the value of the integral image *ii* at coordinates *u,v* contains the sum of pixels values above and to the left of *u,v*, inclusive. Then, it is possible to compute for example, the sum of intensity values in a rectangular region simply by adding and subtracting the cumulative intensities at its four corners in the integral image. Furthermore, the integral image can be computed iteratively using the previous pixels values,

$$ii(u, v) = i(u, v) + ii(u - 1, v) + ii(u, v - 1) - ii(u - 1, v - 1)$$

Most edge detection operators produce edges that are not connected. Before any higher-level scene interpretation modules can be brought to bear on an image, it is often necessary to repair broken edges if such breakages were caused by noise and other artifacts. Edge repair can sometimes be carried out by expanding and shrinking the detected edges in such a way that any connection made during expansion is not lost during the shrinking operation. In a binary image, two pixels are connected if there is a path of neighboring pixels linking them [41, 42]. They are 4-connected if the path can be followed by traversing along the u and v directions only. An 8-connected path is obtained by traversing along the u and v directions as well as in diagonal directions. All pixels connected to a given pixel p in a set S of 1's form a connected component of S. If S has only one component then S is simply connected, otherwise it is multiply connected. The border *S'* of a set S is made up of those pixels of S for which at least one neighbor is in its complement $\overline{\mathrm{s}}$. The i-th iteration of an expansion is given by $S^{(i)} = S'^{(i-1)} \cup \overline{\mathrm{S}}'^{(i-1)}$, and the i-th shrinking iteration is given by $S^{(i)} = S^{(i-1)} \sim S'^{(i-1)} = \overline{\overline{\mathrm{S}}}^{(i-1)} \cup S'^{(i-1)} = S^{(i-1)}\overline{\mathrm{S}}'^{(i-1)}$.

Other operations that may be performed on binary images include border following, thinning, and labeling [42]. Border following can be implemented using crack or chain coding schemes. This is, following the border of the elements in S using 4-connectivity or 8-connectivity. Thinning is similar to shrinking with the exception that the endpoints of the elements in S should not be deleted from the image. In Fig. 1(c) for example, the edges detected using Sobel operators are thinned to form one-pixel wide edges. Labeling consists on assigning an entry in a database to every separately connected component of S.

A powerful and frequently used approach for grouping together the edge elements that form straight lines in an image is based on the concept of Hough transformation [43] that, in its more common implementation, maps a straight line in an image into a single point in (d, θ) space, d and θ being the two invariant parameters in the polar coordinate representation of a line. The (d, θ) space is also known as the Hough space. A generalization of this approach can also be used for grouping together the detected fragments

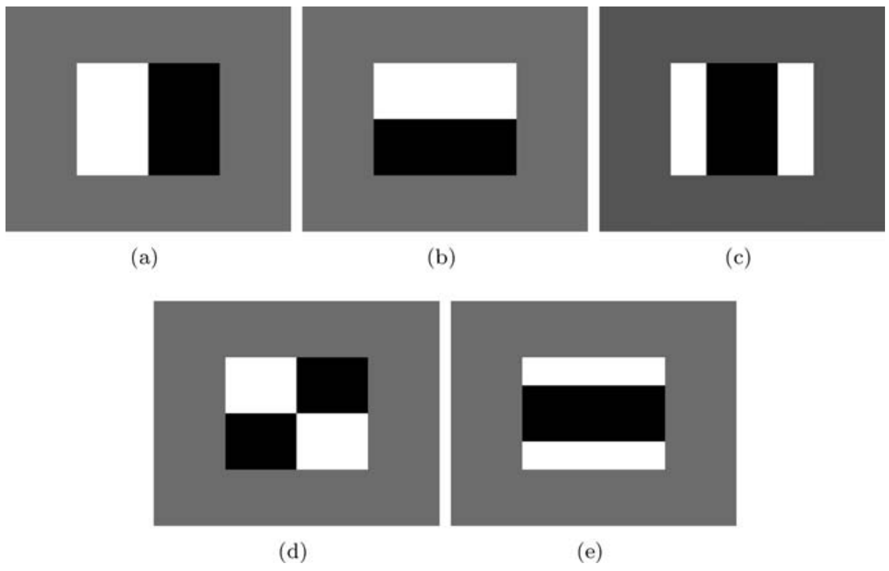


Figure 12. Haar filters

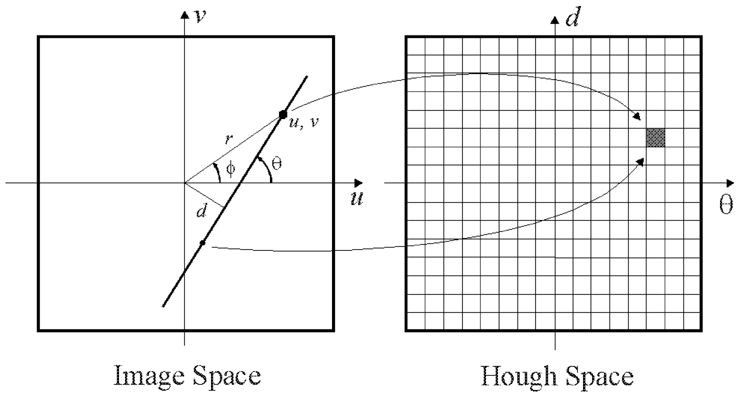


Figure 13. The Hough transform maps straight lines in the image space into single points in the (d, θ) space. It can be used to group together unconnected straight line segments produced by an edge operator.

of smooth curves [44]. For Hough transform based extraction of straight lines, the distance of each edge pixel from the origin is computed by using $r = \sqrt{u^2 + v^2}$, and the orientation by $\phi = \tan^{-1}(v/u)$. See Fig. 13. The edge orientation angle θ is obtained from the output of a Sobel or LoG operator, and the perpendicular distance from the image origin to the edge with point (u,v) and orientation θ is $d = r\,sin\,(\theta - \phi)$. Once d and θ are computed, the corresponding cell in the Hough space is incremented. After processing the entire image, the lines corresponding to the cells with the highest number of hits are redrawn on top of the original image. Then, a raster scan is performed on the entire image to find the points near this line. This idea can be extended to extract curves from the output of an edge detector. Instead of using the parametric equation of a line $d = r\,sin\,(\theta - \phi)$, the generalized parametric equation for the desired curve must be used to define the Hough space, i.e., for circles $(u - u_0)^2 + (v - v_0)^2 = c^2$ defines a 3-D voting array with perpendicular directions u_0, v_0, and c. If the output of the edge detector is not a binary image, the update values for the cells on the Hough space may be weighted with the intensity of the pixel being analyzed.

Another approach to boundary localization includes the use of active contours, namely *snakes* [45]. The classical snakes approach is based on deforming an initial contour curve toward the boundary of the object to be detected. The deformation is obtained by minimizing an energy function designed such that a local minimum is obtained at the boundary of the object. This energy function usually involves two terms, one controlling the smoothness and continuity of the contour curve and the other attracting it to the object boundary. The idea of active contours can also be extended to 3-D object recognition by using 3-D deformable surfaces [46]. In this case, instead of tracking the boundary of an object in a 2-D image, the surface representation of the object is computed using 3-D information, such as that obtained from a structured-light sensor.

The idea behind edge detection, or any other low-level process, is to prepare the image so that specific image components can be clustered. The clustering of image components into higher level organizations such as contours, each from a single object, is known as grouping or perceptual organization [47, 48]. A grouping process can improve the search for an object in a recognition system by collecting together features that are more likely to come from the object rather than from a random collection of features. Most model-based recognition systems exploit such simple grouping techniques.

Interest Points

Another possibility is to represent objects with *interest points* rather than edges, with the advantage that occlusions and cluttered scenes can be dealt with. Moreover, if such interest points are affine invariant, then disparate views of the same object can be handled more easily. Interest points are usually located at distinctive locations in images where image intensities vary two-dimensionally, such as at blobs, T-junctions, Y-junctions, and corners in general. The most popular of these is the Harris corner detector [49], an improvement over the Moravec corner detector [50], that analyses the eigenvalues of the local image autocorrelation matrix

$$M = [\begin{matrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{matrix}]$$

where I_u and I_v are the partial derivatives of I computed by convolution of the image with the Gaussian derivatives $\partial G/\partial u$ and $\partial G/\partial v$, respectively, and centered at pixel coordinates u, v. If the two eigenvalues of M are large an interest point is detected, and with only one large eigenvalue an edge is detected. There is no need however to explicitly compute such eigenvalues. Instead, the value of the Harris corner detector at (u,v) is given by

$$I_{\text{Harris}}(u, v) = \det M - k \text{trace}^2 M$$

The parameter k is adjusted empirically, frequently in the range 0.04–0.15.

Another interest point detector, the Beaudet detector [51] is computed from the determinant of the Hessian matrix

$$H = [\begin{matrix} I_{uu} & I_{uv} \\ I_{uv} & I_{vv} \end{matrix}]$$
$$I_{\text{Beaudet}} = det$$

and, as previously discussed for edges, it is possible to use Haar filter-based approximations of the second order partial Gaussian derivatives [37] in order to reduce its computational cost. The result is the Speed Up Robust Feature (SURF) detector [38].

Another well known corner detector is given at the maxima of the same LoG filter discussed for edges in the previous section. Interestingly enough, the LoG is equivalent to the trace of the Hessian

$$I_{\text{LoG}}(u, v) = \text{trace}\, H$$

Fig. 14 shows the response of the Harris, Beaudet, and LoG cornerness measures when applied to the object image from Fig. 10(a).

Unfortunately, the tracking from one frame to the next of the geometrical features that respond to these operators might still be hard to attain. Affine deformations caused by the change in viewpoint, or by the variation of the reflectance conditions contribute to such difficulty. With that in mind, Shi and Tomasi formulated an image feature selection algorithm optimal by construction from the equations of affine motion [52].

Starting from the assumption that a feature in an object will have similar intensity values on two consecutive images I and J, the affine motion $(\mathbf{D}, \mathbf{d})$ of a window of pixels around such feature $\mathbf{m} = (u, v)^\top$ from the first image to the second can be represented with

$$I(\mathbf{Dm} + \mathbf{d}) \approx j(\mathbf{m})$$

and with the simplification that for small displacements the affine transformation can be modeled by the identity matrix, a Taylor series approximation of the image intensity change is given by the expression $I(\mathbf{m}) + \nabla^\top I(\mathbf{m})\mathbf{d}$. We can then formulate a measure of dissimilarity for a tracked feature in the two image frames simply as the sum of squared differences

$$\varepsilon = (I(\mathbf{m}) + \nabla^\top I(\mathbf{m})\mathrm{d} - J(\mathbf{m}))^2$$

Differentiating with respect to the displacement $\mathbf{d}$, and setting the result equal to zero yields the system

$$(I(\mathbf{m}) - J(\mathbf{m}))\nabla I(\mathbf{m}) = [\begin{matrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{matrix}](\mathbf{m})\mathbf{d}$$

indicating that a feature centered at m can be tracked reliably when the above system is well conditioned. We end up choosing as features the points in the image for which the square of the gradient

$$[\begin{matrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{matrix}]$$

has both eigenvalues larger than a given threshold. The chosen points will be located near corners, in highly textured regions, or in any other pattern that can be tracked reliably.

As such, the feature detectors just described, from the Harris detector to the Shi detector, are not scale invariant. To accommodate for changes in scale, interest points must be localized both in spatial image coordinates (u,v), as well as in scale σ, and the *characteristic* or relevant scale for each interest point must be taken into account when building a multi-scale object descriptor.

