Chapter 16

II - Object recognition

16.1 II - Introduction

This chapter discusses approaches to object recognition. Usually the goal is not only to recognize objects in intensity or range images, but also to determine their pose, i.e. their 3D position and orientation.

As we will see the complexity of the recognition problem depends on whether the 'what' and 'where' problems can be separated.
A second factor determining the complexity of recognition is the layout of the scene. Recognition of an isolated object against a homogeneous background is far easier than recognition in cluttered environments.
The difficulty of recognition is also determined by the degree to which inference of three-dimensional information from two-dimensional data is necessary. Recognizing planar shapes is easier than recognizing 3D shapes, unless range data would be available.

Much of the problems arise from the fact that information is lost under projection, which is a non-invertible operation. Any two-dimensional projection could have arisen from an infinity of three-dimensional scenes. Under very simple viewing conditions the loss of information can be kept to a minimum also. Such conditions are typical for machine vision systems in industry. If the object is completely flat and no surface characteristics have to be judged, all shape information can be obtained from a binary silhouette image if looked from a perpendicular direction. Backlighting combined with thresholding yields a simple and robust basis for recognition if the environment can sufficiently be constrained. Here we will focus on more advanced methods.

The methods to be discussed are organised as follows:

1. pattern recognition approaches,
2. model-based approaches,
3. appearance-based approaches,
4. hybrid approaches.

The distinction between these categories will become clearer as we proceed. It should be emphasised that this classification is not complete and by no means univocally adopted.

16.2 II - Pattern recognition approaches

Approach

As already pointed out, the whole process of shape description and feature extraction is one of bandwith reduction coupled to an increase in information content per bit. At the end of such process one usually ends up with a list of features, a *feature vector*. This vector may contain a few invariant moments, the correlation results for some invariant description with all the model descriptions, etc.

If the discriminating power of the features is sufficient, very simple procedures for classification may suffice. For instance, in the case of the correlation data, the maximal correlation could be selected as the one pointing at the correct model. Similarly, an invariant feature with high discriminant power can be used for *direct indexing*, using a lookup table where for each value of the feature it is specified which is the only candidate object.

Usually, a more sophisticated technique will be called for. Setting out all the sample feature vectors belonging to a class, hopefully a cluster is formed in the feature space (cfr. chapter ??). Typically the clusters for different classes will overlap. Strategies from *pattern recognition* can be utilized in that case.

There are two large directions in pattern recognition theory: *supervised* and *unsupervised techniques*. Unsupervised techniques look for clusters in the data and the user has to label them later (i.e. to attach an interpretation to the clusters). With supervised techniques the user decides on the different classes beforehand, supplies the system with sample data for each class (the training set), whereafter it will model the clusters (e.g. calculate an optimal normal fit) and set the decision rules accordingly. In fact, both approaches normally require iterative refinements, e.g. class mergers and splits in the former case and the elimination of outliers in the latter.

The class assignment of new measured data (i.e. the actual classification) can be guided according to a wide gamut of strategies. A detailed discussion it out of the scope of this course and can be found in pattern recognition textbooks.
This can e.g. be organised following very simple procedures, e.g. by measuring Euclidean distance between the image vector and the centroid of the model clusters and choosing the object class yielding the smallest distance ("nearest neighbour classification"). Although straightforward, one might be more careful in choosing units of measurement along the different axes. Suppose we have to measure the distance between two cities on a map. If our measurement should have any meaning at all, we should make sure that the map is drawn with the same yardstick horizontally and vertically. Not accidentally this happens to be the case with most maps... Choosing appropriate yardsticks is not purely a matter of choosing the same units, however. Somehow these yardsticks should indicate the relative importance of the different features, as expressed e.g. by the Mahalanobis distance

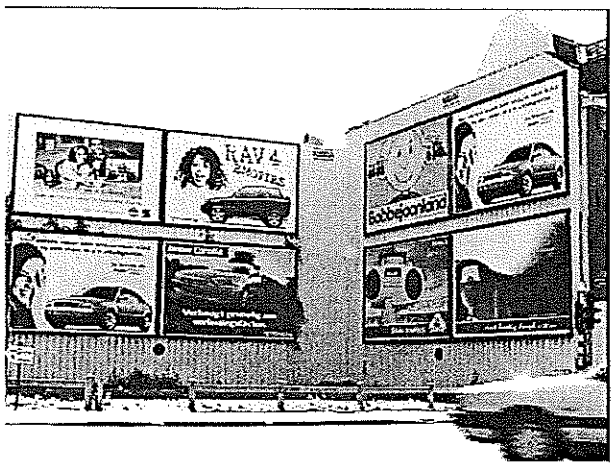


Figure 16.1: *Eleven different types of advertisement panels that were in the set of images to be recognised.*

(chapter ??).
Indeed, if information about the probability density distributions of the features are available, the metric problem can be solved by adapting the metric accordingly or it can be avoided altogether by "maximum-likelihood classification" (e.g. the Bayes classifier) where a feature vector is assigned to the most probable class or "minimum-average-loss classification" (e.g. the minimum-average-loss Bayes classifier) where misclassifications are assigned a cost that might be different for different classes. Maximal likelihood classifiers are optimal, but often difficult to implement efficiently.

An alternative for statistical classifiers are neural networks. Their advantage is that complicated decision boundaries can be constructed in feature hyperspace, without having to spell them out explicitly. This comes at a price in that the decision stage remains somewhat of a black box. A more recent and in many tests more performant approach are the so-called Support Vector Machines. These are more strongly rooted in traditional statistical pattern recognition.

It goes without saying that the success of classification hinges on the choice of relevant features. Good features should be stable within a class and should maximally differ between classes (also see our previous discussions on PCA, ICA, and discriminant analysis). Invariants are often good features in that they eliminate irrelevant variations.

Example

As example for a pattern recognition kind of recognition we take the classification of advertisement panels, as they can be found in the streets of most cities. Fig. 16.1 shows an example with 8 such panels. Marketing organisations are interested to know where clients and their competitors advertise, how often, etc. To that end it is useful to have a system that automatically finds and then recognises different types of such panels. Here focus is on the recognition, not on the segmentation of the panels from their background (a difficult

problem in its own right).

In order to recognize such advertisement panels in street scenes the approach has to be immune against changes in illumination and viewpoint. Fig. 16.2 shows 6 images with the same panel. The patterns show quite some variation in the images: not only are they deformed between views, also the overall brightness and colour saturation are different in different images.

For the recognition the geometric/photometric moment invariants of the previous chapter were used. A test was carried out with over 500 images taken of 11 different types of panels. These are shown in fig. 16.3. The panels were segmented manually (as this was not a test on segmentation but on the performance of these moment invariants).

Different feature vectors were composed and classification results based on these different vectors were compared. Classification was done using a 'jacknifing' procedure. This means that samples are randomly drawn from the set and are classified. The other samples are each time used to train the classifier. In this case this was a simple nearest-neighbour method. Jacknifing is typically used when the set of samples is too small to be splitted into a training and a test set. The results can be shown to prove the classification performance in the limit. The feature vector that came out as the most successful one is

$$\left(C_{12}^{3(GB)}, C_{12}^{4(RB)}, \tilde{C}_{02}^{(GB)}, \tilde{C}_{02}^{(RG)}, C_{02}^{(RB)}\right)$$

where the definitions of these invariants can be found in chapter ??. With this feature vector a correct classification rate of 96% was achieved. It is useful to mention that even better results are possible with moment invariants if a few more modifications are applied, which we cannot go into here.

Discussion

The traditional type of pattern recognition approach as just described is especially useful when the objects to be recognised are completely visible. Even when some of the samples in the set of advertisement panels were partially occluded by traffic signs or passing cars and could nevertheless be recognized correctly, occlusion will often pose problems for these methods. Typically, it is assumed that all the components in the feature vector can be extracted from all samples.

16.3 II - Model-based approaches

16.3.1 Structural methods

Approach

Structural methods use topological constraints (syntactic rules) imposed on the features (semantic primitives). Features are described together with their spatial relationships. Graphs are well-suited for describing such *structural* or *relational models*. Recognition in those cases is a question of (sub)graph matching (isomorphisms, maximal cliques,...).



Figure 16.2: *Eleven different types of advertisement panels that were in the set of images to be recognised.*



Figure 16.3: *Eleven different types of advertisement panels that were in the set of images to be recognised.*

Linguistics have strongly influenced structural methods which are often formulated in terms of "grammars" and "sentence parsing".

