What is observable is the sequence of observations generated by the second mechanism. The sequence of observations is $\boldsymbol{o} = [o_i]$ chosen from a finite set of observation symbols

$$O = \{O_1, O_2, \ldots, O_{\mathrm{L}}\} \ .$$

An observation $o_i \in O$ is generated when the model is in state S_i. The output symbol is chosen depending on the matrix of output probabilities

$$\begin{aligned} \boldsymbol{B} \quad &= \quad [b_{il}] = P(O_l \text{ emitted in state } S_i | s_n = S_i \ ) \\ & \qquad i = 1, \ldots, I \ , \quad l = 1, \ldots, L \ . \end{aligned}$$

A hidden *Markov* model is defined by the tripel

$$\mathrm{HMM} = (\pi, \boldsymbol{A}, \boldsymbol{B}) \ . \tag{3.59}$$

It may be viewed as a stochastic automaton generating a stochastic process. The set of states is chosen according to the task domain. For example, in speech recognition it is a set of subword units of a language, in gesture recognition it may be a set of motion states. For each class, e.g. for each word or each gesture, a separate HMM is defined and its parameters are trained from classified observations. For a new observation the posterior probability is computed that a particular HMM was active when this observation is available. The most probable HMM is chosen according to Equation 3.36. An HMM may be represented by a graph. The nodes are the states, and a link between two nodes is introduced if there is a non-zero transition probability between the two corresponding states.

The three basic computational problems for HMM's are:

- Given an observation sequence $\boldsymbol{o}$ and an HMM with its parameters, what is the *probability* $P(\boldsymbol{o}|\mathrm{HMM})$ of generating $\boldsymbol{o}$ by this HMM?
- Given an $\boldsymbol{o}$ and HMM, what is the most probable *sequence of states* in HMM for generating $\boldsymbol{o}$?
- Given several observations, what are the *parameter values* $\pi, \boldsymbol{A}, \boldsymbol{B}$ maximizing $P(\boldsymbol{o}|\mathrm{HMM})$?

Efficient algorithms to solve these problems are available from the references [564, 625].

3.4 OBJECT RECOGNITION

D. PAULUS

The mapping of real world objects to the image plane including the geometric and the radiometric parts of image formation is basically well understood [766, 219].

From an abstract point of view, a camera is thought of as a geometric engine that projects the 3D world to the 2D image space. The most challenging problems in computer vision which are still not solved entierly are especially related to object

recognition [708]. Up to now, there have been no general algorithms that allow the automatic learning of arbitrary 3D objects and their recognition and localization in complex scenes. The term *object recognition* denotes two problems [708]: classification of an object and determination of its pose parameters. By definition, the recognition requires that *knowledge* or *models* of the object are available; formalisms for such knowledge are introduced in Section 3.5. The key idea is to compare the image with a model. The key issues thus are the choice of the representation scheme, of the selection of models, and the method for comparison.

Two major types of objects be distinguished: On the one hand, solid objects have to be recognized. Such problems arise, e.g., in industrial applications where a construction robot assembles objects from several parts. On the other hand, flexible or deformable objects impose further problems on the recognition algorithm. Such 'objects' are, e.g., humans or animals in observation systems.

3.4.1 Introduction

We assume that N_{K} object classes Ω_κ $(1 \leq \kappa \leq N_{\mathrm{K}})$ are known and represented as 'knowledge' (i.e., models) in an appropriate manner. The representation of the object can be in two dimensions, it may use a full 3D description, or it can contain a set of 2D views of a 3D object. The object models use an object coordinate system and a reference point (mostly on the object) as its origin.