To search an image for the most relevant features along different scales, a 3-dimensional image must first be constructed by iteratively convolving the original image with Gaussian filters of increasing scale. Take for example the extrema of the LoG [53]. In that case, the LoG must be first scale-normalized so that the response of the filter at different scales can be compared. The scale-normalized LoG filter is given by

$$\Delta G = \sigma^2(\frac{\partial^2 G}{\partial u^2} + \frac{\partial^2 G}{\partial v^2})$$

As with the LoG, the determinant of the Hessian and the squared of the gradient interest point detectors can also be scale-normalized [53]. Another operator that takes into account variations of scale is the Difference of Gaussians (DoG) operator [54]. The DoG operator is an approximation to the LoG filter, with the advantage of reducing the computational cost. It is computed by the weighted difference of two consecutive smoothed images.

$$DoG(u, v, \sigma) = (G(u, v, k\sigma) - G(u, v, \sigma)) * I(u, v)$$

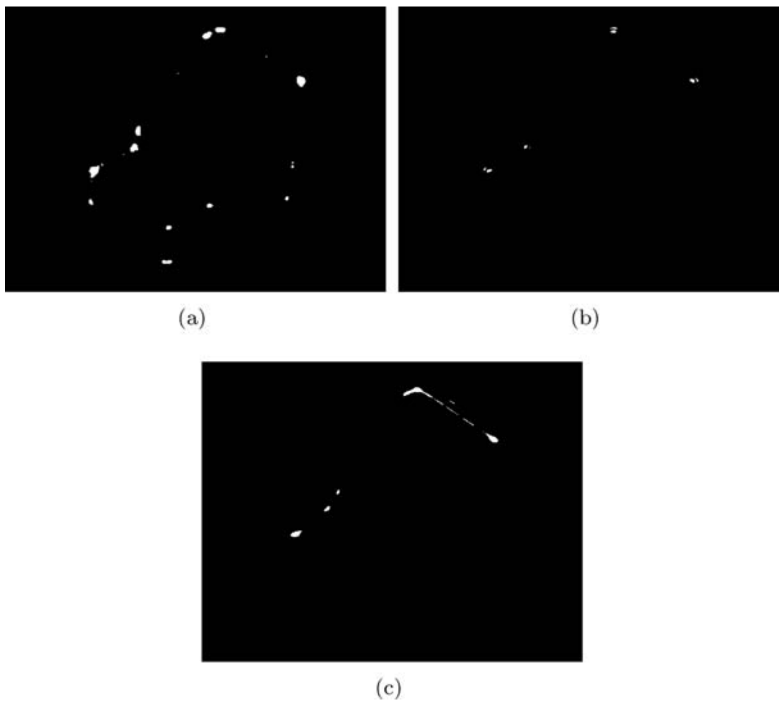


Figure 14. Interest point detectors: (a) Harris, (b) Beaudet, and (c) LoG maxima

In a comparison of the aforementioned scale-normalized filters [55] the LoG outperformed the other operators in terms of amount of interest points detected and correct scale correspondence, with the advantage that it guarantees a single maximum in the scale trace. The DoG filter achieved similar results as the LoG given that it is an approximation of the former, but at a reduced computational cost [56].

The image of an object observed from different viewpoints is said to suffer a perspective transformation. Perspective transformations are difficult to deal with due to their intrinsic nonlinearities, but can be approximated as affine transformations when the camera motion is small, or if the object is located sufficiently away from the camera; and can even be exactly computed for the case of planar objects. In this sense, some researchers have developed lately interest point detectors invariant to affine transformations [57–60].

The affine invariant interest point detector reported in [57] builds up from the Harris detector and iterates over scale, position, and shape on the neighborhood of a point until it converges to an affine invariant interest point. In their approach, a second order moment matrix (image autocorrelation) is used to normalize the image point neighborhood.

Once a set of interest points have been localized both in spatial image coordinates (u,v) and scale (σ), a descriptor must be computed for each of them from their neighborhood appearance. There exist a mirage of descriptors in the literature for object recognition; and vary from moments invariants [61], to differential descriptors such as the steerable filters [32], differential invariants [62], and even distribution descriptors such as the popular SIFT [56]. This last one has demonstrated to outperform other descriptors in terms of correct detection rate over sets of images under widely varying viewing conditions [63].

The SIFT descriptor uses 3D local histograms made up of location and gradient orientations. For each point neighborhood the gradient image is sampled over a 4×4 grid of locations, and its discrete gradient orientation at 8 different orientations is computed. The resulting descriptor is of dimension 128. In order to deal with illumination changes, the description vector is normalized with and Euclidean norm. The SIFT uses the DoG to compute the appropriate scale for each interest point. Fig. 15 shows the type of features that can be extracted when this interest feature operator is used.

Low Level Processing for Structured Light Projection Systems

The 3-D point coordinates obtained from a structured-light system are generally stored in a matrix whose columns correspond to the light stripes used to illuminate the scene and whose rows correspond to horizontal scan lines of the camera used to capture stripe images. For the light stripe indexed i and the camera scan line indexed j, one ends up with three numbers, $x_{i,j}$, $y_{i,j}$, and $z_{i,j}$, that represent the world coordinates of the illuminated point and, if desired, three additional numbers, $R_{i,j}$, $G_{i,j}$, $B_{i,j}$, that represent the RGB color coordinates of the white light reflected by the object point. One is not limited, of course, to using the RGB color space, since it is trivial to transform the color coordinates into any other desired representation. Computed for all i and j, the numbers $x_{i,j}$, $y_{i,j}$ and $z_{i,j}$ constitute a range map of the scene. An example of a range map obtained using structured-light imaging is shown in Fig. 5(b). In what

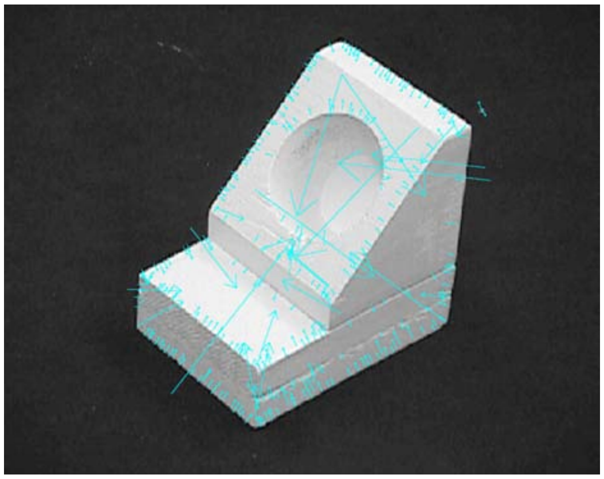


Figure 15. SIFT features

follows, we will use the vector $\mathbf{p}_{i,j}$ to denote

$$\mathbf{p}_{i,\,j} = [x_{i,\,j}, y_{i,\,j}, z_{i,\,j}]^\top$$

After a range map is recorded, the next step is the extraction of analytically continuous surfaces from the scene. In other words, we want to be able to group together object points into planar, cylindrical, conical, and other surfaces that can be described by relatively simple analytical forms. A necessary first step to such grouping is the computation of the local surface normals from a range map. Theoretically at least, the local surface normal at a point $\mathbf{p}_{i,j}$ in a range map can be computed from

$$\hat{\mathbf{n}} = \frac{\frac{\partial \mathbf{p}}{\partial i} \times \frac{\partial \mathbf{p}}{\partial j}}{|\frac{\partial \mathbf{p}}{\partial i} \times \frac{\partial \mathbf{p}}{\partial j}|}$$

but unfortunately this approach does not work in practice because of the noise-enhancing properties of the derivatives. What works very effectively is an approach that is based on assuming that an object surface is locally planar in the vicinity of each measurement. This local surface can be given the following algebraic description:

$$\mathbf{p}_{i,\,j} \cdot \hat{\mathbf{n}} = d$$

at point (i,j) in the range map. Consider now a small squared window $W_{i,j}$, usually 5×5 or 7×7, around a point (i,j). The error between a fitted planar patch and the measured range map values within this window is given by

$$\varepsilon = k, l\varepsilon W_{i,\,j} \sum (\mathbf{p}_{k,l}^\top \hat{\mathbf{n}} - d)^2$$

This error can be re expressed in the following form

$$\varepsilon = \hat{\mathbf{n}}^\top Q \underbrace{(}\ k, l\varepsilon W_{i,\,j} \sum \mathbf{p}k, l\mathbf{p}_{k,l}^\top)\hat{\mathbf{n}} - 2d\mathrm{q}^\top \underbrace{(}\ k, l\varepsilon W_{i,\,j} \sum \mathbf{p}_{k,l}^\top)\hat{\mathbf{n}} + N^2 d^2$$

We evidently must choose the value for the normal that minimizes the error. This optimum value for $\hat{\mathbf{n}}$ is computed by setting equal to zero the partial derivatives of the following Lagrangian

$$l(\hat{\mathbf{n}}, d, \lambda) = \varepsilon + \lambda(1 - \hat{\mathbf{n}}^\top \hat{\mathbf{n}})$$

We get

$$\frac{\partial l}{\partial \hat{\mathbf{n}}} = 2\mathbf{Q}\hat{\mathbf{n}} - 2d\mathbf{q} - 2\lambda\hat{\mathbf{n}} = 0$$

$$\frac{\partial l}{\partial d} = -2\mathrm{q}^\top \hat{\mathbf{n}} - 2N^2 d = 0$$

$$\frac{\partial l}{\partial \lambda} = 1 - \hat{\mathbf{n}}^\top \hat{\mathbf{n}} = 0$$

Substituting the second of these equations in the first, we end up with the following

$$\mathbf{Q}\hat{\mathbf{n}} - \frac{\mathbf{q}\mathbf{q}^\top \hat{\mathbf{n}}}{N^2} - \lambda\hat{\mathbf{n}} = 0$$

or, equivalently,

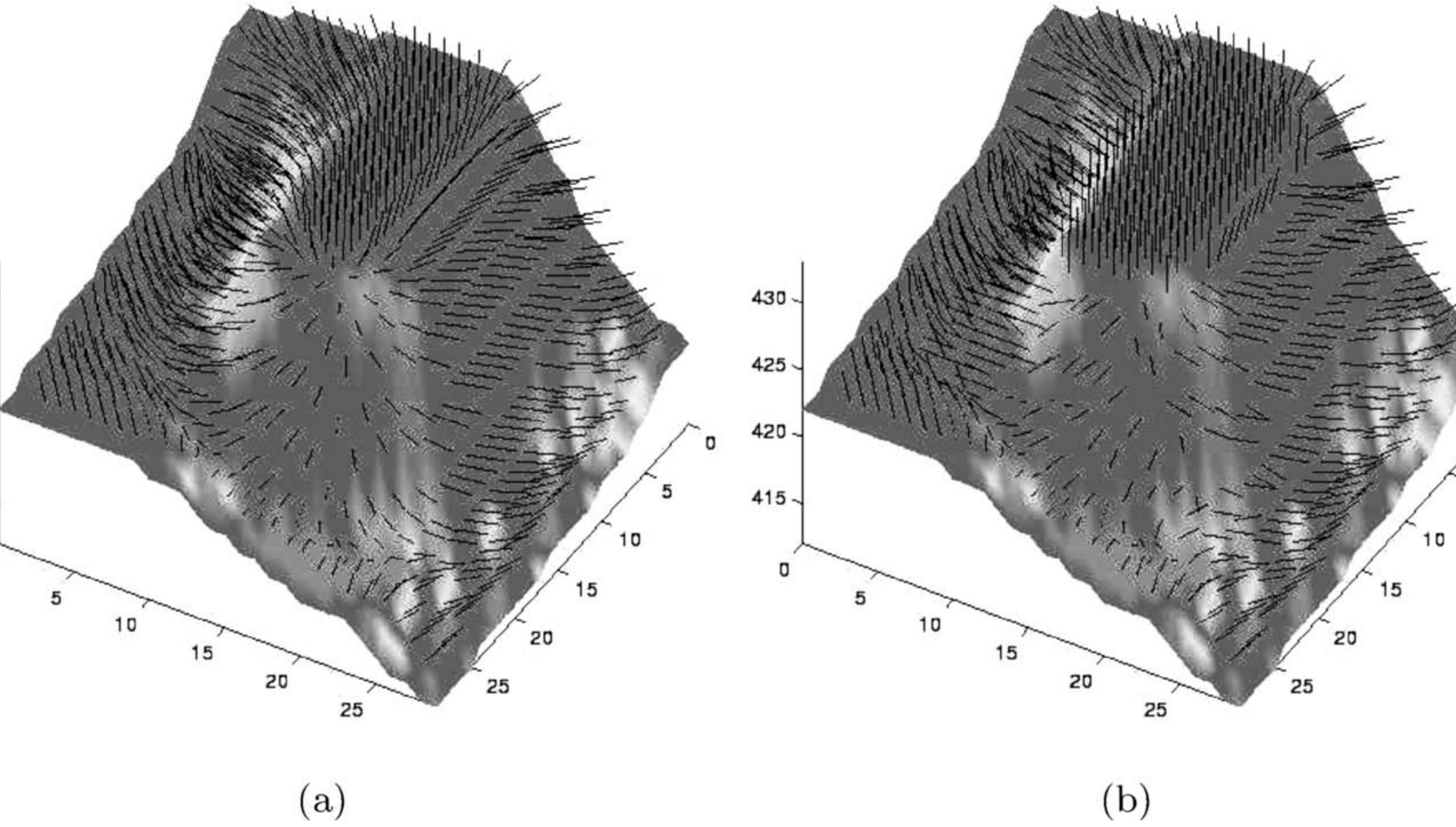
$$\mathbf{R}\hat{\mathbf{n}} = \lambda\hat{\mathbf{n}}$$