The features (primitives) utilized with the structural approach are most often of a simple parametric or qualitative nature: circular arcs, straight segments, convex and concave parts, holes, etc. If range data are available, similar feature can be used for 3-D solids: plane segments, spherical or cylindrical patches, etc. The Gaussian sphere was introduced as a useful tool for their detection (chapter ??). These primitives can then be used as the nodes in a graph, where the edges indicate adjacency relations possibly with additional labels indicating concepts like "to the right of", "below", etc.
The graph structure naturally follows from the feature extraction itself in the case of so-called "axis transforms". The resulting graph structure is usually referred at as the "skeleton" of the shape and its branches (arcs) more or less correspond to the shape's natural parts. Around each point of the "SAT"-skeleton (Symmetric Axis Transform) there exists a circle which is tangent to the boundary in at least two points. The SAT consists of two parts: the structure of the skeleton and the radius of the corresponding circle at each of its points. Emphasis is on the arcs rather than on the nodes of the graph. The skeleton may be used to derive information about the global shape of the object, e.g. in character recognition. The computation of the skeleton is time consuming and the result is very sensitive to noise. Alternative axis transforms like Smoothed Local Symmetries (SLS) have been devised in order to overcome the SATs main difficulties. They should bridge the gap between region and contour-based approaches. These techniques have not entered industrial vision systems yet.
In dealing with the structural relationships, special attention is often paid to certain Gestalt-like *groupings*, e.g. symmetrical, parallel, or collinear edge segments. The underlying *non-accidentalness assumption* is that such configurations can hardly have arisen by accident and therefore point at such groupings being present in the real shape.

Structural methods strongly rely on model data while extracting the features, in contrast to invariant based approaches. The recognition problem is also heavily intertwined with the *localization problem* with this kind of algorithms. Localization means that the pose of the object (position and orientation) are extracted.

The *top-down* or *model-driven* version works as follows. Assuming that both the model and the scene are composed of features such as vertices, holes, straight lines, circular arcs, ... and a specified object is searched for, then one knows in advance what types of features to look for. This means that problem-specific information is used to reduce the feature search in the image. After making an initial feature match hypothesis, the arrangement of other features in the model is used to verify the assumption: it is checked whether the other features in the model can also be found at the predicted places and in the appropriate combinations (according to some syntactic rules and with the possibility that they are occluded). Notice that an assumption is not only about type but also about pose. If additional features are in place, i.e. if they are found at the predicted positions this reinforces the assumption and a refined hypothesis about type and pose can be made. The algorithm cycles through this *hypothesize-and-verify* loop. Looking selectively for a few, specific intial features is an interesting strategy if one only wants to find a piece of a certain

type without being interested in the other objects present.
Bottom-up or *data-driven search* is a second strategy. First, all features in the scene are searched. No problem-specific knowledge is used at that stage. That also means that features will be searched which are not relevant to the application. Then a particularly salient one is selected and all possible models to which it could belong are listed. By considering several features more and more false candidate models can be eliminated. Again, the recognition-localization is realized through the hypothesize-and-verify paradigm. Bottom-up approaches can be interesting if it is not specified which type of object to look for or when all objects in the scene should be recognized. This approach tries to minimize the time needed to analyze the total scene, rather than to minimize the time needed to find an object of a predefined type.
Also hybrid forms have been proposed, which combine top-down and bottom-up techniques.

Examples

Discussion

complicating factor is that pose and type determination are intertwined.

16.3.2 Invariant signatures

Approach

One way of using invariants for recognition is the calculation of *invariant signatures*. An invariant signature expresses an invariant as a function of an invariant parameter. A well-known example is curvature as a function of arclength. This Euclidean signature is known to unambiguously characterise a shape, irrespective of translations or rotations in the image plane.

Examples

Invariant signatures for planar curves First we give an example of signatures for the recognition of planar curves. Consider fig. 16.4. The goal is to recognize shapes as the one shown, irrespective of viewpoint. The figure shows on its top row a perpendicular (left) and oblique view (right) of the same swan shape. There are clear distortions between both, but perspective effects seem to remain small. It is assumed that this is typical for the application. In that case, the deformations can be modelled sufficiently well as affine transformations. Recognition will be based on affine, invariant signatures for the contours. Since the objects contrast well with the background, we can expect to find the complete outlines of the objects. This is born out by the results, as shown on the bottom row of fig. 16.4, where spline approximations of the extracted edges are given. Having the complete outlines implies that it is possible to calculate the areas of these shapes. This is interesting, as will soon become clear.

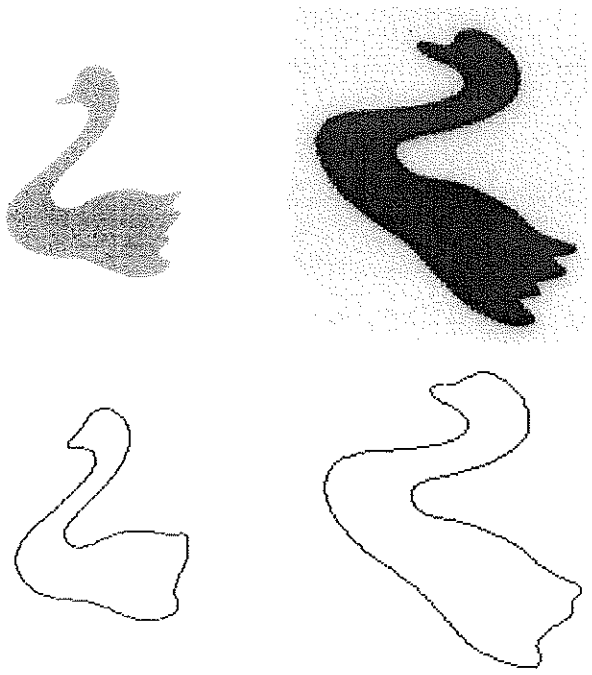


Figure 16.4: **Top row.** *Left: model contour of swan shape, right: image contour taken under pseudo-perspective conditions (i.e. orthographic projection + a scaling) from an oblique viewpoint.* **Bottom row.** *Spline approximations of the contours.*

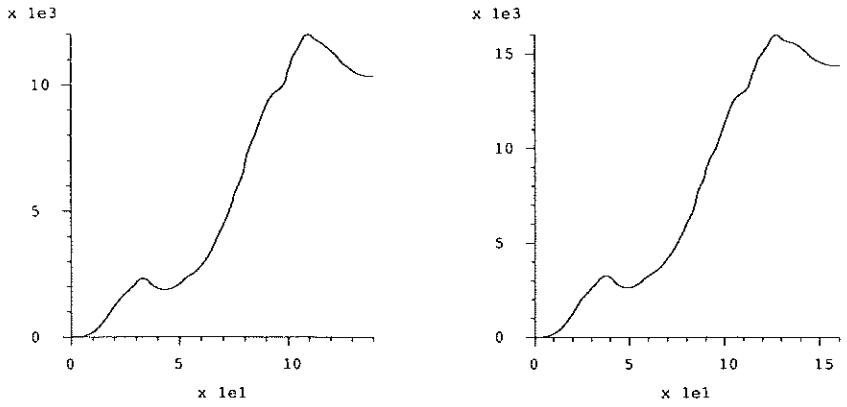


Figure 16.5: *Invariant signatures for the contours in fig. 16.4; Left: for the shape on the left; Right: for the shape on the right.*

Specifically,

$$\int_{t_2}^{t_1} \mathrm{abs}\left(|\mathbf{x}_1 - \mathbf{x}_2 \;\; \mathbf{x}_1^{(1)}|\right) \;\; dt$$

is expressed as a function of

$$\int_{t_2}^{t_1} \mathrm{abs}\left(|\mathbf{x}_1^{(1)} \; \mathbf{x}_1^{(2)}|^{\frac{1}{3}}\right) dt \;\; .$$

In these expressions $\mathbf{x}_2$ is a fixed point, which is also used as the starting point of the integrations (t_2 is the parameter value at $\mathbf{x}_2$). On the other hand, $\mathbf{x}_1$ is a point that varies with t and t_1 represents the corresponding parameter value at any particular position of $\mathbf{x}_1$. Hence, for every position of $\mathbf{x}_1$, we obtain two values, one for each integral. The former is expressed as a function of the latter. The example is a bit special in two respects. First, as invariant for the signature (i.e. as affine counterpart to curvature in the Euclidean signature) we use just another invariant parameter. Since using a parameter introduces the problem of choosing a starting point anyway, we can just as well use that point as starting point for a second parameter as well, without additional overhead and with the numerical advantage that integration brings. Second, the invariant parameters are only relative invariant, i.e. they change by a factor under the affine transformations. This factor is simply $|A|$, with A the 2×2 matrix in the affine transformation, as usual. Since this determinant expresses the factor by which the affine transformation changes areas, its effect can be eliminated by dividing both expressions by the overall area. This yields – for a particular choice of $\mathbf{x}_2$ – the normalized descriptions shown in fig. 16.5. These are theoretically invariant and in practice very similar indeed! Apart from the problem of choosing corresponding starting points $\mathbf{x}_2$, recognition now amounts to comparing these descriptions. As starting point, one can e.g. pick an inflection. One then only has to try different inflections.

