Chapter 5
Three-Dimensional Object Recognition

Abstract Some applications require a position estimate in 3D space (and not just in the 2D image plane), e.g., bin picking applications, where individual objects have to be gripped by a robot from an unordered set of objects. Typically, such applications utilize sensor systems which allow for the generation of 3D data and perform matching in 3D space. Another way to determine the 3D pose of an object is to estimate the projection of the object location in 3D space onto a 2D camera image. There exist methods managing to get by with just a single 2D camera image for the estimation of this 3D $\rightarrow$ 2D mapping transformation. Some of them shall be presented in this chapter. They are also examples of correspondence-based schemes, as the matching step is performed by establishing correspondences between scene image and model features. However, instead of using just single scene image and model features, correspondences between special configurations of multiple features are established here. First of all, the SCERPO system makes use of feature groupings which are perceived similar from a wide variety of viewpoints. Another method, called relational indexing, uses hash tables to speed up the search. Finally, a system called LEWIS derives so-called invariants from specific feature configurations, which are designed such that their topologies remain stable for differing viewpoints.

5.1 Overview

Before presenting the methods, let's define what is meant by "3D object recognition" here. The methods presented up to now perform matching of a 2D model to the 2D camera image plane, i.e., the estimated transformation between model and scene image describes a mapping from 2D to 2D. Of course, this is a simplification of reality where the objects to be recognized are located in a 3D coordinate system (often called *world coordinates*) and are projected onto a 2D image plane. Some of the methods intend to achieve invariance with respect to out-of-plane rotations in 3D space, e.g., by assuming that the objects to be found are nearly planar. In that case, a change of the object pose can be modeled by a 2D affine transformation. However, the mapping still is from 2D to 2D.

In contrast to that, 3D matching describes the mapping of 3D positions to 3D positions again. In order to obtain a 3D representation of a scene, well-known methods such as triangulation or binocular stereo can be applied. Please note that many of the methods utilize so-called range images or depth maps, where information about the z-direction (e.g., z-distance to the sensor) is stored dependent on the $[x, y]$-position in the image plane. Such a data representation is not "full" 3D yet and therefore is often called $2\frac{1}{2}$ D.

Another way to determine the 3D pose of an object is to estimate the projection of the object location in 3D space onto the 2D camera image, i.e., estimate the parameters of a projective transformation mapping 3D to 2D, and that's exactly what the methods presented below are doing. The information provided by the projection can for example be utilized in bin picking applications, where individual objects have to be gripped by a robot from an unordered set of objects.

The projective transformation which maps a $[X, Y, Z]^T$ position of world coordinates onto the $[x, y]$-camera image plane is described by

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{R} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t} \tag{5.1}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \cdot a/c \\ f \cdot b/c \end{bmatrix} \tag{5.2}$$

with $\mathbf{R}$ being a 3×3-rotation matrix, $\mathbf{t}$ a 3D translation vector, and f is determined by the camera focal length. Observe that in order be a rotation matrix, constraints are imposed upon $\mathbf{R}$, which makes the problem of finding the projective transformation a non-linear one, at least in the general case. Detailed solutions for the parameter estimation are not presented in this book; our focus should be the mode of operation of the matching step.

Performing 3D object recognition from a single 2D image involves matching 2D features generated from a sensed 2D scene image to a 3D object model. The methods presented here implement the matching step by establishing correspondences between scene image and model features (or, more generally, feature combinations) and are therefore also examples of correspondence-based schemes.

In terms of object representation, a complete 3D model can be derived either from 3D CAD data or from multiple 2D images acquired from different viewpoints. There exist two main strategies for model representation: an object-centered one, where the model consists of a single feature set containing features collected from all viewpoints (or the entire CAD model, respectively) or a view-centered one where nearby viewpoints are summarized to a viewpoint class and a separate feature set is derived for each viewpoint class.

Algorithms that perform 3D object recognition from a single 2D image are mostly applied in industrial applications, as industrial parts usually can be modeled by a restricted set of salient features. Additionally, the possibility to influence the imaging conditions alleviates the difficulty involved by the fact of relying on configurations of multiple features, because usually it is very challenging to detect them reliably. Three methods falling into this category are presented in the following.

5.2 The SCERPO System: Perceptual Grouping

5.2.1 Main Idea

The SCERPO vision system (Spatial Correspondence, Evidential Reasoning and Perceptual Organization) developed by Lowe [3] is inspired by human recognition abilities. It makes use of the concept of *perceptual grouping*, which defines groupings of features (e.g., lines) that are considered salient by us humans and therefore can easily be perceived (cf. Fig. 5.1).

There is evidence that object recognition of the human vision system works in a similar way. As there should be no assumptions about the viewpoint location from which the image was acquired, these feature groupings should be invariant to viewpoint changes, enabling the algorithm to detect them over a wide range of viewpoints. Lowe describes three kinds of groupings that fulfill this criterion:

- *Parallelism*, i.e., lines which are (nearly) parallel
- *End point proximity*, i.e., lines whose end points are very close to each other
- *Collinearity*, i.e., lines whose end points are located on or nearby a single "master-line."

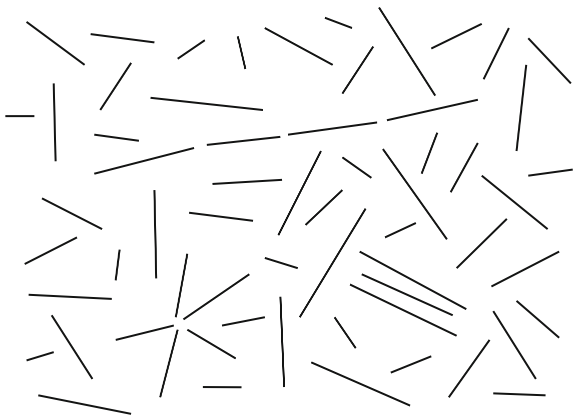


Fig. 5.1 Showing three kinds of perceptually significant line groupings: five *lines* ending at positions which are very close to each other in the *lower left* part, three *parallel lines* in the *lower right*, and, finally, four almost *collinear lines* in the upper part of the picture

5.2.2 Recognition Phase

Based on the concept of perceptual grouping, Lowe proposes the following algorithm for recognition. Lets assume model data is available already, e.g., from CAD data. Figure 5.2 illustrates the approach: as an example, the 3D poses of multiple razors have to be found in a single scene image:

1. *Edge point detection*: at first, all edge points **e** have to be extracted from the image. To this end, Lowe suggests the convolution of the image with a Laplacian of Gaussian (LoG) operator. As this operation relates to the second derivative of image intensity, edge points should lie on zero-crossings of the convolution result. In order to suppress zero crossings produced by noise, pixels at zero crossing positions additionally have to exhibit sufficiently high gradient values in order to be accepted as edge pixels.
2. *Edge point grouping*: derivation of line segments l_i which approximate the edge points best (see also the previous chapter for a brief introduction).
3. *Perceptual grouping* of the found line segments considering all three kinds of grouping. In this step, a group g_n of line segments is built if at least two lines share the same type of common attribute (collinearity, parallelism or proximal end points).
4. *Matching* of the found line groups to model features taking the *viewpoint consistency constraint* into account. The viewpoint consistency constraint states that a correct match is only found if the positions of all lines of one group of the model can be fit to the positions of the scene image lines with a single common projection based on a single viewpoint. In other words, the positions of the lines have to be consistent with respect to the transformation parameters.
5. *Projection hypothesis generation*: each matching of a model line group to some scene image features can be used to derive a hypothesis of the projection parameters between a 3D object pose and the scene image.
6. *Projection hypothesis verification*: based on the hypothesis, the position of other (non-salient) features/lines in the scene image can be predicted and verified. The hypothesis is valid if enough consistent features are found.

As it might be possible to formulate many hypotheses it is desirable to do a ranking of them with respect to the probability of being a correct transformation. Most promising are groups consisting of many lines as they are supposed to be most distinctive and have the additional advantage that most likely all projection parameters can be estimated (because sufficient information is available; no underdetermination). This concept of formulating hypotheses with only a few distinctive features followed by a verification step with the help of additional features can be found quite often as it has the advantage to be insensitive to outliers (in contrast to calculating some kind of "mean").

5.2.3 Example

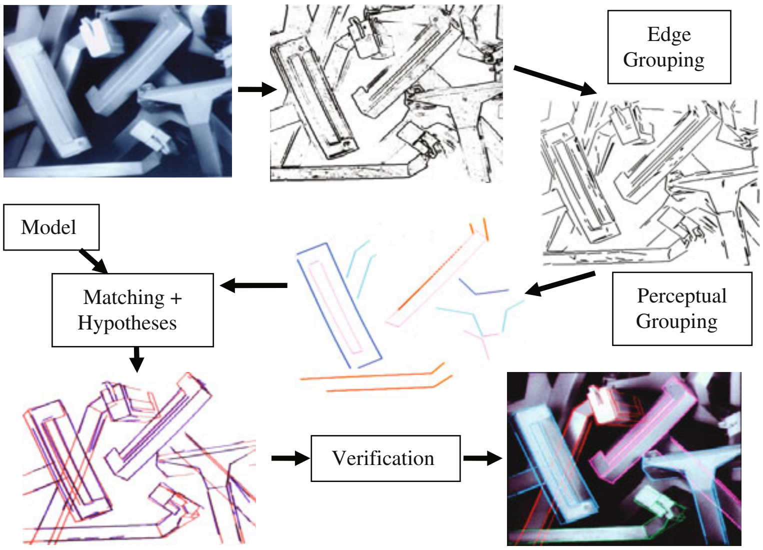


Fig. 5.2 Individual razors are detected by perceptual grouping with the SCERPO system[1]

5.2.4 Pseudocode

```
function detectObjectPosPerceptualGrouping (in Image $I$, in list of model groups $\mathbf{g}_M$, in list of model lines $\mathbf{l}_M$, in distance threshold $t_d$, in similarity threshold $t_{sim}$, out object position list $\mathbf{p}$)

// line segment detection
Convolve $I$ with Laplacian of Gaussian (LoG) operator
$I_G \leftarrow$ gradient magnitude at zero crossings of convol. result
threshold $I_G$ in order to obtain list of edge points $\mathbf{e}$
group edge points $\mathbf{e}$ to line segments $l_i$, if possible
remove very short segments from line segment list $\mathbf{l}_S$

// perceptual grouping
while unprocessed line segments exist in $\mathbf{l}_S$ do
   take next line segment $l_{S,i}$ from list $\mathbf{l}_S$
```

[1]Contains images reprinted from Lowe [3] (Figs. 9, 10, 11, 12, 14, and 16), © 1987, with permission from Elsevier.

```
   init of group $g_n$ with $l_{S,i}$
   **for** all line segments $l_{S,k}$ in the vicinity of $l_{S,i}$
      **if** $\left[endpt\_prox\left(l_{S,k}, g_n\right) \vee collin\left(l_{S,k}, g_n\right) \vee parallel\left(l_{S,k}, g_n\right)\right] \wedge$
      group type fits **then**
         append $l_{S,k}$ to $g_n$
         set type of group $g_n$ if not set already (collinear,
         endpoint prox. or parallel)
      **end if**
   **next**
   **if** number of lines of $g_n$ >= 2 **then**
      accept $g_n$ as perceptual group and add it to list
      of perceptual groups in scene image $\mathbf{g}_S$
      remove all line segments of $g_n$ from list $\mathbf{l}_S$
   **end if**
**end while**

// matching
**for** i = 1 to number of model groups (elements of $\mathbf{g}_M$)
   **for** j = 1 to number of scene groups (elements of $\mathbf{g}_S$)
      **if** viewpoint consistency constraint is fulfilled for
      all lines of $g_{M,i}$ and $g_{S,j}$ **then**
         estimate transform parameters $t$ // hypothesis
         //hypothesis verification
         $sim \leftarrow 0$
         **for** k = 1 to total number of line combinations
            **if** $\left\| t\left(l_{M,k}\right) - l_{S,k} \right\| \leq t_d$ **then**      // positions fit
               increase $sim$
            **end if**
         **next**
        **if** $sim \geq t_{\mathrm{sim}}$ **then**
           append $t$ to position list $\mathbf{p}$
         **end if**
      **end if**
   **next**
**next**
```

5.2.5 *Rating*

An advantage of this procedure is that it is a generic method which doesn't include many specific assumptions about the objects to be detected. Furthermore, one image is enough for 3D recognition.

In order to make the method work, however, it has to be ensured that there exist some perceptual groups with suitable size which are detectable from all over the expected viewpoint range. Compared to the methods presented below only lines are used, which constrains the applicability. Additionally, 3D model data has to be available, e.g., from a CAD model.