where $\mathbf{R}$ is given by

$$\mathbf{R} = \mathbf{Q} - \frac{\mathbf{q}\mathbf{q}^\top}{N^2}$$

The solution to $\mathbf{R}\hat{\mathbf{n}} = \lambda\hat{\mathbf{n}}$ is obviously an eigenvector of the 3 × 3 matrix $\mathbf{R}$, and we choose that solution which corresponds to the smallest eigenvalue, for the simple reason that it can be shown trivially by substitution that the error ε becomes equal to the eigenvalue when we use the corresponding eigenvector of $\mathbf{R}$ for the surface normal. Shown in Fig. 16(a) is the needle diagram of a range map showing the surface normals computed for an object. The orientation of each needle is a projection of the local surface normal on a display plane.

Used without any further modification, the above approach will still fail to give acceptable results if the window $W_{i,j}$ straddles the boundary between two smooth surfaces or includes a jump edge. Such distortions can be virtually eliminated by adaptive placement of the windows in the vicinity of edges. For every point $\mathbf{p}_{i,j}$ the window $W_{i,j}$, is composed of the neighboring points $\mathbf{p}_{k,l}$, with $i - N/2 \le k \le i - N/2$, and $j - N/2 \le l \le j - N/2$. As mentioned earlier, at each of these points we should have a



normal $\hat{\mathbf{n}}_{k,l}$, and a fitting error $\varepsilon_{k,l}$. The idea behind adaptive windowing is a reassignment of the computed normals to each point $\mathbf{p}_{i,j}$, the reassigned normal being chosen from among the neighboring normals on the basis of the smallest product $w_{i,j,k,l}\varepsilon_{k,l}$. The weight $w_{i,j,k,l}$ can be selected as the inverse of the city block distance between the points $\mathbf{p}_{i,j}$ and $\mathbf{p}_{k,l}$.

$$w_{i,j,k,l} = \frac{1}{c + |i-k| + |j-l|}$$

The constant c is chosen such that the distance weighting will be the dominant factor in $w_{i,j,k,l}\varepsilon_{k,l}$. Fig. 16(b) shows the needle diagram of the same range map with adaptive placement of the $W_{i,j}$ windows.

After the local surface normals are computed in the manner presented above, one must segment the range map in such a way that each segment represents a smooth surface. Such surfaces are bounded by crease edges where surface normal discontinuities occur, or by jump edges where neighboring points in the range image correspond to distant points in the scene. Smooth surface segments are recursively generated by starting at any point in a range map and growing outwards while meeting the following two criteria for the neighboring points $\mathbf{p}_{i,j}$ and $\mathbf{p}_{k,l}$.

$$\|\mathbf{p}_{i,j} - \mathbf{p}_{k,l}\| > \text{jump edge threshold}$$

$$\frac{\cos^{-1}(\hat{\mathbf{n}}_{i,j}^{\top}\hat{\mathbf{n}}_{k,l})}{\|\mathbf{p}_{i,j} - \mathbf{p}_{k,l}\|} > \text{curvature threshold}$$

The two thresholds are determined empirically for a given class of objects.

The next step in low-level processing consists in classifying each smooth segment of a range map on the basis of its analytic properties. For most industrial objects, this classification is into planar, conical, or cylindrical; a category called "other" can also be included if desired. This classification is easily done by computing the extended Gaussian image (EGI) of a surface [64]. The EGI of an object surface is obtained by mapping the surface normal at every point onto a sphere of unit radius on the basis of identity of surface normals. In other words, a point $\mathbf{p}_{i,j}$ is mapped to that point of the unit sphere where the outward normal is the same as the one computed at $\mathbf{p}_{i,j}$. The unit sphere is also known as the Gaussian sphere. As shown in Fig. 17, the EGI image of a planar surface is a small patch whose orientation on the Gaussian sphere corresponds to the normal to the plane. For a conical surface, the EGI is a minor circle with its axis parallel to the axis of the conical surface; and for a cylindrical surface, the EGI is a great circle whose axis is again parallel to the axis of the cylinder. The distance from the center of the sphere to the plane containing the circle in each case is $d = \sin\theta$, whereas the radius of the circle is $r = \cos\theta$, θ being the cone angle. Therefore, in order to declare a surface type as planar, conical, or cylindrical, a plane must be fitted to the EGI points. The equation for the best-fitting plane is $\hat{\mathbf{n}}^\top \hat{\mathbf{a}} = d$. This problem is identical to that of fitting a planar patch to the neighboring points on a range image point, and reduces to computing the eigenvector corresponding to the smallest eigenvalue of the matrix

$$\mathbf{R} = \sum_{i=1}^{N} \hat{\mathbf{n}}_i \hat{\mathbf{n}}_i^\top - \frac{\sum_{i=1}^{N} \hat{\mathbf{n}}_i \sum_{i=1}^{N} \hat{\mathbf{n}}_i^\top}{N^2}$$

in the equation $\mathbf{R}\hat{\mathbf{a}} = \lambda\hat{\mathbf{a}}$, where N is the number of points on the segmented surface, $\hat{\mathbf{n}}_i$ are the previously computed normals at each point, and the computed eigenvector $\hat{\mathbf{a}}$ is the axis orientation of the detected surface. The distance $d = \hat{\mathbf{a}}^\top \sum_{i=1}^{N} \hat{\mathbf{n}}_i / N$ is used to characterize the surface type. For a plane $d \approx 1$, for a cone $0 < d < 1$, and for a cylinder $d \approx 0$. Fig. 5(c) shows a segmented range map. In this example, segments 10, 11, 26, 27, and 43 were characterized as conical surfaces, whereas the rest of the segments detected were classified as planar surfaces.

OBJECT REPRESENTATION

Object Representation for Appearance-Based Recognition

The data structures used to represent object models and the data acquired from an image or a range sensor depend on the method used for recognition. In appearance-based recognition, an instance of an object and its pose is computed without first determining the correspondence between individual features of the model and the data [27,65–67]. A vector of global parameters is computed from the sensory data, and it is compared with similar vectors previously obtained during a training session, looking for the best-matching model. If the primary goal is object identification, the vectors computed during the training session correspond to different objects. On the other hand, if the main concern is object pose computation, then each vector computed during the training session corresponds to different views of the same object, provided the object has already been identified.

There exist several ways to construct these global parameter vectors. They can be based on simple geometric attributes such as area, perimeter, elongation, or moments of inertia, or on spatial frequency descriptions such as the discrete cosine transform, Fourier descriptors,wavelets, or eigenimages. When object identification is of primary concern, the attributes selected must be invariant to changes in the object's pose. When the task requires object pose computation, the parameters used should diverge for different views of the same object.

The selection of image attributes for image parametrization in object recognition is also termed parametric appearance matching [65]. For 3-D object recognition, the appearance of an object depends on its shape, reflectance properties, pose in the scene, and illumination conditions. When the illumination conditions are the same for different scenes, the shape and reflectance for a rigid object can be considered as intrinsic properties. An appearance-based object recognition system must learn the objects for identification. To learn an object, the system is presented with multiple views of the same object at different orientations. The result is a large set of images for the same object with high correlation among them. To ease the search for the corresponding object class for a given image, the large set of training images is usually compressed into a low-dimensional representation of object appearance.

One method for image compression, known as *principal components analysis*, is based on the Karhunen-Loéve transform. In this method, all images are projected to an orthogonal space, and then they are reconstructed by using only their principal components. Consider every image to be a random vector $\mathbf{x}$ with dimensionality $N = uv$, where u and v are the image width and height in pixels respectively. All the images for the same object are expected to be highly correlated, and to lie in a cluster in this N-dimensional space. In order to reduce the dimensionality of the space all the training images are projected onto a smaller space minimizing the mean squared error between the images and their projections. The center of the cluster of n images for the same object with varying pose is the point

$$\hat{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$$

and the unbiased sample covariance matrix is given by

$$\Sigma = \frac{1}{n-1} i = 1 \sum (\mathbf{x}_i - \hat{\mathbf{x}})(\mathbf{x}_i - \hat{\mathbf{x}})^\top$$

The projection of each image onto a space of dimensionality $M < N$ can be computed by

$$\mathbf{y}_i = [\begin{matrix} \phi_1^\top \\ \phi_2^\top \\ \vdots \\ \phi_M^\top \end{matrix}](\mathbf{x}_i - \hat{\mathbf{x}})$$

where the vectors $\phi_1, \phi_2, \ldots, \phi_M$ from an M-dimensional basis for the new space with origin at $\hat{\mathbf{x}}$. The basis vectors ϕ_i are orthonormal. This is, they are linearly independent, have unit length, and completely span IR^M. The optimum choice for the ϕ_i are those that satisfy $\Sigma\phi_i = \lambda_i\phi_i$, that is, the eigenvectors that correspond to the M largest eigenvalues of Σ. Fig. 18 shows a 2-D example of the Karhunen-Loéve transform. The eigenvector ϕ_1 is the principal component of the data set, and the projections $\mathbf{y}_i$ on ϕ_1 minimize the error between the data points $\mathbf{x}_\mathrm{i}$ and their projections.

Though a large number of eigenvectors may be required for an accurate reconstruction of an image, a few are generally sufficient to capture the significant appearance characteristics of an object. The space spanned by $\phi_1, \phi_2, \ldots, \phi_M$ is also commonly referred as the eigenspace. If two images are projected into the eigenspace, the distance between the corresponding points in the eigenspace is a good measure of the correlation between the images.

When the goal is to discern among different objects, images of all objects are used together to construct the eigenspace during the training phase. Several images of each object with varying pose and illumination conditions are projected to the eigenspace to obtain a set of points. The set of points for each object is expected to be clustered together representing that object class. Then, an image of the

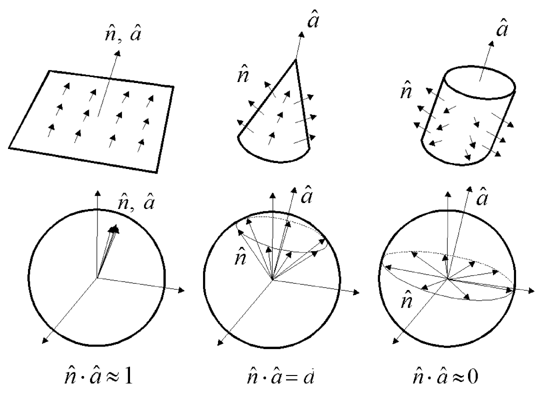

$\hat{n}, \hat{a}$
$\hat{a}$
$\hat{n}$
$\hat{n}\cdot\hat{a}\approx 1$
$\hat{n}\cdot\hat{a}= a$
$\hat{n}\cdot\hat{a}\approx 0$

Figure 17. The extended Gaussian image is used to identify the shape of a segment extracted from a range map. The orientations of the normals at different points in a segment obey different analytic properties for different surface types.