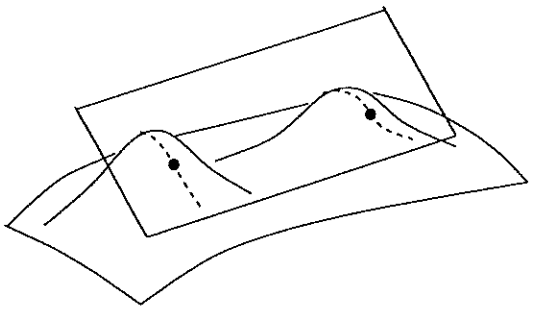


Figure 16.6: *Bitangent planes can roll over the surface, thereby describing pairs of bitangent curves.*

Invariant signatures for 3D surfaces Suppose we can acquire 3D data of the object's shape. This could e.g. be done through a stereo system or a system that uses structured light. The problem that we discuss here is not so much recognition *per se*, but one that is strongly related: the automatic registration of partial 3D data patches in order to build a complete model. 'Registration' here means the fitting together of different parts of the surface. We assume that the different parts or patches overlap. Registration is then a matter of recognizing the overlapping parts in the two patches that we want to fit together at any given time.

the registration problem

Indeed, three-dimensional (3D) models of objects usually are not acquired in one piece. Most reconstruction methods will generate partial 3D descriptions, which subsequently have to be combined into a single model. An important step in this procedure is the aforementioned registration of the partial descriptions, henceforth called 'patches'. There are good algorithms to do this once a good approximation of the relative position of the patches is known. That is, once patches are close to eachother these algorithms can carry out a precise docking, which may be a nerve-wrecking experience if one has to do it interactively. Starting from very different initial positions they might not converge to the correct solution. Providing such initial crude registration automatically is the goal of the technique reported here.

In order to quickly match 3D patches, it is useful to first detect special points or curves on them. These can serve as landmarks that assist in the matching. Invariant descriptions of the landmarks ease the matching even further. The invariants make it possible to compare landmarks on different patches irrespective of the relative positions, orientations, and possibly deformations of the patches. Once correspondences between landmarks have been established, the transformation that maps one surface patch on the other can be estimated. One of the existing registration algorithms can then refine the registration.

Bitangent curves as landmarks

Bitangent points on a 3D surface are point pairs that have the same tangent plane. Such points actually lie on curves, as the bitangent plane can 'roll over the surface'. This is illustrated in fig. 16.6. We will refer to such curves as 'bitangent curves'. In the generic case, the bitangent points correspond to a pair of distinct points on the two curves. They

will be referred to as 'bitangent point pairs'. Bitangent curves are good landmarks, as they are invariant under Euclidean, affine, and even projective transformations. They typically are sufficiently numerous and well spread over the surface. Moreover, the bitangent curve pairs can be extracted efficiently (in a dual space).

In order to support efficient curve matching, the bitangent curves are characterised by invariant signatures, as we did before for planar contours. Here invariance under 3D rotations and translations suffices. In general, building invariant signatures for space curves is a lot more difficult than for planar curves. But that's where the choice of these special bitangent curves comes in. Points on bitangent curves come in pairs. Moreover, they are linked through a bitangent correspondence. This means that a single parameterisation can be used at both points of a bitangent pair. Hence, for every value of a chosen parameter, i.e. for every position of the bitangent plane, there are 2 points with their coordinates and the derivatives thereof to build invariants from. In particular, the two points and the their tangent vectors to the curves all live in the same bitangential plane. Hence, if we restrict ourselves to these data, plane geometry applies. Of course, there are also dependencies in that the tangent vectors in a pair of bitangent points are coplanar. For the invariant signatures, the distance between bitangent points can be expressed as a function of the arclength, e.g. of the longest of the two curves. Similarly, the angle between the tangent vectors to the curves can be used.

Matching patches now boils down to matching their bitangent curve pairs, i.e. comparing the invariant signatures of these curves. As criterion, we use the $L2$-norm. As it may very well happen that only parts of bitangent curves are found on each of the patches, the signatures are divided into segments and these segments are matched. Matching segments are then extended as far as possible. A signature segment corresponds to a pair of segments on a bitangent curve pair and hence its endpoints define four points on the surface patch. If a pair of signature segments is matched successfully, this suggests a match between 4 points on the two surface patches. These typically yield enough information to obtain a crude estimate for the Euclidean transformation between the patches. In order to render the process more efficient and more robust, the search for matching signatures starts with the longest bitangent curves (where length is measured in the dual space).

After a good match is found, it is checked by applying the corresponding transformation and by calculating the closest point on the other patch for a subset of points. If a sufficiently large number of points finds a partner point at close range, then the crude registration is accepted.

An example

Fig. 16.7 shows 6 patches of a Greek statue. These have been registered completely automatically. The left column of figure shows the 6 patches, already arranged in such way that they can be matched pairwise. The patches were given to the system in random order however. The right column shows pairwise matches as found by the system. The invariant signature used for the bitangent curves simply was distance between bitangent points as a function of arclength. Arclength was measured for the longest of the two curves. This was sufficient to merge all 6. In the figure the left column shows the original, high resolution patches. The right column shows registered, smoothed patches and their dominant

bitangent curves. Fig. 16.8 (a) shows all 6 smoothed patches in a single frame and their dominant bitangent curves. Fig. 16.8 (b) shows the result of applying the found transformations to the original patches. After fine registration (based on ICP), the result of fig. 16.8(c) and (d) is obtained.

Discussion

16.3.3 Invariant-based hashing

16.3.4 Approach

hashing

16.3.5 Examples

16.3.6 Discussion

Model-based methods on the other hand depend on the presence of certain features and, in the case of invariant based descriptions, only limited classes of shapes can be dealt with.

16.4 II - Appearance-based methods

16.4.1 Template matching

Approach

This method is only used with simple changes in object pose, e.g. pure translations of the shape in the image. Template matching consists of comparing, usually by correlation or subtraction techniques, sections of the image with images of the objects one is looking for. The image(s) of the object(s) are called template(s). Hence, the 'model' of an object simply is its image.

In order to find the pattern corresponding to the object, the template is shifted over the image until both overlap precisely. In its general form, this approach is very time consuming, especially if also the orientation (2D) or scale of the pattern are unknown. Then the process has to be repeated for several orientations or scales. Processing can be accelerated by first only comparing a subset of sparsely sampled points in the pattern mask and the image. Further processing is only envisaged when the match with the subtemplate exceeds some threshold.

Examples

Discussion

The sensitivity of template matching to minor mismatches increases with perimeter/area, which can be regarded as a measure of complexity. As a consequence, complex patterns