5.3 Relational Indexing

5.3.1 Main Idea

The algorithm presented by Costa and Shapiro [2], which is another example of aiming at recognizing 3D objects from a single 2D image, uses a scheme which is called *relational indexing*. In this method object matching is performed by establishing correspondences, too. However, these correspondences are not identified between single features; a pair of so-called high-level features is used instead. The features utilized by Costa and Shapiro are extracted from edge images (which, e.g., can be calculated from the original intensity image with the operator proposed by Canny [1] including non-maximum suppression) and are combinations of primitives such as lines, circular arcs, or ellipses. Therefore the method is suited best for the recognition of industrial parts like screws, wrenches, stacked cylinders. A summary of high-level features can be found in the top tow rows of Table 5.1. Most of them are combinations of two or more primitives.

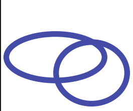
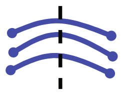
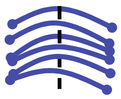
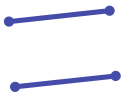
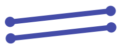
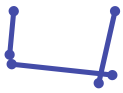
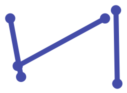
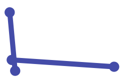
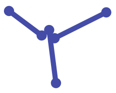
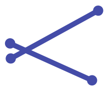
For matching, two of these high-level features are combined to a pair, which can be characterized by a specific geometric relation between the two features, e.g., two features can share one common line segment or circular arc (see also the bottom two rows of Table 5.1).

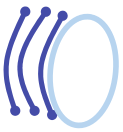
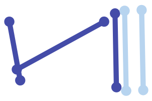
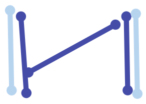
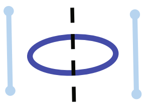
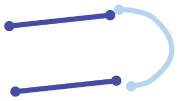
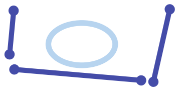
The main advantage of the usage of two features and their geometric relation is that their combination is more salient and therefore produces more reliable matches compared to a single feature. This implies, however, that the object to be recognized contains enough such combinations. Additionally, these combinations have to be detected reliably, which is the more difficult the more complex the combinations are.

Object matching is performed by establishing correspondences between a pair of two high-level model features (and their geometric relation) and pairs found in a scene image. A correspondence is only valid if both pairs are composed by identical feature types and share the same geometric relation. Each of the found correspondences votes for a specific model. By counting these votes hypotheses for object classification as well as pose identification can be derived.

In order to achieve invariance with respect to viewpoint change, a view-based object model is applied. Therefore images taken at different viewpoints are processed for each object in a training phase. Images where the object has a similar appearance are summarized to a so-called *view-class*. For each view-class high-level feature pairs are derived and stored separately, i.e., the model for a specific object class consists of several lists of high-level feature pairs and their geometric relation, one list for each view class.

Table 5.1 High-level feature types (*top two rows*) and types of relations between the features (*bottom two rows*) used by relational indexing

				
Ellipses	Coaxials-3	Coaxials-multi	Parallel-far	Parallel-close
				
U-triple	Z-triple	L-Junction	Y-Junction	V-Junction

		
Share one arc	Share one line	Share two lines
		
Coaxial	Close at extremal points	Bounding box encloses/ enclosed by bounding box

The method is a further example of an indexing scheme; it can be seen as an expansion of geometric hashing, which utilizes single features, to the usage of two features and their relation for generating two indices (one based on the feature types, one based on the geometric relation) when accessing the (now 2D) look-up table. The entries of the look-up table represent view-based models which can be used directly to cast a vote for a specific model – view-class combination.

5.3.2 *Teaching Phase*

The teaching process, which can be repeated for different view classes, consists of the following steps:

1. *Edge image generation*: In the first step all edge pixels have to be identified. A suitable implementation for this step is the Canny edge detector with non-maximum suppression proposed in [1]. In principle one intensity image suffices for the edge image generation. Nevertheless, Costa and Shapiro [2] suggest to combine two intensity images with differently directed illumination in order to exclude edges from the model which are caused by shading.

2. *Mid-level feature extraction*: Line segments l_M, i and circular arcs $c_{M,i}$ are extracted from the edge image. As the main focus here is on the matching method, this step shall not be discussed in detail here (a short introduction to feature detection is given in the previous chapter).
3. *Mid-level feature grouping*: in order to utilize the high-level features described above (denoted by $g_{M,i}$), they have to be extracted first. This is done by grouping the mid-level features detected in the previous step.
4. *High-level feature grouping*: the high-level features just created are combined to pairs (denoted by $pg_{M,i}$) consisting of two high-level features and their geometric relation (see the two bottom rows of Table 5.1). Later on, these groups act as an index into a look-up table for the retrieval of model information.
5. *Look-up table generation*: the last training step consists of generating the look-up table just mentioned (see also Fig. 5.3). The look-up table is a 2D table, where one dimension represents the high-level feature combination (each high-level feature type is labeled with a number; for a specific feature pair, these numbers are concatenated). The other dimension incorporates the geometric relation between the two features; again each relation type is represented by a specific number. After teaching, each element of the look-up table contains a list $h_{M,ab}$ of model/view-class pairs, i.e., numbers of the object model and the class of views from which the feature group was detected in the training phase. If, for example, a pair of parallel lines and a z-triple of lines which share two lines are detected for the model “2” in a viewpoint belonging to view-class “3” during training, the information “2–3” is added to the list of the hash table entry being defined by the indexes of “parallel-far”-“z-triple” and “share two lines.” Please note that, in contrast to geometric hashing or the generalized Hough transform, no quantization of the spatial position of the features is necessary here.

Figure 5.3 shows one step of the 2D hash table generation in more detail. In this example a high-level feature pair (briefly called “feature pair” in the following) consisting of parallel-far lines (e.g., defined as feature number 2) and a z-junction

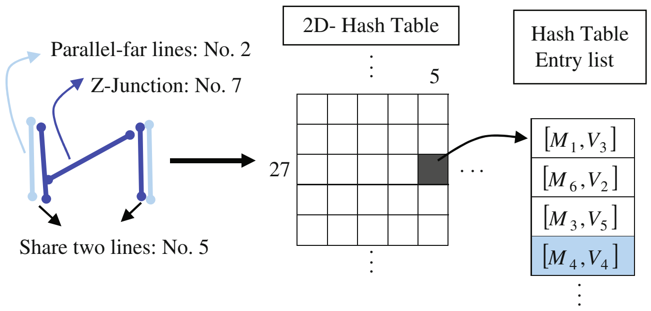


Fig. 5.3 Illustrating one step of the hash table generation

(e.g., defined as feature number 7) has been detected. Their relation is characterized by the sharing of two lines (number 5). The feature combination "27" as well as relation "5" define the two indexes a and b of the entry $h_{M,ab}$ of the 2D hash table which has to be adjusted: The list of this entry, where each list entry defines a model number and view-class combination $\left[M_i, V_j\right]$, has to be extended by on entry (marked blue). Here, the feature combination 27-5 belongs to model number 4 and was detected in view-class number 4.