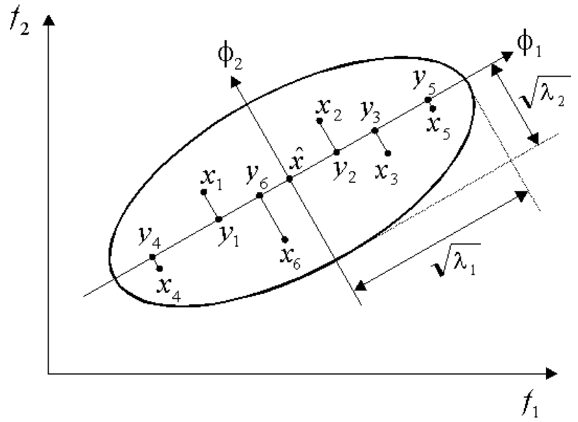


Figure 18. A 2-D example of the Karhunen-Loéve transform. The projections y_i into the reduced space spanned by ϕ_1 minimize the error between the data points, $\mathbf{x}_i$ and their projections.

object to be recognized is also projected to the eigenspace, and is classified as belonging to the closest cluster class it maps to. To estimate the orientation of an object once it has been identified, the same scene image is mapped to an eigenspace made of only the training samples for that object. A manifold is constructed by interpolating these training points using a curve that is parametrized by pose, scale, or illumination. The closest point in the manifold obtained provides an estimate of the pose and illumination conditions of the object [65].

In a noncorrespondence based recognition system, object representation is appearance-based. That is, instead of identifying local object features in the sensed data, global parameters are computed from the whole image. For this reason, most appearance-based recognition systems developed to date require that the variations in scene illumination be small and that the objects not be occluded. Although the nonocclusion and illumination constraints can be met for a wide range of vision applications, the more general case is still a difficult problem. An example application for the recognition of faces where occlusions are accounted for with the use of hidden Markov models is presented in Ref. [68]. The advantage of the appearance-based method is that it is not necessary to define a representation or a model for a particular class of objects, since the class is implicitly defined by the selection of the training set. On the other hand, model-based recognition systems can be designed to deal with situations where cluttered scenes and changes in illumination are present. The latest approaches to the solution of the object recognition problem consider the integration of both model-based methods and appearance-based analysis.

Object Representation for Model-Based Recognition

Three central issues arise when trying to achieve object recognition using a model-based approach: (i) the features used to describe an object should be such that they can be extracted from an image; (ii) it should be possible to aggregate the features into object models appropriate for recognizing all objects in a given class; and (iii) the correspondence or matching between image features and model features should permit recognition of objects in a complex scene [2]. Consider for example the case of 3-D object recognition from 3-D data when objects are in bins of the sort shown in Fig. 5. Each of these objects can be given a geometrical representation whose fundamental constituents are surfaces, edges, and vertices. Such a representation can drive a model-based computer vision system, because, as described earlier, it is possible to extract such geometric features from range maps. Of course, the geometrical features themselves need a representation, the common one being via what are known as attribute-value pairs.

Consider for example the object shown in Fig. 19(a) whose wire-frame representation is shown in Fig. 19(b, c). The surface F of this object can be represented by the following set of attribute-value pairs:

Shape: planar
Area: 4516.1 mm^2
Color: white
Normal axis orientation: (0,0,1)
Adjacent surfaces: $\{E, G, J, K\}$
Angles with adjacent surfaces: $\{-90^\circ, 90^\circ, 90^\circ, 90^\circ\}$

Similarly, the edge l feature in Fig. 19(b) can be represented by

Shape: line
Length: (l) 88.9 mm
Type: convex
Delimiting vertices: {3,10}
Delimiting surfaces: $\{B, E\}$
Orientation: (0.643, −0.766,0)

and the vertex feature 10 by

Location: (165.1 mm, 101.6 mm, 57.15 mm)
Adjacent vertices: {3,9,11}
Outgoing edges $\{l, r, s\}$
Surrounding surfaces: $\{B, E, K\}$

In general, for geometry-based model descriptions, a feature can be any simple geometric entity such as a vertex, an edge, or a surface. In these model representations, surfaces whose algebraic descriptions are up to the second order are easy to represent. For example, in Fig. 19(b), the surface C is a truncated conical surface that would be represented by the following data structure:

Shape: *conical*
Area: 6964.0 mm^2
Color: white
Normal axis orientation: (−0.494,−0.588,0.640)
Top radius: 19.05 mm
Base radius: 31.75 mm
Height: 50.8 mm
Adjacent surfaces: $\{B, D\}$

Once we have settled on what features to use to describe an object, we need to address the second issue raised at the beginning of this section, viz., how to aggregate the features into object models. We evidently need to embed the features in some structure that will also capture the spatial relations between the features. The most fundamental data structure that computer science makes available to us for representing relational information is an attribute graph. The nodes of such a graph can represent each of the features we have discussed, and the edges represent adjacency or any other relationship between the features.

For example, Fig. 20 shows a simple attribute graph for the three-dimensional object from Fig. 19. In this object model representation the nodes in the graph correspond to the object surfaces, and the edges indicate the adjacency attribute. The number associated with an edge in the graph is related to the length of the edge connecting the two surfaces in the object model.

To construct an object model, we need to learn the attribute-value pairs for each feature in the object. In an industrial setting, these values would often be available in the CAD files coming from the design department; but if such information is not at one's disposal, a user-guided learning system that extracts these values from training samples can be used. In the MULTI-HASH system [13], for example, an interactive learning process is used to compute the attribute values from training samples. The user presents to the system each object that the system is expected to recognize in many different poses (this can be done easily by placing objects in a sandbox) and, with the help of a pointing device, establishes correspondences between the features on a model object and the features extracted from sensed data. In this manner, the system automatically determines what attribute values to use for describing the different features on a model object. An advantage of such learning systems is that they take into account the measurement noise that is always present in the data. The learning session used to construct a model base of the objects to be recognized usually takes place off line.

Measurement of Attribute Values from Scene Data

As was mentioned before, the first step in actual model-based object recognition is the segmentation of sensory data and then the extraction of attribute values for each of the segments. Subsequently, one can try to establish a correspondence between the segmented features thus obtained and the features in an object model.

Attributes that are measured using sensory data should, under ideal circumstances, be invariant to rotation, translation, scale, ambient illumination, background clutter, and so on. If a range sensor is used, the very nature of the data generated guarantees many of these invariances, provided a sufficient number of data points are collected for a given feature. With such a sensor, the area of a planar surface will be the same, as long as the surface is entirely visible and discounting the effect of highly oblique angles between the surface normal and the line of sight to the sensor. By the same token, the radius of curvature estimated for a cylindrical surface will be the same regardless of the viewpoint angle from the sensor to the surface. But, at the same time, one has to bear in mind that even with 3-D sensors, occlusion may cause large variations in attribute values as the viewpoint angle between the sensor and the surface is changed. In Ref. [13], the susceptibility of some attribute values to occlusion is taken care of in the design of matching criteria between model attribute values and scene attribute values. For example, for an attribute value such as the area of a surface, we can only demand that the area extracted from a surface in the scene be less than the area of the corresponding surface in the object model.

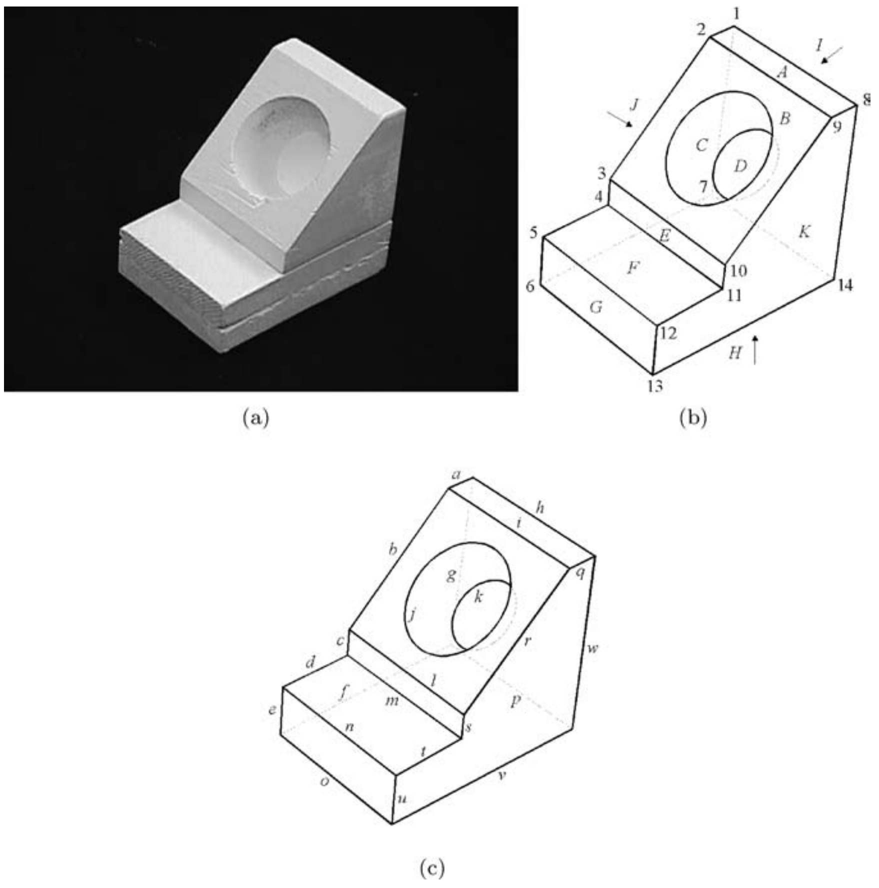


Figure 19. The geometrical representation of a typical 3-D object consists of features such as surfaces, edges and vertices, and for each feature of a set of attribute-value pairs.

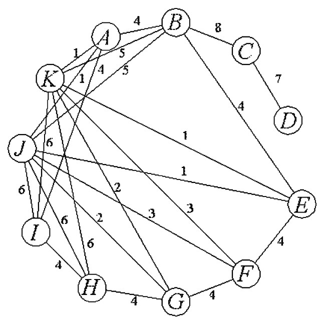


Figure 20. The nodes represent the object features, in this case the object surfaces; and the arcs indicate the adjacency between surfaces. The number on each arc is related in the following manner to the length of the physical edge joining the two surfaces in the object model: (1) 12.7 mm, (2) 38.1 mm, (3) 50.8 mm, (4) 88.9 mm, (5) 99.2 mm, (6) 114.3 mm, (7) 119.7 mm, and (8) 199.5 mm.

Lack of invariance in attribute values poses a bigger problem for recognizing 3-D objects in 2-D images. Basically all geometrical attributes, such as perimeter and area, vary with the scale, translation, rotation, and ambient illumination in 2-D images, not to mention that it is extremely difficult to carry out an automatic segmentation of the images to establish a relationship between the features in an object model and the segments extracted from the image.