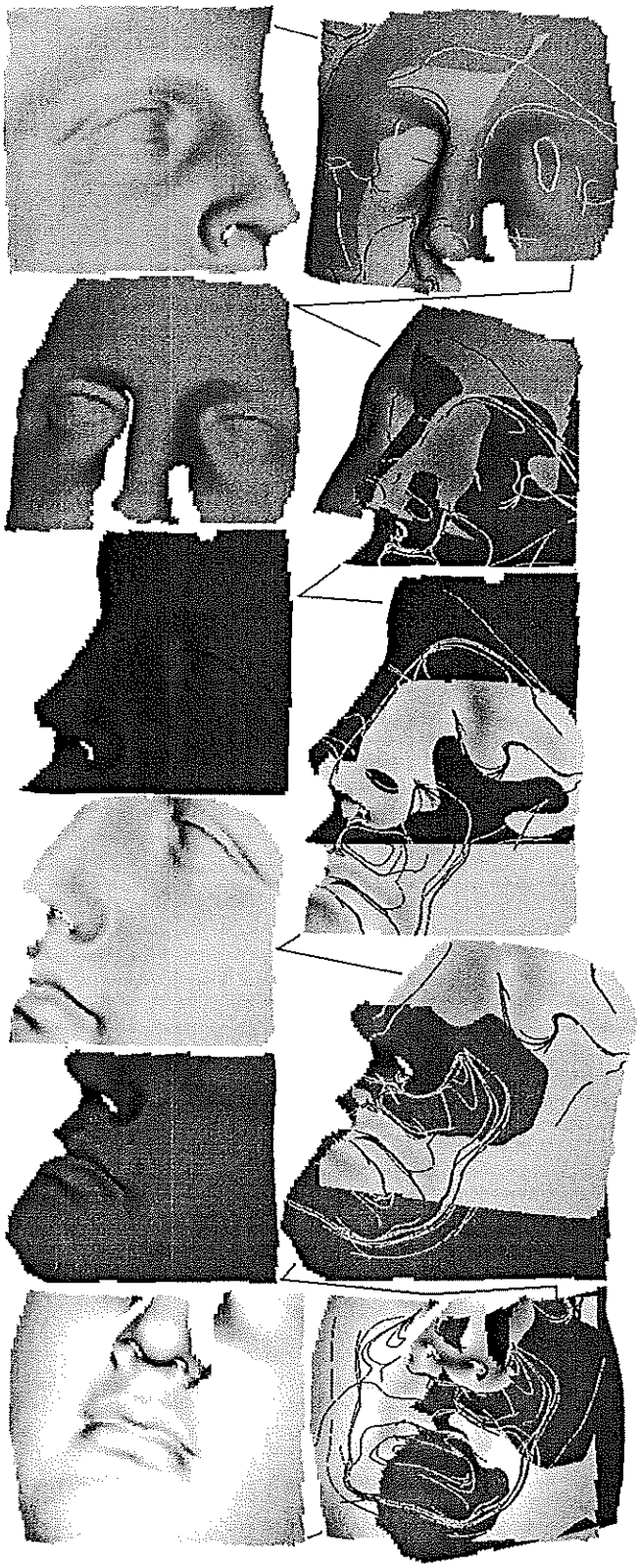
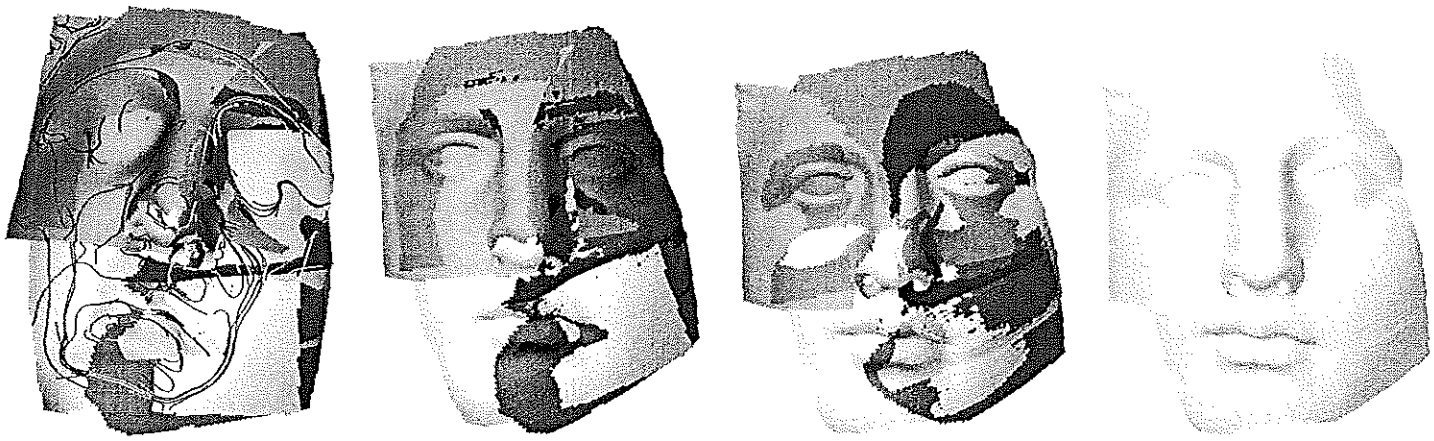


Figure 16.7: *Left column: available patches. Right column: registered 2 by 2.*



(a) (b) (c) (d)

Figure 16.8: *The patches of fig. 16.7 after registration. Different patches have different shades of grey, corresponding to the shades in fig. 16.7. (a) The smoothed patches with the bitangent curve pairs. (b) The result of applying the found transformations to the original patches to obtain a crude registration, (c) The final result after a traditional fine registration, starting from the crude registration of (b). (d) The final result again, but now with all patches in the same colour*

which will often require more computations with other methods, simplify the determination of their precise position, orientation, and/or scale with template matching.

16.4.2 The generalised Hough transform

16.4.3 Correlation with eigenimages

Approach

Rather than building explicit models, one can take images of an object or scene and use the images themselves as model. Recognition then becomes a matter of comparing images. During a training phase, a sufficiently wide variety of model images is taken, to make sure that images presented for recogntion will be similar to at least one of the model images. Recognition should normally work if images are taken from many viewpoints and under several lighting conditions. There are six degrees of freedom to account for object pose and three degrees of freedom to deal with changes in the position of a point light source (linear combinations of which yield arbitrary light sources). As described later, image intensities will be normalised to deal with changes in lighsource radiance. Suppose 128×128 images are used. Then each image represents a point in a 16384-dimensional space. When continuously changing the pose and illumination parameters it can be expected that the image and hence this point will generate a 9-dimensional manifold embedded in the 16384-dimensional space. Of course, in practice one can only sample this manifold, i.e. take images for a finite number of pose and illumination parameters. The manifold needs to be

created through interpolation.

If different objects generate different images, then the manifolds corresponding to each of the objects will not interset. Given an image of an object that has to be recognized (and for which it is assumed a manifold has already been created(, the questions is to which manifold its point is closest in the 16384-dimensional space. Working in such high-dimensional space is computationally demanding. PCA is used to lower the dimensionality of a problem. PCA captures the variations in the image data by projecting the image onto a lower-dimensional subspace of eigenvectors extracted from the covariance matrix C of the image intensities:

$$C_{ij} = \sum_k (I_i(k) - \mu_i) \ \ (I_j(k) - \mu_j)$$

where I is a vector that contains all image intensities in lexicographic order and where the sum is taken over the different images k. Typically, 10 to 20 eigenvectors suffice because images show strong correlations between the pixels. Note that these eigenvectors can be interpreted as images with which the given image is correlated, hence the name 'eigenimages' that is sometimes used.

The 'SLAM' (Software Library for Appearance Matching) algorithm consists of the following steps

1. *Image normalization* A bounding box is constructed around the object. A standard image resolution (e.g. 128×128) to represent the data in the box. The average intensity and its variance are normalized to zero and unity, resp.

2. *Building the database* Each normalized image is added to the image database, with a label identifying the object.

3. *Creating the subspace* A PCA is applied to the database images and a subspace is created based on the 10 eigenvectors with the largest eigenvalues.

4. *Object manifold generation* For each object the sample images are combined into a manifold, through spline fitting. In the original SLAM algorithm objects only rotated about an axis under fixed illumination, hence the manifolds are curves (i.e. one-dimensional manifolds).

5. *Object manifold sampling* The fitted manifold is then sampled to provide sufficient coverage for all envisaged viewing conditions. If, for instance, images where only taken every 5° of rotation, the manifold can be sampled every 1°.

6. *Object recognition* Given an image of an object to be recognized, the object first is delineated with a bounding box, sampled at the fixed resolution, normalized and then projected onto the subspace of the eigenvectors. The nearest manifold sample point yields the object identity.



Figure 16.9: *Objects used for the recognition tests with SLAM.*

Examples

Fig. 16.11 shows 100 objects that have been used in tests with the SLAM system. Given the specific restrictions of SLAM, recognition was actually performed on images with isolated objects, on an even background. Such images are shown in fig. 16.10. Fig. 16.11 shows SLAM's graphical user interface (GUI). At the bottom left window one sees an example of a manifold representing the object in the top right window.

faces

Discussion

The appearance-based approach has several advantages:

It can deal with a wide variety of shapes. Recognition has been demonstrated for objects that would be very difficult to handle with model-based techniques.

The appearance-based method deals with several types of variations in the same way. Not only different viewpoints are taken into account, the same approach can also deal e.g. with changing illumination.

Training the system only requires taking images. There is no need for the development of a recognition strategy or interactive interventions to that effect.

There are also important disadvantages, however.