5.3.3 Recognition Phase

The beginning of the recognition process is very similar to the teaching phase. In fact, step 1–4 are identical. At the end of step 4, all feature pairs $pg_{S,k}$ which could be extracted from the scene image by the system are known. The rest of the method deals with hypothesizing and verifying occurrences of objects in the scene based on the extracted $pg_{S,k}$ and proceeds as follows:

5. *Voting*: For each high-level feature pair $pg_{S,k}$ the two indexes a and b into the look-up table (a is based on the two high-level features of the pair and b is based on their geometric relation) can be derived. The element $\mathbf{h}_{M,ab}$ of the look-up table that can be addressed by the two indexes consists of a list of models which contain a feature pair $pg_{M,i}$ with feature types as well as relation being identical to the ones of $pg_{S,k}$ and therefore support its occurrence in the scene image. Each list entry $h_{M,ab.l}$ casts a vote, i.e., a specific bin (relating to the model index defined in $h_{M,ab.l}$) in an accumulator array consisting of indexes for all models (i.e., the model database) is incremented by one.
6. *Hypothesis generation*: hypotheses for possible model occurrences in the scene image can be generated by searching the accumulator for values above a certain threshold t_R
7. *3D Pose estimation*: based on all feature pairs supporting a specific hypothesis hy an estimation of the 3D object pose can be derived. To this end, the matched 2D positions of the features in the scene image to their corresponding 3D positions of the 3D object model (e.g., a CAD model) are utilized. In order to estimate a 3D pose six such 2D–3D feature matches are required. Details of the estimation scheme, which in general requires nonlinear estimation, but can be linearized in special cases, can be found in [2].
8. ***Verification***: roughly speaking, a hypothesis hy is valid if enough edge points of the back-projected 3D model into the camera image (with the help of the estimated 3D pose of step 7) are located near an edge point detected in the scene image. To this end, the directed Hausdorff distance $h\left(t\left(m\right), I\right) = \min\limits_{i \in I} \|m - i\|$ to the scene image edge point set I is calculated for each back-projected model edge point $m \in M$. In order to consider a hypothesis as valid, two conditions must hold: First of all, the average of the Hausdorff distances for all back-projected model edge points must remain below a threshold t_{dist} and, second, the fraction of model pixels with actual distance below t_{dist} must be above the value t_{fr}:

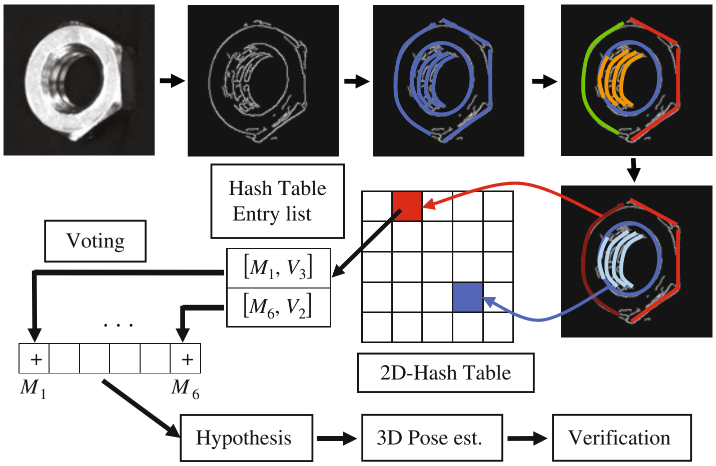


Fig. 5.4 Screw nut detection with relational indexing

$$\frac{1}{M}\sum_{m\in M} h\left(t\left(m\right), I\right) \le t_{\text{dist}} \tag{5.3}$$

$$\frac{N\left(m \in M | h\left(t\left(m\right), I\right) \le t_{\text{dist}}\right)}{M} \ge t_{fr} \tag{5.4}$$

where $N\left(m \in M | h\left(t\left(m\right), I\right) \le t_{\text{dist}}\right)$ denotes the number of model points with a distance at most equal to t_{dist}. In [2] t_{dist} is set empirically to 5 pixels and t_{fr} to 0.65. Please note that t_{fr} controls the amount of occlusion which should be tolerated by the system.

The entire recognition process is illustrated in Fig. 5.4: after edge extraction and edge pixel grouping, several mid-level features (marked blue) are detected. In the next step, they're grouped to so-called high-level features (rightmost image in top row; each high-level feature is indicated by a different color). Subsequently, the high-level features are combined to pairs (e.g., the combination ellipse-coaxials combination marked blue and the u-triple-ellipse part-combination marked red). Note that other combinations are also possible, but are not considered here for better visibility. Each combination can be used as an index into the 2D hash table built during training. The hash table list entries are used during voting. Again, not all list entries are shown because of better visibility.