These problems are fortunately not as daunting for recognizing 2-D planar objects in 2-D images. Some of the attributes that can be measured after segmentation in 2-D images include the perimeter and the moments of area of a segment. The perimeter of a segment can be computed by following the crack code or the chain code of the segment boundary representation, and the moments of area of a segment can be defined as summations over all the pixels in a segment along the u and v directions. The pq moment for a segment Ω in an image is given by

$$m_{pq} = u\varepsilon\Omega \sum v\varepsilon\Omega \sum u^{p} v^{q} I(u, v)$$

where $I(u, v)$ is the normalized gray-level intensity in the image and can be set to 1 for binary images. The zeroth moment m_{00} gives the area of a segment. The center of the segment can be computed by $[\overline{u}, \overline{v}]^\top = [m_{10}/m_{00}, m_{01}/m_{00}]^\top$. Moreover, the central moment given by

$$\mu_{pq} = u\varepsilon\Omega \sum v\varepsilon\Omega \sum (u - \overline{u})^p (v - \overline{v})^q I(u, v)$$

is invariant to translations, and $\eta_{pq} = \mu_{pq}/\mu_{00}^{\gamma}$, where $\gamma = (p + q)/2 + 1$, is scale invariant.

Other attributes that can be computed on features extracted from 2-D images are the segment bounding rectangle, the rectangularity F_R, the circularity F_C, and the radius R:

$$F_R = \frac{A_\Omega}{A_{\text{BR}}}$$

$$F_C = \frac{P_\Omega^2}{4\pi A_\Omega}$$

$$R = \frac{max u, v\varepsilon\Omega \sqrt{(u - \overline{u})^2 + (v - \overline{v})^2}}{min u, v\varepsilon\Omega \sqrt{(u - \overline{u})^2 + (v - \overline{v})^2}}$$

where A_Ω is the area of the segment, A_{BR} is the area of the bounding rectangle, and P_Ω the perimeter of the segment. An entry with the minimum and maximum u and v coordinates of the bounding rectangle can be included in the list of attributes for any given segment. The center of the bounding rectangle is a useful descriptor of segment position.

Inclusion relationships can also provide significant information for identifying a particular feature within an object model. When the features are planar segments in 2-D images, each segment descriptor will have a "parent" field that points to the surrounding segment. Additional links can be assigned for "child" and "sibling" segments. Ultimately, the background segment will be at the root node for all segments in an inclusion description tree. As long as the number of segments in a scene is not too large, these inclusion relationships can be obtained by searching through this tree. The number of "holes" present in a segment is termed the *genus*, and is equivalent to the number of "children" for a given segment in the inclusion description tree.

Automatic Learning

In many cases it is quite difficult, often impossible, to come up with a user-engineered set of relevant features to describe and object robustly. It is possible nonetheless, to train a learning system with multiple views of the object to be recognized to automatically choose the most distinctive features by itself. In such case, the problem is considered as that of finding a robust classifier from training instances of object and non-object classes. To this end, one can resort to conventional pattern recognition methodologies such as Support Vector Machines [69, 70] or Neural Networks [71]. One technique that has proved effective both in terms of rate of classification and computational cost, for difficult tasks such as face identification from images, is the use of weighted linear classifiers (boosting) [40,9].

The idea behind boosting is that the chaining of weak classifiers, each with guaranteed at least 50% classification success rate, can lead to a very strong classifier. In general a weak classifier can represent the presence in the object class of any given object feature such as an edge or even a homogeneous region. These features must be easy to compute, as they must be tested over all possible scales and locations, and over all input images, and its rate of classification succes be computed. Once training is completed, the algorithm evaluates the trained classifier over a sample image, usually at real-time. The *AdaBoost* algorithm [72] for example, builds a strong classifier H from the weighted linear combination of weak classifiers h

$$H = \Sigma \alpha_i h_i$$

The algorithm iterates, extracting on each round the weak classifier h_i which better separates the training samples with respect to the misclassification error. At each iteration, the algorithm re-weights more heavily those samples that have not been properly classified, with the hope that the next chosen classifier will be able to do so. The classification error ei is computed at each round as the sum of the weights for the misclassified samples, and the classification power α_i is assigned according to the error value over the training set

$$\alpha_i = \frac{1}{2} \ln(\frac{1 - e_i}{e_i})$$

A nice extension to the AdaBoost methodology for the detection of moving objects on temporal sequences is to embed not only spatial features, but temporal ones as well [73].

MODEL HYPOTHESIS GENERATION AND MODEL MATCHING

Appearance-Based Matching

As mentioned before, for appearance-based object recognition [66,67,27,65], the object models will correspond to topological structures in a multidimensional attribute space. Several training images of the same object with small viewpoint variations will usually map to different points in the attribute space, forming a manifold parametrized by pose. Different objects will correspond to different manifolds. The image obtained from the scene to be analyzed must be projected into the attribute space. The closer the projection of this image is to any of the manifolds, the greater the probability that the scene object is the model object corresponding to the manifold. Bear in mind from our discussion of appearance based object model representation that the attribute space used for model matching is obtained by reducing the dimensionality of the image space using one of many possible techniques, such as principal components analysis, discriminant analysis, or multidimensional scaling.

When estimating the distance between a scene data entry and the manifolds in the attribute space, the closest entry in the attribute space is called the *nearest neighbor*, and several measures can be used to compute the distance between nearest neighbors. The simplest of these distance measures is the Euclidean distance

$$d = \sqrt{(\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y})}$$

where $\mathbf{x}$ is the data point corresponding to the image of the unknown object as projected into the attribute space, and $\mathbf{y}$ is one of the training samples also projected into the attribute space. When the cluster of samples for the object class to be tested is assumed to have a normal distribution with covariance matrix Σ, a more appropriate measure of image similarity is the Mahalanobis distance

$$d = \sqrt{(\mathbf{x}-\mathbf{y})^{\top}\Sigma^{-1}(\mathbf{x}-\mathbf{y})}$$

The problem of model matching using appearance-based methods consists in finding the training image that minimizes the distance to the image containing the unknown object. If this distance is within a certain threshold, we can say that the unknown object belongs to the same class as the training data point. The advantage of using appearance-based methods over geometry-based ones is that it is not necessary to define a geometric representation for a particular class of objects, since the class is implicitly defined by the selection of the training samples. On the other hand, we may need a large number of training samples.

Model-Based Matching

In matching scene features to model features using model-based techniques, the following steps are usually taken: image processing, feature extraction, hypothesis generation, hypothesis verification, and pose refinement. In the previous sections we have discussed various image-processing and feature extraction techniques. Now we will focus our attention on how to generate scene-to-model match hypotheses, and on how to verify these hypotheses by comparing scene features with model features.

While local features yield more robust matching in the presence of occlusion and varying illumination conditions than the parameter vectors used for appearance-based recognition, they are also less distinctive for discriminating between similar objects. There may be many scene features that could match an individual model feature, or one scene feature that is present in multiple object models. In order to find the correct correspondence one needs more information, and this is typically obtained by considering relational properties of features to create *local feature sets*. A local feature set will now correspond to a unique set of features from an object model. By grouping features into local feature sets we reduce the uncertainty in assigning a set of features to a particular object model, thus facilitating the matching process. Each feature by itself will often be too simple and incapable of providing sufficiently discriminating information for identification. But when several features are combined into a local feature set, they will provide sufficient information to generate hypotheses for scene-to-model matching.

When 3-D objects contain vertices formed by the intersection of planar faces, such vertices together with these planar faces can be used for local feature sets. Other possibilities for local feature sets include three non collinear vertices, a straight edge and a non collinear vertex, or three non coplanar surfaces. Fig. 21 shows a local feature set for the object displayed in Fig. 19, being in this case the vertex 12 and the set of surfaces [F, G, K] that surround it. The only restriction on a local feature set is that it must contain the minimal grouping of features necessary to uniquely obtain the pose transform that relates an object in the scene to an object model.

A local feature set in a model is considered to match a scene feature set if each of the corresponding attribute values for every feature match. Different criteria are used for comparing attribute values, depending on whether they are qualitative or quantitative. Attributes that are qualitative are considered to match if their labels are identical, whereas quantitative attributes match if their values fall within a range of each other.

Once a local feature set is extracted from the scene, we need to compare it with our model base and try to establish a correspondence, or match. When the number of features that constitute a local feature set and the number of objects in the database are both small, then a straightforward approach to model matching consists of sequentially examining each model in turn, and accepting as possible solutions only those models for which there exists a correspondence of scene and model features. The number of scene-to-model comparisons needed to classify an object grows exponentially with the number of features in the model description. For this reason, sequential examination of the model base for feature matching is not computationally efficient for problems involving large libraries of objects or large numbers of features per object. Instead, cleverer model matching algorithms must be devised. Most approaches to model-based recognition and localization cast the scene-to-model correspondence part of the problem as a search problem, the search being for a consistent match between a set of model features and the local feature set extracted from the scene.

Recognition by Alignment

The comparison of scene with model feature sets will usually generate a set of hypotheses. Each of these hypotheses will constitute a possible solution to the spatial correspondence problem, providing as a result a transformation matrix that relates the pose of the object in the scene to the object model. Each hypothesis may relate a scene local feature set to different local feature sets in one or more object models. To further disambiguate among the possible scene-to-model correspondences, the rest of the features in the object model must be mapped back to the scene with the corresponding transformation. If enough nearby features are found to match between the scene and the model, then we can declare that there exists a scene-to-model match. The corresponding transformation matrix will provide information regarding the position and orientation of the matched object in the scene.

The method of breaking the recognition process into the two stages of hypothesis generation and verification is also known as *recognition by alignment* [74]. The alignment method can help overcome some of the most important difficulties in the recognition of 3-D objects in 2-D images: viewpoint invariance, error in attribute measurement, and partial occlusions.

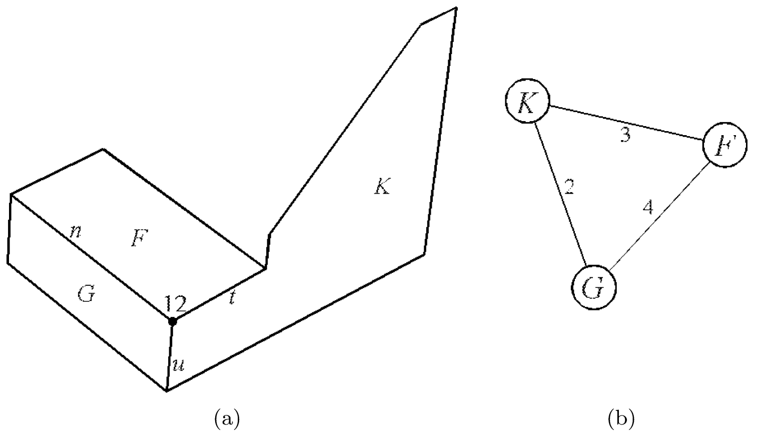


Figure 21. (a) Local feature set consisting of vertex 12 and the set of surfaces [*F, G, K*] that surround it. (b) The number on each arc is related to the length of the physical edge joining the corresponding pair of surfaces: (2) 38.1 mm, (3) 50.8 mm, and (4) 88.9 mm.

To handle the fact that any view of the object to be recognized can appear in the image or images taken from the scene, hypotheses are generated for matches between all possible scene local feature sets and model local feature sets. Once a match has been hypothesized, the verification stage allows for small errors in the measurement of the attributes. It is only required that the attributes of the verification features match their counterparts in the model within certain thresholds, usually established empirically. Occlusion, on the other hand, is handled by generating hypotheses using features from the model and the scene that are robust to partial occlusions, such as corner points or pieces of line segments. Even when the object is not entirely visible, if an entire local feature set can be located, along with the necessary verification features, a match hypothesis can be evaluated as true.

However, two major problems are present when using matching by alignment. First, the features used for building a scene local feature set can easily belong to different objects, to shadows, or the background. Although these sets of features most likely will not find a match in the model, the complexity of the search for a scene-to-model correspondence will be affected by the performance of the grouping technique used in the construction of a local feature set. The second problem arises from the fact that the error in locating the image features will propagate and magnify the error in the computation of the pose transform for a local feature set. As a result, the predicted pose of the object may not even approximate that of the scene.