Building the image database is a rather painstaking process. The more degrees of freedom there are, the larger becomes the number of images that need to be taken. The



Figure 16.10: *Images used in the recognition tests with SLAM.*

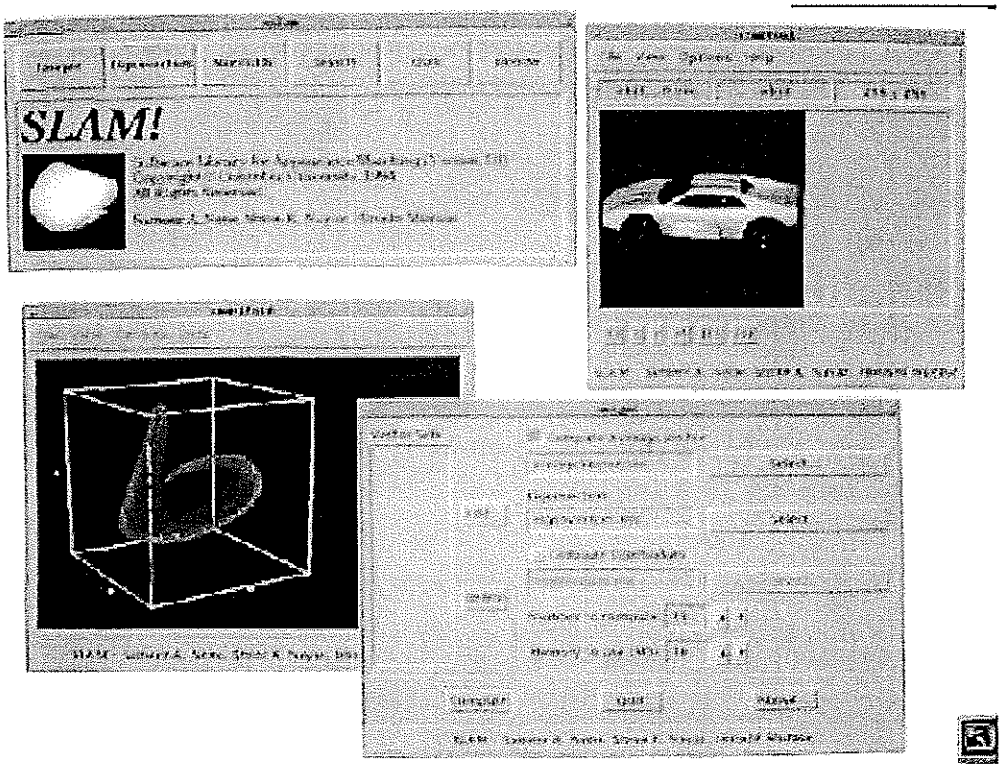


Figure 16.11: *SLAM's graphical user interface.*

method does not extrapolate well, in the sense that recognition needs a model image that is close to it. Model images should cover the complete manifold quite completely.

The object needs to be separated from the background. This is not an easy task to automate as segmentation is not robust unless objects are e.g. lying on a homogeneous background. Using a simple bounding box is not necessarily enough if backgrounds change a lot.

16.5 II - Hybrid (view-based) techniques

This section describes techniques that combine appearance and model based strategies. They are sometimes called *view-based* techniques. Typically, this class of methods models objects on the basis of several views, but rather than taking the images directly as the components of the model, features are extracted from them. The order of the presentation in this section also corresponds with the chronological order by which these methods were developed.

16.5.1 Aspect graphs

Approach

The study of aspect graphs has a long tradition in computer vision. Aspect graphs are built for the recognition of 3D solids. As with appearance-based methods the idea is store several views of an object, rather than a 3D model. In principle, the number of possible viewpoints is infinite. Fortunately, it is sufficient to store some *aspects*, i.e. views for which the image topology of visible edges only continuously changes with changing viewpoint. For polyhedra, this means that no vertices or edges appear or disappear. The aspects are usually referred to as *characteristic views*.

The method goes as follows. Looking at an object from a certain viewpoint, one makes a line drawing of the scene (viz. the computer does an edge extraction). From different viewpoints the line drawings will also differ. However, for certain, nearby viewpoints they will be topologically equivalent. Figure 16.12 shows a sequence of views of a cube. Viewpoints that yield two equivalent views can then be taken together in equivalence classes. In the case of orthographic projection, in which case the distance to the object is irrelevant, all viewpoints can be represented by points on a sphere with the viewing direction as the inward normal. Figure 16.13 shows this view sphere representation. The sphere can then be tesselated in patches or *cells* containing equivalent views, i.e. representing an aspect. In the case of perspective projection, the sphere cannot be used. Distance then also is an important parameter and the complete three-dimensional space of viewpoint positions has to be considered. The cells become volumes.

Usually the information is stored as an aspect graph. The aspects are the nodes and the arcs between them indicate the crossing of a cell boundary between two neighbouring cells. Crossing such a boundary, it is said that *an event* (or *singularity*) occurs, i.e. the topology

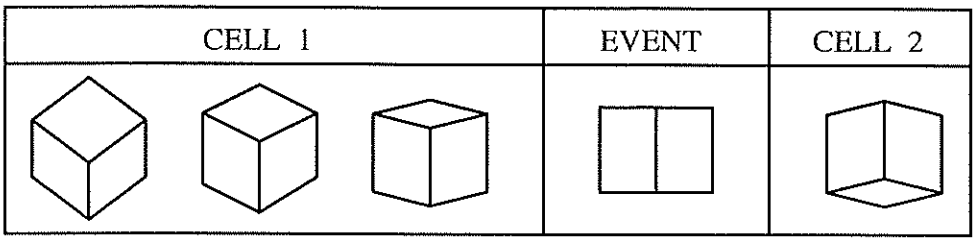


Figure 16.12: *Five views of a cube as seen by a moving observer. From left to right: three subsequent views belonging to the same aspect, a view taken at the boundary of the first aspect, i.e. a non-generic view corresponding to an event or singularity, and a view from the newly entered aspect.*

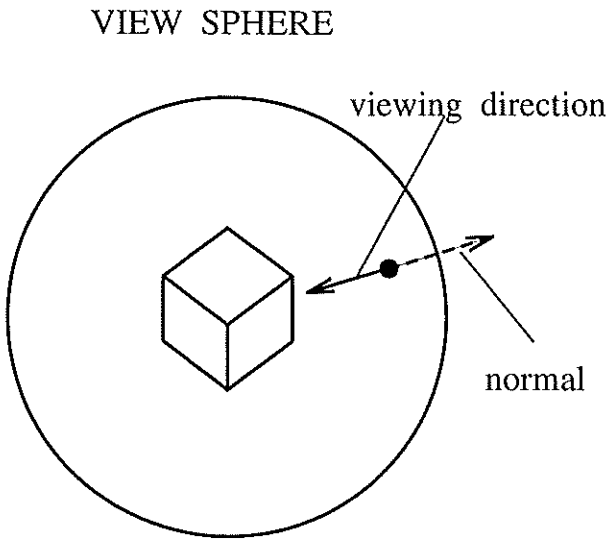


Figure 16.13: *Each point on the view sphere represents on orthographic viewing direction.*

of the line drawing changes. Views taken within the cells are referred to as *generic*, whereas views at the boundaries are called *accidental* or *non-generic*. Having such an aspect graph, the recognition of an object normally is carried out in a number of steps:

1. identify the nodes based on consistency with the viewed topology and possibly more quantitative information as well (e.g. invariants measurable from a planar patch visible in that view),

2. if there are several candidate models, it might be necessary to move in order to check whether events are consistent also, or alternatively to invoke additional metrical measurements (e.g. knowing that lengths or angles should fall within certain bounds),

3. refine the estimate of the viewpoint position, usually again by measuring certain parameters, and by iterative comparison of the image and the calculated projection of the model onto the image plane.

In fact aspect graphs reduce the problem of recognizing a truly 3-dimensional object to one of 2D recognition with all the aspects initially playing the role of a different 2D-model.

Several caveats can be raised against this approach. First of all, as we know edge detection is often the weak link in the chain of image processing steps. It is clear that we heavily depend on its output in order to extract the line drawings and hence the topological structure. Moreover, emphasis will usually be on edge junctions and edge singularities rather than edges themselves, and these are especially difficult to extract. Secondly, building the models (aspect graphs) is a difficult job. This model building step should be automated, which is not straightforward. The number of nodes can grow excessive. If one considers an algebraic surface of degree d, the size of the aspect graph can be $O(d^{12})$. It goes without saying that the calculation of exact aspect graphs from 3D models is far from trivial. Basically, only the extraction of exact aspect graphs for polyhedra and surfaces of revolution is manageable. The first class is quite easy to handle because events correspond to the viewpoint crossing the planes of the object's faces. The second class is easier because the cells on the view sphere are all circular rings around the object's symmetry axis.