5.3.4 Pseudocode

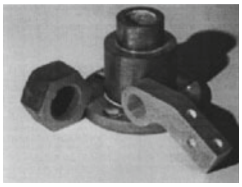
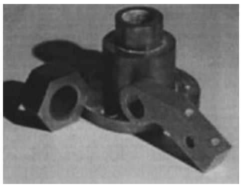
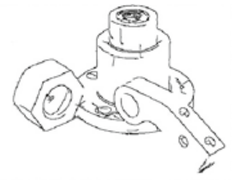
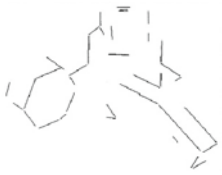
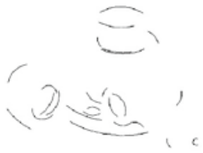
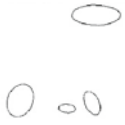
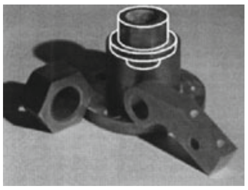
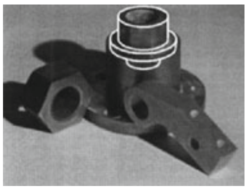
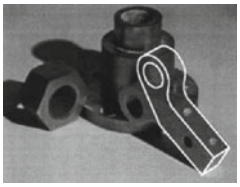
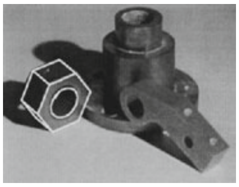
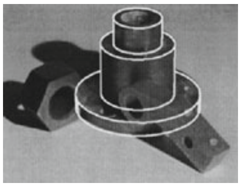
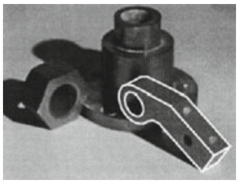
```
function detectObjectPosRelIndex (in Image $I$, in 2D hash table $\mathbf{H}$, in model data $M$ for each object model, in thresholds $t_R$, $t_{dist}$ and $t_{fr}$, out object position list $\mathbf{p}$)
```

```
// detection of high-level feature pairs
detect edge pixels (e.g. Canny) and arrange them in list $\mathbf{e}$
group edge points $\mathbf{e}$ to line segments $l_{S,k}$ and circular arcs
$c_{S,k}$ and add each found mid-level feature to list $\mathbf{l}_S$ or $\mathbf{c}_S$
group mid-level features to high-level features $g_{S,k}$, if
possible , and build list $\mathbf{g}_S$
build pairs of high-level features $pg_{S,k}$, if possible, and
collect them in list $\mathbf{pg}_S$

// voting
Init of 1-dimensional accumulator $accu$
for k = 1 to number of list entries in $\mathbf{pg}_S$
// derive indexes $a$ and $b$ for accessing the 2D hash table:
    $a \leftarrow$ concatenation of the types of the two high-level
    features $g_{S,i}$ and $g_{S,j}$ which build $pg_{S,k}$
    $b \leftarrow$ index of geometric relation between $g_{S,i}$ and $g_{S,j}$
    retrieve model list $\mathbf{h}_{M,ab}$ from $\mathbf{H}(a,b)$
    for l = 1 to number of model entries in $\mathbf{h}_{M,ab}$
        increment $accu$ for model defined by $h_{M,ab,l}$
    next
next

// hypothesis generation
for all local maxima of $accu$ (bin index denoted by $m$)
   if $accu(m) \geq t_R$ then
      match model features to the found $l_{S,k}$ and $c_{S,k}$
      estimate $\mathbf{t}$ based on the involved feature matches
      add hypothesis $hy = [\mathbf{t}, m]$ to list $\mathbf{hy}$
   end if
next

// hypothesis verification
for i = 1 to number of hypotheses $\mathbf{hy}$
   calculate directed hausdorff distances of back-projected
   model edge point set
   if equations 5.3 and 5.4 are fulfilled then
      append $hy_i$ to position list $\mathbf{p}$
   end if
next
```

5.3.5 Example

Object recognition results achieved with this method are summarized in Table 5.2 (with images taken from the original article of Costa and Shapiro [2]), where

Table 5.2 Illustrating the performance of 3D object recognition with relational indexing[2]

		
Intensity image with directed illumination from the *left*	Intensity image with directed illumination from the *right*	Combined edge image. Edges resulting from shading are eliminated
		
Found line features	Found circular arc features	Found ellipse features
		
Incorrect hypothesis	Incorrect hypothesis	Incorrect hypothesis
		
Correct hypothesis	Correct hypothesis	Correct hypothesis

man-made workpieces have to be detected. The first row shows the two input images (with illumination from different directions: left and right) together with the combined edge image extracted from them. The different types of mid-level features derived from the edge image are shown in the second row. The third row contains some incorrect hypotheses generated in step 6 of the recognition phase; however, all of them did not pass the verification step. All three objects of the scene were found with correct pose, as the bottom row reveals.

[2]Contains images reprinted from Costa and Shapiro [2] (Figs. 22, 23, and 24), © 2000, with permission from Elsevier.

5.3.6 Rating

Experiments performed by the authors showed quite impressive results as far as recognition performance is concerned. They reported no false detections in various test images, whereas almost all objects actually being present in the images were detected at correct position at the same time, despite the presence of multiple objects in most of the images, causing considerable amount of occlusion and clutter.

On the other hand, however, the method relies on the objects to have a "suitable" geometry, i.e., at least some of the high-level features defined in Table 5.1 have to be present. Additionally, the feature groups must be detectable in the scene image. Indeed, the major constraint of the method stems form the instability of the detection of the feature groups: sometimes a feature is missing, sometimes a feature is split because of occlusion (e.g., a long line might be detected as two separate line segments), and so on. Bear in mind that the feature group is only detected correctly if all mid-level features are found correctly as well!

5.4 LEWIS: 3D Recognition of Planar Objects

5.4.1 Main Idea

In order to recognize objects with arbitrary 3D pose it is desirable to derive features from the image of an object which remain constant regardless of the relative position and orientation of the object with respect to the image acquisition system. Such so-called *invariants* allow for 3D object recognition from a single 2D image of arbitrary viewpoint, because the viewpoint is allowed to change between teaching and recognition phase. The invariant value remains stable even if the object undergoes a projective transform. One example using invariants is perceptual grouping described by Lowe [3] which was already presented in a previous section. Although it can be shown that such invariants don't exist for arbitrarily shaped 3D objects, invariants can be derived for specific configurations of geometric primitives (e.g., lines or conics) or a set of linearly independent points.

The LEWIS system (*L*ibrary *E*ntry *W*orking through an *I*ndexing *S*equence) developed by Rothwell et al. [4] makes use of two different types of invariants when performing recognition of planar objects (in this context "planar" means that the object is "flat," i.e., can be approximated well by a 2D plane).

Please note that, in contrast to the methods presented above, just a perspective transformation mapping a 2D model to a 2D image (with eight degrees of freedom, see Section 3.1) is estimated here. However, an extension to non-planar objects, where the estimated transformation describes a projection of the 3D object pose onto a 2D image plane, is possible and provided by the same research group [5]. The principle of invariants remains the same for both methods, and that's the reason why the LEWIS method is presented here.

Observe that the usage of invariants imposes restrictions on the objects to be recognized as they have to contain the aforementioned specific geometric configurations (examples will follow) and, additionally, the invariants have to be detected reliably in the scene images. A class of objects which meets these constraints is man-made, planar objects like spanners, lock striker plates, metal brackets (recognition examples of all of them are given in [4]), and so on.