We have argued that the number of features needed for object identification is bounded by the minimal grouping of features necessary to uniquely obtain the pose transform relating an object in the scene to an object model. Other researchers have proposed other bounds on the number of features needed for identification. In Ref. [8], for example, this number is said to be determined as a function of the probability that an erroneous match will occur. In this case, the number of matching features will depend on the number of model features, the number of scene features, the types of features used, and bounds on the positional and orientational errors in the data. The probability that a random arrangement of scene features will be considered as an object decrease when the number of features required to agree with the model increases, and a threshold f_0 on the fraction of model features required for recognition is obtained with the expression

$$f_0 \geq \frac{log \frac{1}{\delta}}{m\, log(1 + \frac{1}{ms\bar{c}})}$$

where m is the total number of model features, s is the total number of scene features, δ is defined as the probability that there will be $m\, f_0$ or more events occurring at random and $\bar{c}$ depends on the particular type of feature being matched and the bounds on the sensor error. For the case of 2-D line segments in 2-D images $\bar{c}$ has the form

$$\bar{c} = \frac{2\varepsilon_a \varepsilon_p \bar{\alpha} \bar{L}}{\pi D^2} + \frac{\varepsilon_a \varepsilon_p^2}{\pi D^2}$$

where ε_a and ε_p are bounds on the angular and positional error for a data feature (line segments in this case), $\bar{L}$ is the average edge length in the model, $\bar{\alpha}$ the average amount of occlusion of an edge in the scene, and D the linear extent of the image. The above formula can be simplified in the case where the features are vertices instead of edges, and has also been extended for edges in 3-D space.

Graph Matching and Discrete Relaxation

Once a local feature set has been extracted from the scene, it must be compared against sets of features from the object model. A sequential search for a set of features in the object model that produces an acceptable match hypothesis may be very time-consuming if the object model contains a large number of features or the number of object classes is large. To avoid sequential search during scene-to-model correspondence we can resort to various graph theoretic methods.

Using relational attributes, a set of features can be expressed as a graph $G = (V, E)$. The set V of nodes in the graph contains the features, and the edges in the set E represent the relations among features. The set of edges, E, is a subset of the Cartesian product $V \times V$, and each of these edges can be labeled according to the type of relational attribute: adjacency, parallelism, perpendicularity, and so on.

Producing a scene-to-model match hypothesis is equivalent to finding out that a subgraph of the graph representing the object model is isomorphic to the graph extracted from the scene. That is, there exists a one-to-one correspondence between the nodes in the two graphs preserving the graph structure. Given a model graph $G^M = (V^M, E^M)$ and a scene graph $G^S = (V^S, E^S)$, an isomorphism is a one-to-one function f of V^S onto V^M such that, for every edge $e^S_{ij} = \{v^s_i, v^S_j\}$ in E^S, the edge $e^M_{ij} = \{\ f(v^S_i),\ \ f(v^S_j)\} = \{v^M_i, v^M_j\}$ is in E^M. If the scene contains partial occlusions of the object to be recognized, then the graph G^S may contain fewer nodes than G^M, so that $|V^S| \leq |V^M|$. The problem then changes to that of subgraph isomorphism, that is, to find the largest subgraph of G^M isomorphic to G^S. In practice, it is only necessary to find the isomorphism in the object model graph for the subgraph corresponding to a local feature set extracted from the scene.

Subgraph isomorphisms can be detected by finding the maximal clique in a graph. A *clique* of size m of a graph is a completely connected subgraph of m nodes in the graph. Given the graphs of an object model G^M and a local feature set extracted from the scene G^S, we can construct an *association graph* G^A as follows. Each node v^A in G^A is the pair (v^M_i, v^S_j) such that the features v^M_i and v^S_j have the same attributes. An edge e^A_{12} exists in G^A between the nodes $v^A_1 = (v^M_{i_1}, v^S_{j_1})$ and $v^A_2 = (v^M_{i_2}, v^S_{j_2})$ if and only if the edges $e^M_{i_1 i_2}$ in G^M are the same as the edges $e^S_{j_1\,j_2}$ in G^S. This expresses the fact that the matches $(v^M_{i_1}, v^S_{j_1})$ and $(v^M_{i_2}, v^S_{j_2})$ are compatible.

Consider for example, the attribute graph presented in Fig. 20, and the local feature set from Fig. 21. Assume that the only attribute that we can extract from the scene for each surface is the number of adjacent surfaces. And assume, for this simple example, that we cannot differentiate between two surfaces if their number of adjacent surfaces is the same. The only attributes we can extract for an edge are its length and its delimiting surfaces. Following these restrictions, surfaces A, E, F, G, H, and I are all similar to four surrounding surfaces each; surface D to one; surface C to two; surface B to five; and surfaces J and K to seven each. Edges a, c, q, and s have length 1; edges e and u have length 2; edges d and t have length 3; edges h, i, l, m, n, o, and p have length 4; edges b and r have length 5; edges f, g, v, and w have length 6; edge k has length 7; and edge j has length 8. The nodes of the association graph G^A in Fig. 22 consist of all the possible combinations between the model features and the surfaces in the local feature set extracted from the scene. The edges in the association graph indicate the possibility of the pair of model features $(v^M_{i_1}, v^M_{j_2})$ matches with the pair of scene features $(v^S_{i_1}, v^S_{j_2})$. Observe that for two matches to be compatible, the length of the delimiting edges in both cases must match too. For example, in the association graph the vertices $F - F$ and $G - G$ are connected because the delimiting edge for the model features v^M_F and v^M_G has length 4, as well as the delimiting edge for the scene features v^S_F and v^S_G. On the other hand, even though there are edges e^M_{AJ} and e^M_{FK} in the attribute graph, these delimiting edges have different lengths, inhibiting the possibility of a match between the model features v^M_A and v^M_J and the scene features v^S_F and v^S_K.

Feasible hypotheses for the model matching problem are obtained by finding the largest completely connected subgraph in G^A, that is, the largest possible number of correct matches of features between the object model and the scene feature set. The most important drawback of the clique-finding problem, and consequently of the subgraph isomorphism problem, is that it is NP-complete, this is, its complexity grows exponentially in the number of nodes of the association graph. It has been shown however, that the graph isomorphism problem is solvable in polynomial time for graphs satisfying a fixed degree bound [75].

In our example the maximal clique is the one formed by the set of nodes $\{(v^M_F, v^S_F), (v^M_G, v^S_G), (v^M_K, v^S_K)\}$. Another maximal clique is given by the set $\{(v^M_F, v^S_F), (v^M_G, v^S_G), (v^M_j, v^S_K)\}$, and further verification steps may need to be carried out to discriminate between the two possibilities. This is referred to as *hypothesis verification*, and can be done after an initial computation of the pose of the hypothesized object in the scene is obtained, by searching for additional features in the scene that match features outside the local feature set in the hypothesized object rotated and translated in accordance with the hypothesized pose.

Another method for graph matching that is solvable in polynomial time is *bipartite matching*. Bipartite matching is the problem of dividing a graph into two different groupings, and to assign each node from one of the groupings to a node in the other grouping. If these two groupings correspond to the scene features and the model features in one graph G^{SM}, and if we draw arcs between the nodes in the two groups on the basis of their similarities, as in Fig. 23, a scene-to-model match hypothesis can be represented by the maximal bipartite graph that can be extracted from G^{SM}. Every scene feature v^S_i may bear similarities with many model features. However, for recognition to be correct, we want every scene feature to match a distinct model feature, that is, the matching between scene and model features must be injective.

We need to prune the graph G^{SM} by eliminating the injective-mapping violating arcs until we find a bipartite match. A sequential search for unacceptable arcs between scene and model nodes can become combinatorially extensive, but can be replaced by parallel techniques to make the computation feasible. One way of doing this is by *discrete relaxation* [18]. In general, relaxation in the computer vision context refers to a manner of iterative processing over a cellular structure in which decisions for each cell are made purely locally but subject to contents in the neighboring cells. Since the connection of a node to its neighboring nodes is fundamental to a graph, relaxation extends very naturally to computations over graphs. We must first create a graph by connecting each scene feature node to all possible model feature nodes on the basis of some similarity criterion (i.e., similarity of attribute values). These connections are then pruned by enforcing relational constraints, as observed in the scene, between different pairs of nodes in the graph. If the iterative application of this constraint enforcement leads to a unique arc between each node in the scene graph and the corresponding node in the model graph, we have accomplished scene interpretation via discrete relaxation. After relaxation, the assignment of scene to model features in GSM is unique for a sufficiently

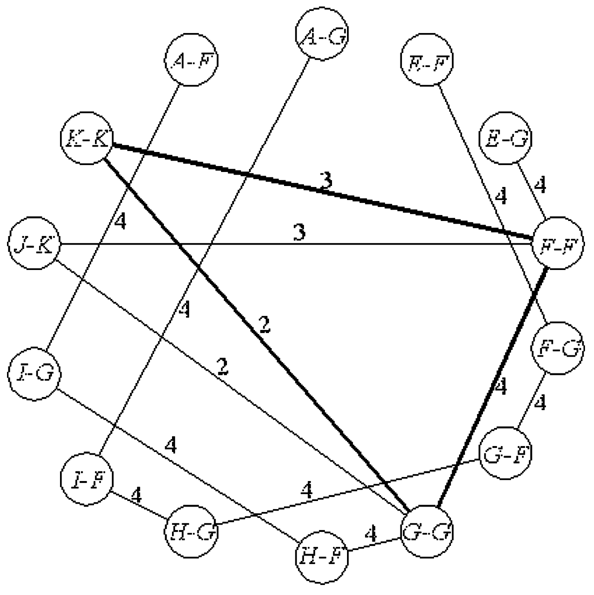


Figure 22. Graph matching by maximal cliques. The model and scene graphs are combined on an association graph, which shows the compatibility of individual feature matches. The maximal clique $\{(v_F^M, v_F^S), (v_G^M, v_G^S), (v_K^M, v_K^S)\}$ indicates the best match between the scene local feature set and the object model.

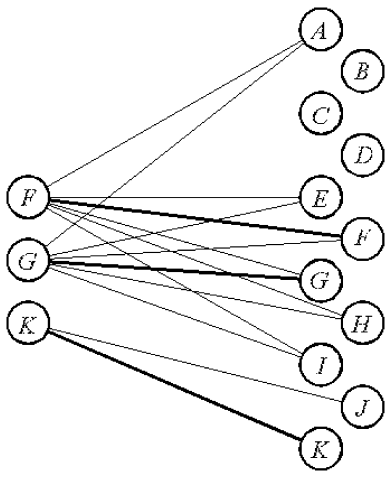


Figure 23. Bipartite Matching. The left column represents a local feature set, and the right column a set of model features. If a line joins a scene node with a model node, that means the two nodes have similar attributes. An acceptable hypothesis match between scene features and model features must be injective.

large number of model features, allowing us to compute a possible pose transform.

Scene-to-model correspondence using bipartite matching and discrete relaxation is particularly useful when the number of object classes in the model library is large, and when the objects involved possess a large number of features. Both these factors lead to large search spaces for object identification and pose computation and may render the problem too hard to solve using other model-based methods.

The Feature Sphere Approach

Another approach to solving scene-to-model correspondence is through the use of feature spheres in combination with the search tree of Fig. 24. The first h levels of the tree describe the different ways for h model features to match h scene features. The second part of the tree represents the verification stage that is implemented with the help of a feature sphere representation of objects. A path from the root to a leaf is a recognition sequence. An example of a 3-D object recognition system that uses this approach is the 3D-POLY system [11]. Using this technique, we can identify an object and compute its pose.