Examples

Discussion

The problem of generating aspect graphs is of high complexity. Moreover, exactly the kind of events that are crucial in the definition of topologically different views are very hard to extract from images. Aspect graphs mainly are of academic interest only.

16.5.2 Interest points characterised by Euclidean invariants

Approach

Comparing the model-based and image-based techniques, the first have the advantage that they can deal with cluttered scenes because they operate on features that are extracted from the image. The image-based technques, on the other hand, work globally on (a part of) the image containing the object. These methods cannot cope with scene clutter, unless one would manually segment the object from its background and only a small part of the object is occluded. On the other hand, by focusing on raw intensities instead of features, these methods are not dependent on the presence of any predefined feature and, as a result, can deal with broad classes of objects.

In what can be considered an attempt to get the best from both worlds, a number of hybrids techniques have been proposed. They also work with sets of images to represent the object, just like the image-based approaches. In these images the object is still assumed to ly on a homogeneous background. But the sequel differs in that the model images are analysed further to extract features, both during modeling and later during recognition. These are based on relatively small neighbourhoods of special points. In the work of Schmid and Mohr these are corner points, detected with the Harris corner detector and around these points circular neighbourhoods are considered. This locality is a first important advantage over purely appearance-based strategies, as it becomes possible to deal with overlap and scene clutter at the recognition stage. As long as a sufficient number of these features are found in the image of the object to be recognized, recognition can be successful. It is not necessary to first segment the object from its background during recognition. Moreover, the intensity patterns within the neighbourhoods are not used in their raw form either. From these data features are extracted that remain invariant under rotation in the image plane. This is a second, important advantage over the pur appearance-based approach. these features can now be used irrespective of a rotation about an exis parallel to the optical axis of the camera. Rather than having to take model images for all such rotations, a single image will suffice.

Suppose G is a Gaussian convolution filter and I represents image intensity. Then, $G_x = \frac{\partial}{\partial x}(G * I) = G * \frac{\partial I}{\partial x}$ and $G_y = \frac{\partial}{\partial y}(G * I) = G * \frac{\partial I}{\partial y}$ represent smoothed values for the first derivatives of the intensity pattern, often referred to as the first 'Gaussian derivatives'. Similarly, higher order 'Gaussian derivatives' can be calculated, which will be denoted with similar notations. The profile of the Gaussian drops of to nearly zero, thereby effectively representing a circular neighbourhood. Schmid and Mohr used the following combinations of Gaussian derivatives that are all invariant under image rotation:

$$
\begin{gathered}
G \\
G_x^2 + G_y^2 \\
G_{xx}G_x^2 + 2G_{xy}G_xG_y + G_{yy}G_y^2 \\
G_{xx} + G_{yy} \\
G_{xx}^2 + 2G_{xy}G_{yx} + G_{yy}^2
\end{gathered}
$$

In order to also deal with objects appearing at different scales in the images, these

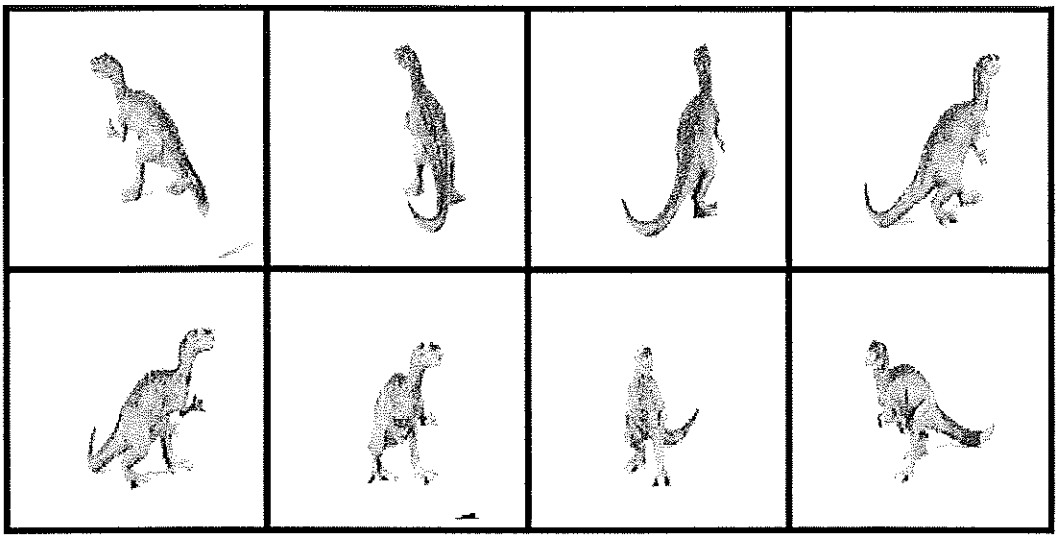


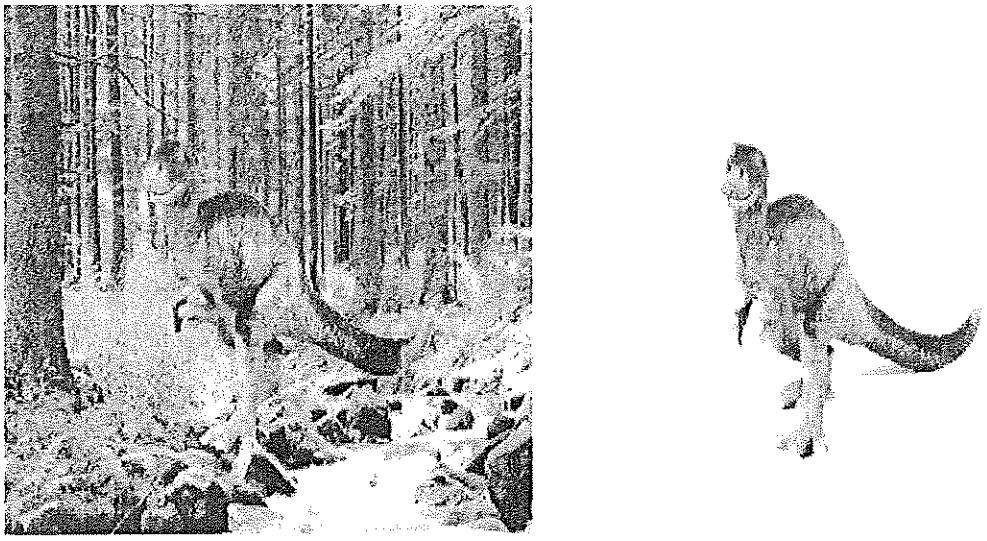
Figure 16.14: *Some of the images used for the creation of a model of a dinosaur shape.*

rotation invariants are extracted for Gaussians of different sizes. For each size the above invariants form a feature vector. If one want to deal with changes in viewing direction as well, then multiple images need to be taken from different directions, as in the case of the appearance-based methods. In later work Schmid and Mohr extended the level of invariance further to photometric changes. This will not be described here, but is covered by the method of the next section.

Feature vectors extracted for an image are compared against the feature vectors for the model images, based on a Mahalanobis distance. Hashing quickly yields good candidates for this matching. In order to further reduce the chances of accidental matches between the image and a model image, the spatial configuration of the matches is also taken into account. It is demanded that if the feature vectors of corners in the given image and a model image match, then also at least half of the points in the neighbourhood around the corners in both images should match and that the angles between the joins connecting them are similar.

Examples

First, a result obtained with the method of Schmid and Mohr is shown. Fig. 16.14 shows a series of images that were used for the modeling of a dinosaur shape. Then, the image in fig. 16.15(a) is presented to the system, after which it produces image 16.15(b) as the most similar image in its database. Not only has the object been recognized correctly, in spite of the cluttered background, the system also produces the correct pose.



(a) (b)

Figure 16.15: *(a) Input image. (b) Image that has obtained most votes.*

Discussion

This hybrid method combines the advantages of the model-based and the appearance-based approaches. Due to the use of local invariants, it can deal with occlusions, scene clutter, and changing backgrounds. These are important advantages inherited from the model-based approaches. The invariance of the features also reduces the number of model images that need to be taken.

On the other hand, the extraction of models does not require more than taking images from the point of view of the user, as all further feature extraction is automatic. The complexity and diversity of the objects that can be dealt with is also high, just as with appearance-based methods pure.