The outline of the LEWIS method is as follows: Characteristic invariants are detected from edge images for each object to be recognized and stored in the model database in an off-line training step. During recognition, invariants of a scene image are derived with identical procedure. In order to speed up matching, indexing is applied: the value of a single invariant derived from the scene image leads to a hypothesis for the presence of an object model which contains an invariant of identical type and with similar value. In other words, if an invariant can be detected in a scene image, its value is derived and serves as an index into a hash table for retrieving a list of object classes that contain an invariant of identical type and with similar value. In a last step, the hypotheses are verified (resulting in acceptance or rejection of a hypothesis) using complete edge data.

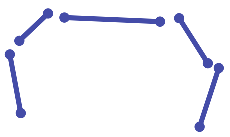
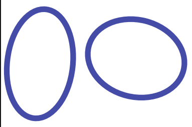
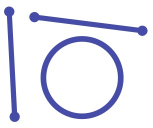
5.4.2 Invariants

For 3D object recognition of planar objects Rothwell et al. [4] make use of two types of invariants: algebraic and canonical frame invariants. As far as algebraic invariants are concerned, they consist of a scalar value which can be derived from a specific geometric configuration of coplanar lines and/or conics. This value remains stable if the underlying feature configuration undergoes a projective transformation. Table 5.3 summarizes the geometric configurations which are utilized by the LEWIS method.

Two functionally independent projective invariants can be derived from five coplanar lines. Given five lines

$$\mathbf{l}_i = [\mu_i \sin\theta_i, -\mu_i \cos\theta_i, \mu_i d_i]^T \,; i \in \{1, \ldots, 5\} \tag{5.5}$$

Table 5.3 Illustration of three different types of geometric configurations of *line* and *circular* features utilized by the LEWIS system in order to derive algebraic invariants

		
Five lines	Two conics	One conic and two lines

(where θ_i denotes the orientation of line i with respect to the x axis, d_i the distance of the line to the origin, and μ_i the scale factor introduced because of the usage of homogeneous coordinates) the invariants I_{L1} and I_{L2} are defined by

$$I_{L1} = \frac{|\mathbf{M}_{431}| \cdot |\mathbf{M}_{521}|}{|\mathbf{M}_{421}| \cdot |\mathbf{M}_{531}|} \tag{5.6}$$

$$I_{L2} = \frac{|\mathbf{M}_{421}| \cdot |\mathbf{M}_{532}|}{|\mathbf{M}_{432}| \cdot |\mathbf{M}_{521}|} \tag{5.7}$$

where the matrices $\mathbf{M}_{ijk}$ are built by a column-wise concatenation of the parameters of three lines $\left[\mathbf{l}_i, \mathbf{l}_j, \mathbf{l}_k\right]$. $\left|\mathbf{M}_{ijk}\right|$ denotes the determinant of $\mathbf{M}_{ijk}$.

Before we define the invariants where conics are used, let's introduce the representation of conics first. A conic is defined by the set of points $\mathbf{x}_i = \left[x_i, y_j, 1\right]$ satisfying $ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i + f = 0$, or equally, the quadratic form

$$\mathbf{x}_i^T \cdot \mathbf{C} \cdot \mathbf{x}_i = 0 \text{ with } \mathbf{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \tag{5.8}$$

In the presence of two conics $\mathbf{C}_1$ and $\mathbf{C}_2$, two independent invariants I_{C1} and I_{C2} can be derived:

$$I_{C1} = \frac{tr\left(\mathbf{C}_1^{-1} \cdot \mathbf{C}_2\right) \cdot |\mathbf{C}_1|^{1/3}}{|\mathbf{C}_2|^{1/3}} \tag{5.9}$$

$$I_{C2} = \frac{tr\left(\mathbf{C}_2^{-1} \cdot \mathbf{C}_1\right) \cdot |\mathbf{C}_2|^{1/3}}{|\mathbf{C}_1|^{1/3}} \tag{5.10}$$

where $tr(\cdot)$ denotes the trace of a matrix.

Two lines and one conic lead to the invariant I_{LC} which is defined by

$$I_{LC} = \frac{\left(\mathbf{l}_1^T \cdot \mathbf{C}^{-1} \cdot \mathbf{l}_2\right)^2}{\left(\mathbf{l}_1^T \cdot \mathbf{C}^{-1} \cdot \mathbf{l}_1\right) \cdot \left(\mathbf{l}_2^T \cdot \mathbf{C}^{-1} \cdot \mathbf{l}_2\right)} \tag{5.11}$$

Canonical frame invariants I_V can be applied to the more general class of planar curves. As a projective transformation is specified by eight parameters, it can be defined by four non-collinear planar points. If four non-collinear points can be uniquely identified on a planar curve, the mapping of these points onto a so-called *canonical frame*, e.g., a unit square, uniquely defines the eight parameters of a projective transformation. In this context "uniquely defines" means that the system always detects the same positions on the curve, regardless of the viewpoint from which the image was taken. To this end, four points around a concavity of the curve are utilized (see Fig. 5.5). Now the entire curve can be transformed into the canonical frame. The transformed curve remains stable regardless of the viewpoint.

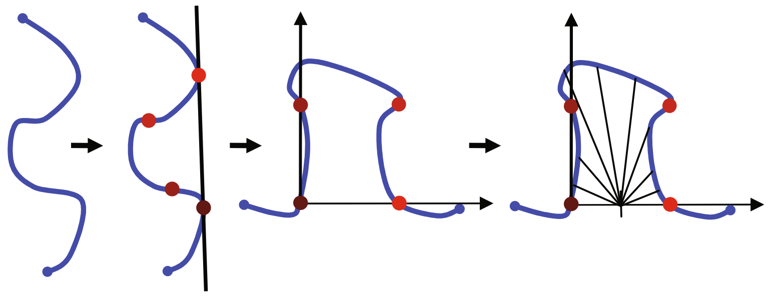


Fig. 5.5 Summarizing the derivation of canonical frame invariants as implemented in the LEWIS method, where curves of connected edge pixels which feature a concavity are exploited

For this reason a signature which is a projective invariant can be derived from the transformed curve. To this end the lengths of equally spaced rays ranging from the point $[1/2, 0]$ to the transformed curve are stacked into a vector and serve as a basis for the desired invariant I_V (details see [4]).