As depicted in Fig. 24, a hypothesis can be formed with h features in the hypothesis generation feature set, and the remaining $n - h$ features in the scene can then be used for verification. In principle, if a hypothesis is correct (i.e., the scene object is indeed an instance of the candidate model after transformation), then the remaining $n - h$ features in the scene should match their counterparts on the model using the same transformation. The process of matching

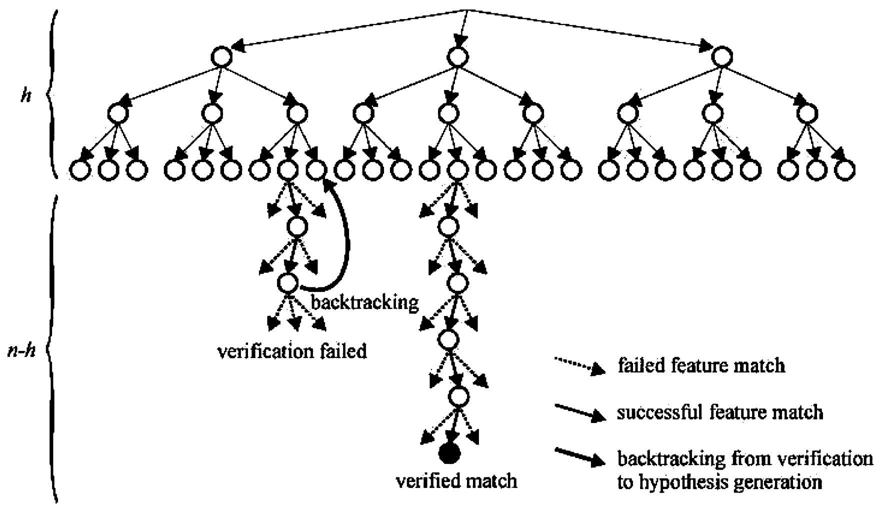


Figure 24. A data driven search tree is divided in two parts at level h. The first part represents the hypothesis generation stage while the second part represents the verification stage.

scene to model features cannot be performed in the opposite direction, since not all model features will be present on any scene in the case of occlusions.

If any of the remaining $n - h$ features in the scene cannot be matched to a model feature, that implies that the current hypothesis is invalid, because either the selected model object is not the right one, or the computed transformation is not accurate. Therefore, when a scene feature does not match any model feature under the candidate transformation, the matching algorithm must generate another transformation hypothesis. For this hypothesis generation scheme, the search is exhaustive over the model features in the sense that at every node shown on the hypothesis generation part in Fig. 24, a scene feature must be compared with all the features of the candidate object model. Therefore, at each node, the complexity is proportional to the number of features in the object model. The complexity for hypothesis generation is exponential in the number of features per hypothesis generation feature set. For rigid polyhedral objects, this number is typically 3, although its precise value depends upon how carefully the hypothesis generation feature sets are constructed. On the other hand, using the feature sphere data structure for object representation [11], the complexity of verification is made proportional to the total number of features on the model. The overall complexity of the recognition process is thus made a low-order polynomial in the number of features on the model, this being a substantial improvement over the exponential complexity of a brute force search.

A system that extends this idea for model matching to the use of hash tables for fast hypothesis generation is the MULTI-HASH system [13]. The advantage of this system over other model-based approaches resides in the learning stage, in which a multiple attribute hash table for fast hypothesis generation is built. By automatically constructing a hash table for object classification, the system is able to synthesize, under supervised training, the most discriminant features that separate one class of objects from another. During training in the MULTI-HASH system, the human operator specifies the correspondences between model feature sets and scene feature sets, as well as the object class. The system uses this information to construct models of uncertainty for the values of the attributes of object features. Using these uncertainty models, a decision tree is generated, which is transformed into a hash table for fast model matching.

Spatial Correspondence using Range Data

If range data are used for 3-D object recognition, then the transformation matrix between the scene and model feature sets can be computed by solving a set of equations of the form

$$[\begin{smallmatrix}\mathbf{p}_m^i\\1\end{smallmatrix}] = [\begin{smallmatrix}\mathbf{R} & \mathbf{t}\\0 & 1\end{smallmatrix}][\begin{smallmatrix}\mathbf{p}_s^i\\1\end{smallmatrix}]$$

when there is a one-to-one correspondence between the model point $\mathbf{p}_m^i$ and the scene point $\mathbf{p}_s^i$. The rotation submatrix $\mathbf{R}$ describes the orientation of the object in the scene, and the vector $\mathbf{t}$ represents the translation of the object from a reference coordinate frame in the model space to its position in the scene. A good estimate of the transformation matrix can be computed if a sufficient number of scene points can be related to their model counterparts.

In the approach used in Refs. [11] and [18] for pose computation, a solution for the rotation matrix $\mathbf{R}$ is computed by minimizing the sum of the squared errors between the rotated scene directional vectors and the corresponding model directional vectors. A directional vector $\mathbf{v}_s^i$ is the vector that describes the orientation in the scene of feature i. Similarly, $\mathbf{v}_m^i$ describes the orientation of the corresponding feature in the model space. The solution to this minimization problem gives an estimate of the orientation of the scene object with respect to the model object. The minimization problem can be expressed as

$$\frac{\partial}{\partial \mathbf{R}} i = 1 \sum \|\mathbf{R}\mathbf{v}_s^i - \mathbf{v}_m^i\|^2 = 0$$

To solve this minimization problem we resort to the use of quaternions [16]. A quaternion is a 4-tuple that describes the rotation around a unit vector $\hat{\mathbf{a}}$ through an angle θ:

$$\mathbf{Q} = [\begin{matrix} \cos(\theta/2) \\ \hat{\mathbf{a}}\sin(\theta/2) \end{matrix}]$$

Now, an ordinary directional vector $\mathbf{v}^{\mathrm{i}}$ would be represented in the quaternion form as $(0, \mathrm{v}^{\mathrm{i}^\top})^\top$, and its rotation by $\mathbf{Q}$ would result in the quaternion $(0, (\mathbf{R}\mathbf{v}^i)^\top)^\top$. By substituting quaternions for the various quantities in our minimization problem, it can be shown to be identical to

$$\frac{\partial}{\partial \mathbf{R}}(\mathbf{Q}\mathbf{A}\mathbf{Q}^\top) = 0$$

where $\mathbf{A}$ is given by

$$\mathbf{A} = i = 1 \sum \mathbf{B}_i \mathbf{B}_i^\top$$

$$\mathbf{B}_i = [\begin{matrix} 0 & -c_x^i & -c_y^i & -c_z^i \\ c_x^i & 0 & b_z^i & -b_y^i \\ c_y^i & -b_z^i & 0 & b_x^i \\ c_z^i & b_y^i & -b_x^i & 0 \end{matrix}]$$

and

$$\begin{aligned} \mathbf{b}^i &= \mathbf{v}_s^i + \mathbf{v}_m^i \\ \mathbf{c}^i &= \mathbf{v}_s^i - \mathbf{v}_m^i \end{aligned}$$

The quaternion $\mathbf{Q}$ that minimizes the argument of the derivative operator in our new differential equation is the smallest eigenvector of the matrix $\mathbf{A}$. If we denote this smallest eigenvector by the 4-tuple $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^\top$, then it follows that the rotational angle θ associated with the rotational transform is given by

$$\theta = 2\cos^{-1}(\alpha_1)$$

and the axis of rotation would be given by

$$\hat{\mathbf{a}} = \frac{(\alpha_2, \alpha_3, \alpha_4)^\top}{\sin(\theta/2)}$$

Then, it can be shown that the elements of the rotation submatrix $\mathbf{R}$ are related to the orientation parameters $\hat{\mathbf{a}}$ and θ by

$$\mathbf{R} = [\begin{matrix} a_x^2 + (1 - a_x^2)c_\theta & a_x a_y(1 - c_\theta) + a_z s_\theta & a_x a_z(1 - c_\theta) + a_y s_\theta \\ a_x a_y(1 - c_\theta) + a_z s_\theta & a_y^2 + (1 - a_y^2)c_\theta & a_y a_z(1 - c_\theta) - a_x s_\theta \\ a_x a_z(1 - c_\theta) - a_y s_\theta & a_y a_z(1 - c_\theta) - a_x s_\theta & a_z^2 + (1 - a_z^2)c_\theta \end{matrix}]$$

where $s_\theta = \sin(\theta)$, and $c_\theta = \cos(\theta)$.

Once the rotation submatrix $\mathbf{R}$ is computed, we can use again the matched set of scene and model points for the hypothesized match to compute the translation vector $\mathbf{t}$

$$\mathbf{t} = i = 1 \sum \tilde{\mathbf{p}}_m^i - \mathbf{R} i = 1 \sum \tilde{\mathbf{p}}_s^{-i}$$

Generalized Hough Transform

Another standard method for reducing the search for the pose of a hypothesized scene object is to use a voting scheme, such as the generalized Hough transform [44]. This method is an extension of the same voting scheme that we discussed for grouping together edge elements to extract lines or curves in an image. The Hough transform method can be generalized for model matching if the voting space comprises the viewpoint parameters. In the two-dimensional case, for example, the Hough space can be three- or four-dimensional: one dimension for the angle of rotation, two for the translation of the object along the u and v axes, and (if desired) another dimension for representing the scale at which an object appears on the scene. For 3-D object recognition the Hough space becomes six- or seven-dimensional (three dimensions for rotation, three for translation, and one for scaling).

The generalized Hough transform implementation for the classification of 2-D rigid objects from 2-D images consists of the following five steps:

1. Define an object template in terms of a discrete set of points from the set of features in the object model. Choose a reference point as the template center, and compute the angle α and distance r of the reference point relative to the points chosen on the template definition. Finally group these values into bins with the same gradient direction. This is, for each point in the template, compute the orientation of the boundary at that point, and store the r and α values in a table indexed by gradient value.
2. Define the Hough space in terms of the position, orientation, and scale of the expected objects in the image relative to the template. If for example we know the scale of the objects in the image is fixed, we need not include the scale dimension in the Hough space.
3. Run an edge operator, such as Sobel or Prewitt, over the image to extract the edge strength and direction at each pixel.
4. For every edge point (u_i, v_i) with edge orientation θ_i equal to the orienta tion of an edge in the template, look in the previously computed table for the possible relative locations (r, α) of the reference point. Compute the predicted template reference point

   $$u_c = u_i + sr\cos(\alpha + \phi), \quad v_c = u_i + sr\sin(\alpha + \phi)$$

   where s and ϕ are the discrete values of the scale and orientation being considered.
5. For each point from the scene features, we now have the coordinates (u_c, v_c), ϕ, and possibly s of a cell in the Hough space. Increment this cell by one count. The cell with the largest number of votes will provide the correct position, orientation, and scale of the object recognized from the scene.

The main advantage of the generalized Hough transform method is that it is somewhat insensitive to noise and occlusions. On the other hand, as in most model-based methods, a good geometric description of the objects to be recognized is necessary. Another drawback of this method is that the number of matches to be considered grows exponentially with the number of points in the object template. To overcome this problem, variants of the generalized Hough transform method have been purposed, such as geometric hashing [76]. But the most important drawback of this approach is that in order to have reasonable accuracy for the computed pose one must sample the Hough space quite finely, and that leads to the testing of enormous numbers of possibilities. The method is then equivalent to correlating the object model with the scene model over all possible poses and finding the best correlation. One can argue that this is the same drawback as the one seen for the appearance-based methods discussed earlier.

SUMMARY

Object recognition entails identifying instances of known objects in sensory data by searching for a match between features in a scene and features on a model. The key elements that make object recognition feasible are the use of diverse sensory input forms such as stereo imagery or range data, appropriate low level processing of the sensory input, clever object representations, and good algorithms for scene-to-model hypothesis generation and model matching.

Whether data acquisition takes place using video images or range sensors, an object recognition system must pre-process the sensory data for the extraction of relevant features in the scene. Once a feature vector is obtained, the problem now is that of correspondence. Provided a training session has taken place, a search for a match between model features and scene features is performed. A consistent match and the corresponding transformation give a solution to the problem of object recognition.