16.5.3 Interest points characterised by affine invariants

Approach

The approach of the previous section was able to reduce the number of reference views necessary to recognize objects and this mainly due to its exploitation of invariance. The use of a single model image for different viewpoints is still quite limited however, as invariance is restricted to rotation about the optical axis and translation. This section describes a method that takes the exploitation of invariance one step further and allows the same model view to be used from more general viewpoint changes. Euclidean invariance is replaced by affine invariance. Local image patches are considered, as was the case with Schmid and Mohr's approach. These patches are considered to be approximately planar and sufficiently small so that affine deformation is a good model for the way in which such patches transform when viewed from another direction. Using affine features then allows the same reference

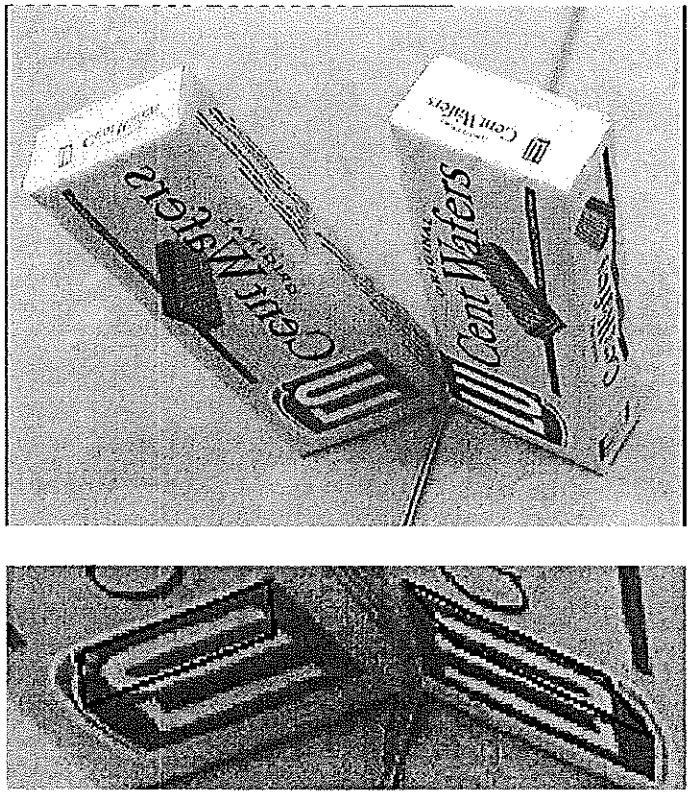


Figure 16.16: *(a) Image of a package and its mirror image, yielding a second view from another relative direction. (b) Two parallelogram-shaped patches as they should be generated by the system: when the viewpoint changes the shapes of the patches are transformed automatically so that they always cover the same physical part of the scene.*

view to be used for a wider set of viewpoints. Moreover, invariance can be extended to photometric changes as well, in order to deal with varying illumination conditions. The selection of image patches and the choice of geometric/photometric invariant features to describe them is now described in more detail.

The crux of the matter is the selection of the patches. Simple circular shapes can no longer be used, as a patch that is viewed as a circular patch when viewed head-on has to be deformed to an elliptical shape when viewed obliquely. Hence, elliptical shapes would be a better choice. Parallelograms are a simpler alternative, as affine deformations map parallelograms to parallelograms. These are the shapes considered here. Fig. 16.16(a) shows a package and its mirror image. Hence, in the image we have to copies of the same shape, but from different viewpoints. The problem now is to come up with parallelogram-shaped patches that change shape with changing viewpoint as to ensure that they cover the same, physical part of the scene. Fig. 16.16(b) shows two such patches, that do indeed represent the same part of the package. Such patches should be selected in a given image, without any a prior knowledge about the corresponding patches in the model image(s). Putting the same remark differently, the problem is not finding physically corresponding

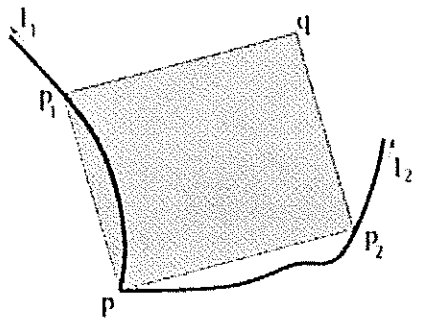


Figure 16.17: *For corners on curved edges, affinely invariant, parallelogram-shapede patches can be constructed (see text).*

patches given a pair of images, but coming up with patches in an image and then, given another view of the same scene, coming up with physically corresponding patches solely based on this second image.

To reduce the complexity of the problem, we restrict ourselves to finding affinely invariant regions for corner points making use of nearby edges. Also, a distinction is made between two different cases: curved edges and straight edges.

Case 1: Curved Edges

Let $\mathbf{p} = (x_p, y_p)$ be a corner point, lying on an edge. Fig. 16.17 shows the corner and the edge on which it lies. If we now select a pair of points $\mathbf{p_1}$ and $\mathbf{p_2}$, on the edge, with one on either side of the corner $\mathbf{p}$, then these three points define a parallelogram. This is also shown in fig. 16.17. We should now select parallelograms in such a way that corresponding parallelograms are selected when first the forementioned geometric/photometric transformations are applied to the image. This means that the criteria that we use in making this selection should be invariant under these transformations. In this particular case, selecting a parallelogram amounts to selecting a pair of points ($\mathbf{p_1}$,$\mathbf{p_2}$).

In a first step, the number of possibilities is reduced by linking the choice of $\mathbf{p_1}$ to that of $\mathbf{p_2}$. To that end, the positions along the edge of the points $\mathbf{p_1}$ and $\mathbf{p_2}$ are specified using two parameters l_1 and l_2:

$$l_i = \int abs(\ |\mathbf{p_i}^{(1)}(s_i)\ \ \mathbf{p} - \mathbf{p_i}(s_i)|\ )\ ds_i \qquad i = 1, 2$$

with s_i an arbitrary curve parameter, $\mathbf{p}^{(1)}(s_i)$ the first derivative of $\mathbf{p_i}(s_i)$ with respect to s_i, $abs()$ the absolute value and $|..|$ the determinant. Then, a point $\mathbf{p_1}(l_1)$ on one side of the edge can be associated with a point $\mathbf{p_2}(l_2)$ on the other side of the edge by having $l_1 = l_2$. Both l_1 and l_2 are relative, affine invariants, but their ratio $\frac{l_1}{l_2}$ is an absolute affine invariant and therefore the association between the points following this criteria is also affinely invariant. If the image is affinely distorted, the corresponding, same point pairs ($\mathbf{p_1}$,$\mathbf{p_2}$) will have $l_1 = l_2$. From now on, we will simply use l when referring to $l_1 = l_2$.

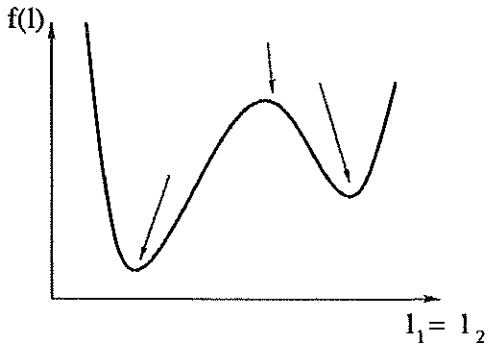


Figure 16.18: *For the selection of the point* $\mathbf{p_1}$ *a feature is evaluated for the intensity pattern within its parallelogram. Positions of* $\mathbf{p_1}$ *are fixed where this features reaches its extremal values.*

The selection of two points $\mathbf{p_1}$ and $\mathbf{p_2}$ is now reduced to the selection of a single point $\mathbf{p_1}$, as this choice implies that of $\mathbf{p_2}$. Nevertheless, there still remains an infinite number of possibilities for $\mathbf{p_1}$ (in the continuous case). This should be reduced to a finite number of selected positions for $\mathbf{p_1}$. To this end, a feature value is extracted based on the intensity or colour pattern $\mathbf{wp_1}$ within the parallelograms that come with the different choices of $\mathbf{p_1}$. Positions of $\mathbf{p_1}$ are selected for which this feature reaches its extremal values. It goes without saying that the extremal values of this feature should again be invariant under both the geometric and photometric changes that are considered. This selection process is illustrated in fig. 16.18. Such feature is not difficult to find. The average value of one of the colour bands is an example. Since it is not guaranteed that such feature actually reaches an extremum over the limited l-interval that is considered, a set of features are tested.

Case 2: Straight Edges

In the case of straight edges, the method described above cannot be applied, since the relative affinely invariant parameter l will be zero. As straight edges are prominent features for man-made objects it is worthwhile to look for an alternative.