Figure 5.5 illustrates the proceeding: In a first step, four non-collinear points around a concavity are uniquely identified with the help of a common tangent (see [4] for details). Next, these four points are mapped onto a unit square. Subsequently, all points of the curve can be transformed to the thus defined canonical frame. The lengths of equally spaced rays (shown in black in the right part; all originating at the point $[1/2, 0]$) are the basis for the desired invariant. Observe that – compared to algebraic invariants – there are less restrictions on the shape of the object. However, the curve is not allowed to be arbitrarily shaped, as it is required to detect a common tangent passing through two distinct points of the curve.

5.4.3 Teaching Phase

The model database can be built iteratively by extracting the model data for each object class from a training image. For each model class, the data consists of a collection of edge pixels, geometric features (lines, conics and curves of connected edge pixels), and the invariant values derived from the features. The data is obtained as follows:

1. *Identification of edge pixels*: In the first step all edge pixels with rapidly changing intensity are found, e.g., with the operator proposed by Canny [1] (including non-maximum suppression). For many man-made, planar objects, the set of thus obtained edge pixels, captures most of the characteristics.
2. *Feature extraction*: Subsequently, all primitives which could potentially be part of a configuration being suitable for derivation of invariants (namely lines, conics and curves) are extracted from the edge image.

3. *Feature grouping*: Now the lines and cones are grouped to one of the three configurations from which algebraic invariant values can be derived (see Table 5.3), if possible.
4. *Invariant calculation*: Subsequently, several invariant values can be calculated and added to the object model, algebraic invariants, as well as canonical frame invariants.
5. *Hash Table creation*: For speed reasons, the invariant values can be used to create hash tables $\mathbf{H}_{L1}, \mathbf{H}_{L2}, \mathbf{H}_{LC}, \mathbf{H}_{C1}, \mathbf{H}_{C2}$, and $\mathbf{H}_V$ (one table for each functionally independent invariant). Each table entry consists of a list of object models which feature an invariant of appropriate type and with a value that falls within the hash table index bounds.

The data of each object model, which is available after teaching, essentially consists of the following:

- A list $\mathbf{e}$ of edge pixels
- Lists of lines $\mathbf{l}_M$, conics $\mathbf{c}_M$ and curves $\mathbf{v}_M$ which could be extracted out of $\mathbf{e}$.
- Feature groups $\mathbf{g}_{M,L}$ (5-line configurations), $\mathbf{g}_{M,LC}$ (2-line and conic configurations), $\mathbf{g}_{M,C}$ (2-conic configurations) which serve as a basis for invariant calculation.
- Invariant values $I_{L1,i}$, $I_{L2,i}$, $I_{LC,j}$, $I_{C1,k}$, $I_{C2,k}$, and $I_{V,l}$ derived from the entries of $\mathbf{l}_M$, $\mathbf{c}_M$, and $\mathbf{v}_M$.

5.4.4 Recognition Phase

Recognition of the objects shown in a scene image is performed by comparing invariants. To this end, invariants have to be derived from the scene image, too. Hence, steps 1–4 of recognition are identical to training. In the following, classification and verification are performed as follows:

5. *Hypothesis formulation by indexing*: in order to formulate a hypothesis for the occurrence of a specific object based on a specific invariant value the following two conditions must hold:

   – An invariant of the same type (e.g., based on five lines) exists in the model database.
   – The value of the model database invariant is similar to the scene image invariant value.

   A fast hypothesis formulation can be achieved by the usage of hash tables: each table entry, which covers a range of invariant values, consists of a list of all object models containing an invariant of the same type whose value also falls into this range. As we have different types of functionally independent invariants, multiple hash tables $\mathbf{H}_{L1}$, $\mathbf{H}_{L2}$, $\mathbf{H}_{LC}$, $\mathbf{H}_{C1}$, $\mathbf{H}_{C2}$, and $\mathbf{H}_V$ are used. At this stage, each invariant leads to a separate hypothesis. Based on the model data as well as the extracted scene image feature groups, the transformation parameters **t** can be derived.

6. *Hypothesis merging*: instead of a separate verification of each hypothesis it is advantageous to combine them if they are consistent. As a joint hypothesis is supported by more features, it is more reliable and the transformation parameters can be estimated more accurately. The merging process is based on topologic and geometric compatibility of different hypotheses, details can be found in [4].
7. *Hypothesis verification*: a (joint) hypothesis is verified if it is still broadly supported when all edge pixels and/or features (and not only the invariant values) are taken into account. Verification is performed by back-projecting the edge pixel point set as well as all extracted lines and conics of the model to the scene image. A hypothesis is accepted if more than a certain proportion of the model data is consistent with the scene image data. Two lines are regarded consistent if their orientation is similar, conics have to possess similar circumference and area and finally, edge pixels must have similar position and gradient orientation. Because back-projection of many edge pixels is expensive in terms of runtime, it is preferable to perform another verification in advance: in general, when calculating the eight parameters of the projective transform, it is possible to formulate an over-determined equation system. Over-determination should be possible, because the number of available features should exceed four non-collinear points which are necessary to determine eight transformation parameters. Consequently, if it is not possible to compute common transformation parameters where the error (due to the over-determination) remains small, the hypothesis can be rejected immediately. Otherwise the parameters just calculated can be used for the aforementioned back-projection verification.