BIBLIOGRAPHY

1. G. J. Agin. Vision systems. In S. Y. Nof, editor, *Handbook of Industrial Robotics*, pages 231–261. John Wiley & Sons, New York, 1985.
2. R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, **18**(1): 67–108, Mar. 1986.
3. N. Ayache and O. Faugeras. HYPER: A new approach for the recognition and positioning of two-dimensional objects. *Pattern Recogn.*, **22**(1): 21–28, 1986.
4. R. A. Brooks. Model-based three-dimensional interpretations of two-dimensional images. *IEEE Trans. Pattern Anal. Machine Intell.*, **5**(2): 140–150, Mar. 1983.
5. D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artif. Intell.*, **31**(3): 355–395, Mar. 1987.
6. T. D. Alter and W. E. L. Grimson. Fast and robust 3D recognition by alignment.In *Proc. 4th IEEE Int. Conf. Comput. Vision*, pages 113–120, Berlin, 1993.
7. W. E. L. Grimson and D. P. Huttenlocher. On the sensitivity of the Hough transform for object recognition. *IEEE Trans. Pattern Anal. Machine In-tell.*, **12**(3): 255–274, Mar. 1990.
8. W. E. L. Grimson and D. P. Huttenlocher. On the verification of hypothesized matches in model-based recognition. *IEEE Trans. Pattern Anal. Machine Intell.*, **13**(12): 1201–1213, Dec. 1991.
9. A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *IEEE Trans. Pattern Anal. Machine Intell.*, **28**(3): 416–431, Mar. 2006.
10. R. C. Bolles and P. Horaud. 3DPO: A three-dimensional part orientation system. *Int. J. Robot. Res.*, **5**(3): 3–26, 1986.
11. C. H. Chen and A. C. Kak. Robot vision system for recognizing objects in low-order polynomial time. *IEEE Trans. Syst., Man, Cybern.*, **18**(6): 1535–1536, Nov. 1989.
12. A. J. Vayda and A. C. Kak. A robot vision system for recognition of generic shaped objects. *Comput. Vis. Image Und.*, **54**(1): 1–46, Jul. 1991.
13. L. Grewe and A. C. Kak. Interactive learning of a multiple-attribute hash table classifier for fast object recognition. *Comput. Vis. Image Und.*, **61**(3): 387–416, May 1995.
14. P. J. Flynn and A. K. Jain. BONSAI: 3D object recognition using constrained search. *IEEE Trans. Pattern Anal. Machine Intell.*, **13**(10): 1066–1075, Oct. 1991.
15. T. J. Fan, G. Medioni, and R. Nevatia. Recognizing 3D objects using surface descriptions. *IEEE Trans. Pattern Anal. Machine Intell.*, **11**(11): 1140–1157, Nov. 1989.
16. O. D. Faugeras and M. Hebert. Representation, recognition, and localization of 3D objects. *Int. J. Robot. Res.*, **5**(3): 27–52, 1986.
17. W. E. L. Grimson and T. Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *Int. J. Robot. Res.*, **3**(3): 3–35, 1984.
18. W. Y. Kim and A. C. Kak. 3D object recognition using bipartite matching embedded in discrete relaxation. *IEEE Trans. Pattern Anal. Machine Intell.*, **13**(3): 224–251, Mar. 1991.
19. L. G. Shapiro and H. Lu. Accumulator-based inexact matching using relational summaries. *Mach. Vision Appl.*, **3**(3): 143–158, 1990.
20. A. S. Mian, M. Bennamoun, and R. Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Trans. Pattern Anal. Machine Intell.*, **28**(10): 1584–1601, Oct. 2006.
21. M. F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, and S. Dickinson. Object recognition as many-to-many feature matching. *Int. J. Comput. Vision*, **69**(2): 203–222, Aug. 2006.
22. O. Faugeras. *Three-Dimensional Computer Vision. A Geometric Viewpoint*. The MIT Press, Cambridge, 1993.
23. R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximum cliques. *IEEE Trans. Pattern Anal. Machine Intell*. **11**(11): 1168–1180, Dec. 1989.
24. K. Ikeuchi and T. Kanade. Automatic generation of object recognition programs. *Proc. IEEE*, **76**(8): 1016–1035, Aug. 1988.
25. M. Kirby and L. Sirovich. Application of the Karhunen-Loéve procedure for the characterization of human faces. *IEEE Trans. Pattern Anal. Machine Intell.*, **12**(1): 103–108, Jan. 1990.
26. M. Turk and A. Pentland. Eigenfaces for recognition. *J. Cognitive Neu-rosci.*, **3**(1): 71–86, 1991.
27. H. Murase and S. K. Nayar. Visual learning and recognition of 3D objects from appearance. *Int. J. Comput. Vision*, **14**(1): 5–24, Jan. 1995.
28. T. V. Pham and A. W. M. Smeulders. Sparse representation for coarse and fine object recognition. *IEEE Trans. Pattern Anal. Machine Intell*. **28**(4): 555–567, Apr. 2006.
29. A. Kosaka and A. C. Kak. Stereo vision for industrial applications. In S. Y. Nof, editor, *Handbook for Industrial Robotics*, pages 269–294. John Wiley & Sons, New York, 1999.
30. H. S. Yang and A. C. Kak. Edge extraction and labeling from structured light 3D vision data. In S. Haykin, editor, *Selected Topics in Signal Processing*, pages 148–193. Prentice Hall, Englewood Cliffs, 1989.

31. J. Clark, A. M. Wallace, and G. L. Pronzato. Measuring range using a triangulation sensor with variable geometry. *IEEE Trans. Robot. Automat.*, **14**(1): 60–68, Feb. 1998.

32. W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Anal. Machine Intell.*, **13**(9): 891–906, 1991.

33. B. M. T. Haar Romeny. *Front-End Vision and Multi-Scale Image Analysis*. Springer-Verlag, 2003.

34. D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman, San Francisco, 1982.

35. F. Heitger. Detection using suppression enhancement.Technical Report TR 163, CTL, Swiss Fed. Inst. Tech., 1995.

36. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Electrical Engineering and Computer Science Series. MIT Press, Cambridge, 1992.

37. M. Villamizar, A. Sanfeliu, and J. Andrade-Cetto. Computation of rotation local invariant features using the integral image for real time object detection. In *Proc. 18th IAPR Int. Conf. Pattern Recog*., volume **4**, pages 81–85, Hong Kong, Aug. 2006. IEEE Comp. Soc.

38. H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proc. 9th European Conf. Comput. Vision*, volume **3951** of*Lect. Notes Comput. Sci.*, pages 404–417, Graz, 2006. Springer-Verlag.

39. C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proc. IEEE Int. Conf. Comput. Vision*,page 555, Bombay, Jan. 1998.

40. P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, **57**(2): 137–154, May 2004.

41. D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, Engle-wood Cliffs, 1982.

42. A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume**1–2**. Academic Press, New York, 1982.

43. P. V. C. Hough.Methods and means for recognizing complex patterns. U.S. Patent No. 3,069,654, 1962.

44. D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recogn.*, **13**(2): 111–122, 1981.

45. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Comput. Vision*, **1**(4): 321–331, 1987.

46. V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert. Minimal surfaces based object segmentation. *IEEE Trans. Pattern Anal. Machine Intell*. **19**(4): 394–398, Apr. 1997.

47. D. Jacobs. Robust and efficient detection of salient convex groups. *IEEE Trans. Pattern Anal. Machine Intell.*, **18**(1): 23–37, Jan. 1996.

48. D. G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, MA, 1985.

49. C. G. Harris and M. Stephens. A combined corner edge detector. In *Proc. Alvey Vision Conf.*, pages 189–192, Manchester, Aug. 1988.

50. H.P. Moravec. Towards automatic visual obstacle avoidance. In *Proc. Int. Joint Conf. Artificial Intell.*,page 584, Cambridge, 1977.

51. P. R. Beaudet. Rotational invariant image operators. In *Proc. 4th IAPR Int. Conf. Pattern Recog.*, pages 579–583, Tokyo, 1978.

52. J. Shi and C. Tomasi. Good features to track. In *Proc. 9th IEEE Conf. Comput. Vision Pattern Recog.*, pages 593–600, Seattle, Jun. 1994.

53. T. Lindeberg. Feature detection with automatic scale selection. *Int. J. Comput. Vision*, **30**(2): 79–116, 1998.

54. D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. IEEE Int. Conf. Comput. Vision*, pages 1150–1157, Corfu, Sep. 1999.

55. K. Mikolajczyk. itDetection of Local Features Invariant to Affine Transformations. PhD thesis, Institut National de Polytechniques de Grenoble, 2002.

56. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, **60**(2): 91–110, 2004.

57. K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. 7th European Conf. Comput. Vision*, volume **I**, pages 128–142, Copenhagen, 2002. Springer-Verlag.

58. A. Baumberg. Reliable feature matching across widely separated views. In *Proc. 14th IEEE Conf. Comput. Vision Pattern Recog.*, pages 1774–1781, Head Island, Jun. 2000.

59. T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure. *Image Vision Comput.*, **15**(6): 415–434, 1997.

60. T. Tuytelaars and L. J. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. InM. Mirmehdi and B. T. Thomas, editors, *Proc. British Machine Vision Conf.*, Bristol, 2000.

61. L. Van Gool, T. Moons, and Ungureanu D. Affine/photometric invariants for planar intensity patterns. InB. Buxton andR. Cipolla, editors, *Proc. 4th European Conf. Comput. Vision*, volume **1065** of *Lect. Notes Comput. Sci.*, pages 642–651, Cambridge, Apr. 1996. Springer-Verlag.

62. L. M. J. Florack, B. M. Haar Romeny, J. J. Koenderink, and M. A. Viergever. Scale and the differential structure of images. *Image Vision Comput.*, **10**(6): 376–388, Jul. 1992.

63. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Machine Intell.*, **27**(10): 1615–1630, 2005.

64. B. K. P. Horn. Extended Gaussian images. *Proc. IEEE*, **72**(12): 1671–1686, Dec. 1984.

65. S. K. Nayar, S. Nene, and H. Murase. Subspace methods for robot vision. *IEEE Trans. Robot. Automat.*, **12**(5): 750–758, Oct. 1996.

66. J. L. Edwards. An active appearance-based approach to the pose estimation of complex objects. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, volume **3**, pages 1458–1465, Osaka, Nov. 1996.

67. J. L. Edwards and H. Murase. Coarse-to-fine adaptive masks for appearance matching of occluded scenes. *Mach. Vision Appl.*, **10**(5): 232–242, 1998.

68. A. Martinez. Face image retrieval using HMMs. In *Proc. IEEE CVPR Workshop Content-based Access Image Video Lib.*, pages 25–39, Fort Collins, Jun. 1999.

69. M. Pontil and A. Verri. Support vector machines for 3D object recognition. *IEEE Trans. Pattern Anal. Machine Intell.*, **20**(6): 637–646, 1998.

70. A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Trans. Pattern Anal. Machine Intell.*, **23**(4): 349–361, 2001.

71. H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. Pattern Anal. Machine Intell.*, **20**(1): 23–38, 1998.

72. R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object

detection. In *Proc. 25th German Pattern Recog. Sym.*, pages 297–304, Magdeburg, Sep. 2003.

73. P. A. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proc. IEEE Int. Conf. Comput. Vision*, pages 734–741, Nice, Oct. 2003.
74. D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. J. Comput. Vision*, **5**(2): 195–212, 1990.
75. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.
76. P. Suetens, P. Fua, and A. J. Hanson. Computational strategies for object recognition. *ACM Computing Surveys*, **24**(1): 5–61, Mar. 1992.

See: Artificial Intelligence, Automatic Guided Vehicles, Computer Vision, Image Segmentation, Image Sensors, Object Detection, Robot Vision, Stereo Image Processing.

JUAN ANDRADE-CETTO
MICHAEL VILLAMIZAR
Institut de Robótica i Informática Industrial, Barcelona, Spain