A straightforward extension of the previous technique would then be to search for local extrema of some appropriate function in a two-dimensional search-space for independent choices of $\mathbf{p_1}$ and $\mathbf{p_2}$. Unfortunately, the extrema of the simple features that were proposed earlier tend to be not very outspoken in the 2-dimensional case. The feature values often show valleys and ridges with a small slope. By considering two features and by looking at the intersections of such valleys and ridges, stable selections for the positions of $\mathbf{p_1}$ and $\mathbf{p_2}$ can be made. This is illustrated in fig. 16.19. Of course, it is not guaranteed that the valleys and ridges will indeed intersect. That is why again different combinations of features are tested. Moreover, the special case where two valleys or ridges (almost) coincide must be detected and rejected, since the intersection will not be accurate in that case.

Characterising the patches Now that we are able to extract affinely invariant regions in an image, local, affine moment invariants can be computed and used to match a query

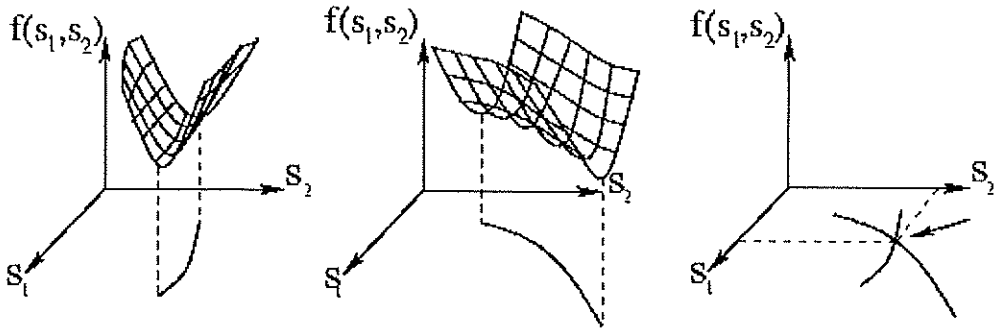


Figure 16.19: *For straight edges the intersection of the "valleys" and "ridges" of two different features are used.*

image with images in the database representing the same scene or object. First, the moment invariants are described. Next, the object recognition process is discussed in more detail.

Voor het vervolg: meer specifieke lijst van de momentinvarianten

As in the region finding step, we consider invariance both under affine geometric changes and linear photometric changes, with different offsets and scalefactors for each of the three color bands.

For each region, a feature vector is composed, consisting of affine moment invariants. These can be matched quite efficiently with the invariant vectors computed for the regions in the database images, using a hashing-technique. In this way, combinatorics can be avoided, reducing the computation time from $O(nN)$ to $O(n)$ (with n the number of regions in the query image and N the total number of regions in all of the database images). As a result, the processing time is independent of the size of the database.

As a first invariant– and, as shown by our experiments, a quite distinctive one – we use the "type" of the region found. Here, with "type" we refer to the method used to find the region: was it found using the method for curved edges or with the method for straight edges ? and what functions were used ? Only if the type of two regions corresponds, can they be matched.

The other invariants are all affine moment invariants. Care has to be taken, however, that they are sufficiently distinctive. More specifically, one must pay attention to the fact that the region finding process turns some invariants into trivial cases.

Due to space limits, a detailed description of the invariants used in our experiments can not be given here. Briefly, moment invariants up to the second order are computed, that use each of the three color bands to obtain a higher distinctive power.

Examples

As an example a coke can is recognized from a scene. Fig. 16.20 shows the views that were taken to build the model. Note the small number of views that is used. There are only 7 views taken from the different sides on one from the top. The side views are on the average

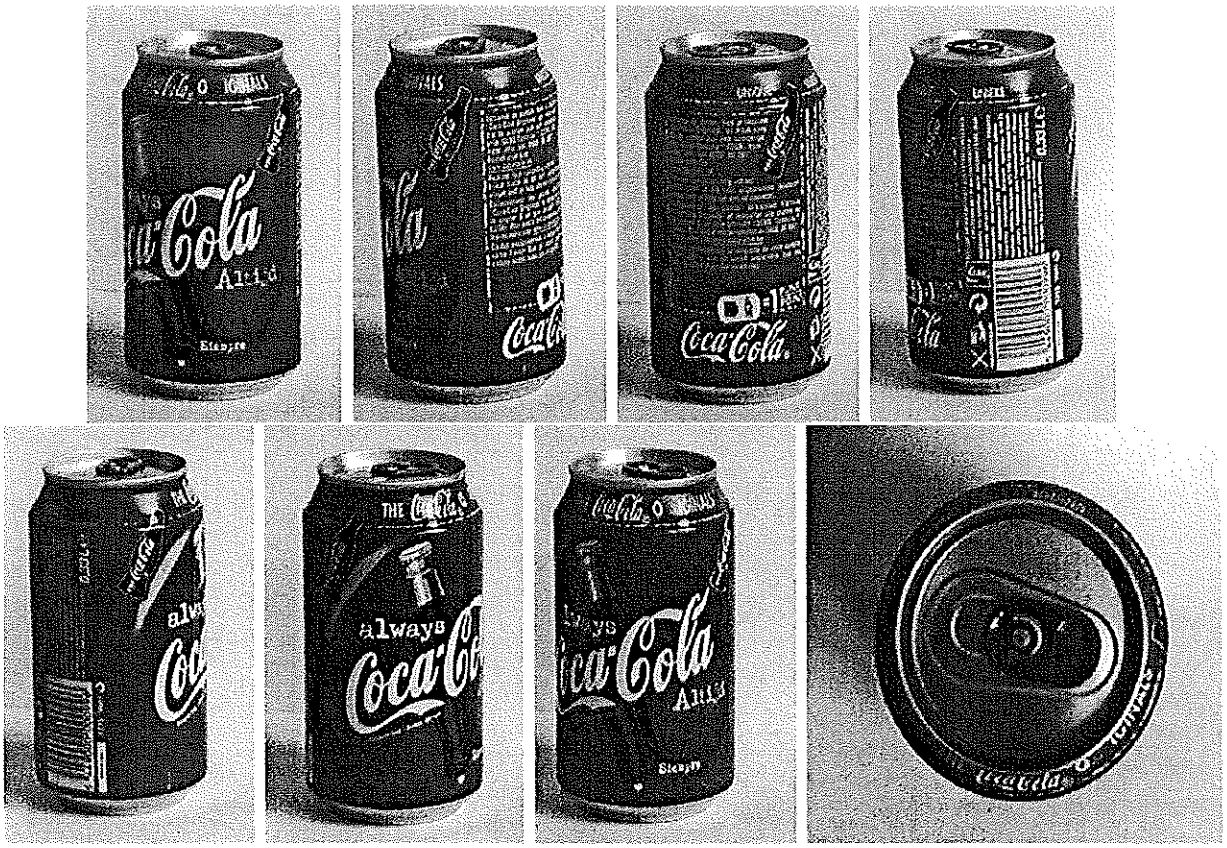


Figure 16.20: *Model views for a coke can.*


COMPUTER
VISION
AND IMAGE
UNDERSTANDING
Coke
Coke
Coke

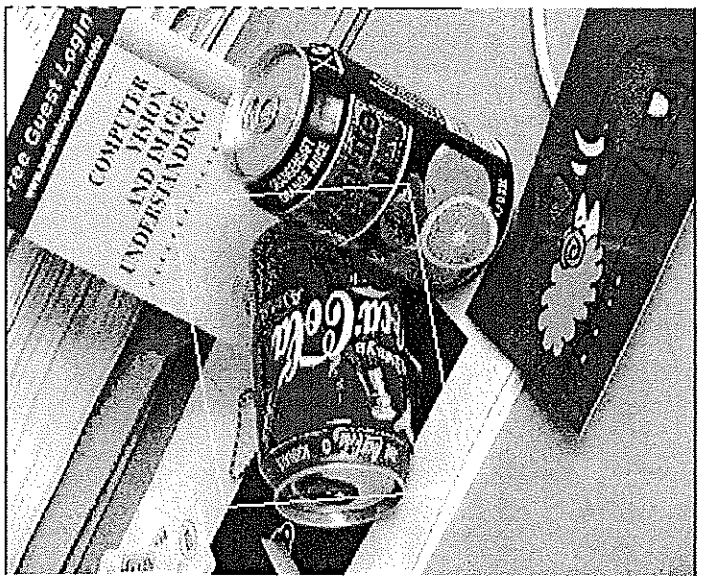


Figure 16.22: *Region where the coke can has been identified.*