5.4.5 Pseudocode

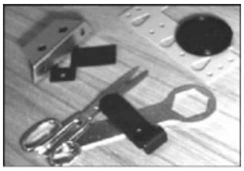
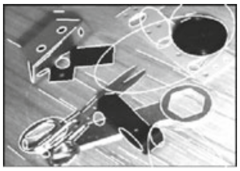
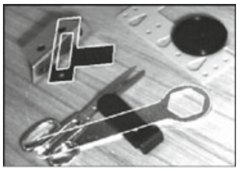
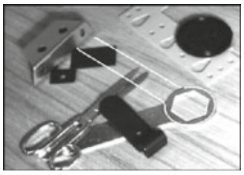
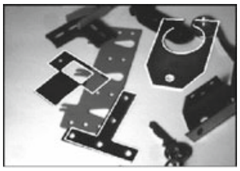
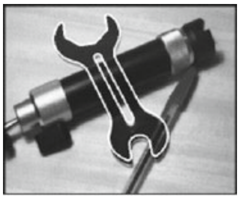
```
function detectObjectPosLEWIS (in Image $I$, in hash tables
$\mathbf{H}_{L1}$, $\mathbf{H}_{L2}$, $\mathbf{H}_{LC}$, $\mathbf{H}_{C1}$, $\mathbf{H}_{C2}$, and $\mathbf{H}_V$, in model data $M$ for
each object model, in similarity threshold $t_{sim}$, out object
position list $\mathbf{p}$)

// invariant calculation
detect all edge pixels (e.g. Canny) (summarized in $\mathbf{e}$)
group edge points $\mathbf{e}$ to line segments $l_{S,i}$, cones $c_{S,k}$ and
curves $v_{S,l}$ and add each found feature to one of the lists
$\mathbf{l}_S$, $\mathbf{c}_S$ or $\mathbf{v}_S$
detect all 5-line configurations $g_{S,L,i}$ and build list $\mathbf{g}_{S,L}$
detect all 2-line/1-conic configs $g_{S,LC,j}$ and build list $\mathbf{g}_{S,LC}$
detect all 2-conic configurations $g_{S,C,k}$ and build list $\mathbf{g}_{S,C}$
calculate algebraic invariants $I_{L1,i}$, $I_{L2,i}$, $I_{LC,j}$, $I_{C1,k}$ and
```

```
$I_{C2,k}$ for all elements of lists $\mathbf{g}_{S,L}$, $\mathbf{g}_{S,LC}$, and $\mathbf{g}_{S,C}$
(equations 5.6 - 5.11)
calculate canonical frame invariants $I_{V,l}$ for all elems of $\mathbf{v}_S$

// matching
// generation of hypotheses
for i = 1 to number of list entries in $\mathbf{g}_{S,L}$
   retrieve model list $\mathbf{h}_{M,L1}$ from $\mathbf{H}_{L1}$ (index specified
    by $I_{L1,i}$)
   for m = 1 to number of model entries in $\mathbf{h}_{M,L1}$
      estimate $\mathbf{t}$ based on $g_{S,L,i}$ and $g_{M,L,n}$
      add hypothesis $hy = [\mathbf{t}, m]$ to list $\mathbf{hy}$ // m: model index
   next
   repeat this proceeding with $I_{L2,i}$
next
repeat this proceeding with $\mathbf{g}_{S,LC}$ and $\mathbf{g}_{S,C}$ and $\mathbf{v}_S$ (here,
take the four non-collinear points for estimating $\mathbf{t}$)
// hypothesis merging
for i = 1 to number of hypotheses $\mathbf{hy}$ -1
   for j = i+1 to number of hypotheses $\mathbf{hy}$
      if $similarity\left(hy_i, hy_j\right)$ then
         $hy_k = merge\left(hy_i, hy_j\right)$
         replace $hy_i$ by $hy_k$ and delete $hy_j$
      end if
   next
next

// hypothesis verification
for i = 1 to number of hypotheses $\mathbf{hy}$
   $sim \leftarrow 0$
   verify lines: adjust $sim$ based on the position similarity
   between $l_{S,i}$ and $\mathbf{t}\left(l_{M,i}\right)$
   repeat this for all cones and edge pixels
   if $sim \geq t_{sim}$ then
      append $hy_i$ to position list $\mathbf{p}$
   end if
next
```

5.4.6 Example

The following examples show object poses found and verified by the LEWIS system as white overlays on the original gray scale scene images. They demonstrate that the system is able to detect multiple objects even in the presence of heavy occlusion and/or background clutter, but also disclose some limitations of the method.

Table 5.4 Recognition examples taken from the article of Rothwell et al. [4].[3] The threshold for the edge match in the verification step was set to 50%

		
Scene image containing seven planar objects which partly occlude each other. Two of them (spanner and lock striker plate) are part of the model database	Detected lines and conics superimposed in white. Altogether, 100 lines and 27 conics are found by the system	The two model objects are both found with correct pose: the striker plate with 50.9% edge match based on a single invariant, the spanner with 70.7% edge match based on three invariants
		
Example of a false positive due to clutter: the spanner was identified in wrong pose with 52.1% edge match	Three detected objects in another scene image with 74.4% edge match (2 invariants), 84.6% (1 inv.) and 69.9% (3 inv.) from left to right	Spanner detected with the help of canonical frame invariants with 55.5% edge match

5.4.7 Rating

As the examples show, it is indeed possible to utilize invariants for 3D object recognition of planar objects with only a single image independent of the viewpoint from which it was acquired. Experiments performed by the authors, where objects were rotated full circle with a certain step size, revealed that algebraic as well as canonical frame invariants remained stable (the standard deviation was at approximate 1.5% of the mean value). There are numerous examples where objects were found in spite of considerable occlusion and/or clutter. Moreover, the system is also capable of identifying multiple objects in a single scene image. Compared to an alignment approach, the number of hypotheses to be tested in the verification step could be reduced dramatically (by 2–3 orders of magnitude for some example images).

[3]With kind permission from Springer Science+Business Media: Rothwell et. al. [4], Figs. 27, 28, 32, and 42, © 1995 Springer.

On the other hand, however, the system sometimes tends to generate false positives, especially in the presence of heavy clutter. This is due to the fact that clutter leads to a dense occurrence of features. Consequently, sometimes a spurious solution can occur when an invariant calculated from clutter can be matched to the model database (one example is shown in Table 5.4). The system doesn't consider texture or, more generally, appearance information which could contribute to alleviate this effect. Additionally, the method only works well if objects are suited, i.e., if they contain several feature primitives like lines or conics. More seriously, many features have to be detected reliably in order to make the method work. This makes the invariant calculation instable: for example, if only a single line of a five-line group is occluded by another object, it is not possible to calculate the corresponding invariant value any longer. In the meantime it is common sense that this limitation is the main drawback of methods relying on invariants.

References

1. Canny, J.F., "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 8(6):679–698, 1986
2. Costa, M. and Shapiro, L., "3D Object Recognition and Pose with Relational Indexing", *Computer Vision and Image Understanding*, 79:364–407, 2000
3. Lowe, D.G., "Three-Dimensional Object Recognition from Single Two-Dimensional Images", *Artificial Intelligence,* 31(3):355–395, 1987
4. Rothwell, C.A., Zisserman, A., Forsyth, D.A. and Mundy, J.L., "Planar Object Recognition using Projective Shape Representation", *International Journal of Computer Vision*, 16:57–99, 1995
5. Zisserman, A., Forsyth, D., Mundy, J., Rothwell, C., Liu, J. and Pillow, N., "3D Object Recognition Using Invariance", *Artificial Intelligence,* 78(1–2):239–288, 1995