We also assume that an image is given which may contain data in 2D, 2.5D or 3D. For intensity images in 2D it may be either monochrome, color or a multi-channel image. The object recognition problem can be formalized when the following sub-problems are identified:

The *object classification* problem is to identify an object in the given image $\boldsymbol{f}$. This may be particularily hard if the object is partially occluded, if many objects are in the scene, if the object is small, etc. This part of the problem formally corresponds to a function δ

$$\delta : \boldsymbol{f} \rightarrow \kappa; \kappa \in \{1 \ldots N_{\mathrm{K}}\} \quad . \tag{3.60}$$

The *pose estimate* problem is to compute pose parameters $\boldsymbol{R}$ and $\boldsymbol{t}$ for an object Ω_κ, which determine the transformation of the camera coordinate system to object coordinates (Figure 3.12). Depending on the dimension of the input data and on the object representation, these may be two-dimensional or three-dimensional rotation matrices and translation vectors.

In the following we assume a 3D world containing solid objects and a 2D image. In projective geometry, the transformation parameters can be divided into two groups. There are several representations for the transformation parameters. In [554, 610], in-plane and out-of-plane transformations are distinguished (see Figure 3.12). The rotation matrix $\boldsymbol{R}$ can be described elegantly by the Rodrigues formula, quaternions or by Euler angles (cmp. [219]). Advantages and disadvantages of these representations for object recognition purposes are discussed in [335]

The algorithms for 3D object recognition published so far differ with respect to

- the type and dimension of sensor data,

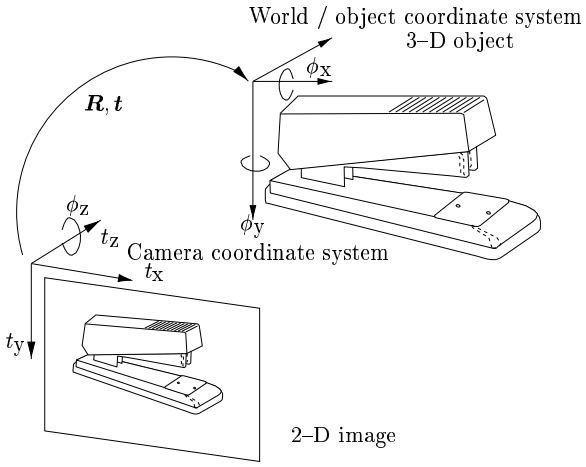

Figure 3.12. Different components of rotation and translation of a 3D object.

- the representation scheme for object models,
- the object classifi cation strategy,
- the strategy for pose recognition, and
- the learning of object models.

It is beyond the scope of this section to provide an exhaustive overview and a lucid discussion of all models successfully applied in computer vision. Instead, we describe the most common and some new approaches and give a general outline of their characteristics.

The organization of the section is as follows: The next section introduces an example of how geometric features can be used for object classifi cation and pose determination. As an example of appearance based methods, we introduce eigenspaces in Section 3.4.3. Arguments for a probabilistic formulation of object recognition modules are listed in Section 3.4.4 and one example of an approach using segmented features as well as one statistical appearance based method is mentioned. The use of invariant features for object recognition is outlined in Section 3.4.5. The architecture of a system for active object recognition is presented in Section 3.4.6. We summarize our review in Section 3.4.7.

3.4.2 Object Recognition Using Geometric Models

State-of-the-art approaches dealing with high-level vision tasks are essentially dominated by model-based object recognition methods [360]. As we outline in the following, such models consist of primitives that can be compared directly to the information found in the segmented images. Whereas the purpose of geometric models is to facilitate the generation of images and models thus had to be similar to the visualization

pipeline, the models suitable for object recognition need to have similar structure as segmentation objects (see Section 3.1).

The classical approach is to use geometric models and to represent them explicitly [19]. Generally speaking, segmentation algorithms decompose given images in primitives, reference models are matched to observations, and distance measures are the basis for class decisions and pose estimates. The selection of discriminating image features and the adequate representation of object models defi ne herein the vital problems which have to be solved when designing a vision system. We assume that a segmentation object $\boldsymbol{O}$ is created from an input image $\boldsymbol{f}$ and that an object model C exists which is in a format compatible with the segmentation data. A (partial) match between $\boldsymbol{O}$ and C is found when a subset of the data in the model is found to correspond to a subset of the segmentation object. A function

$$\zeta : C \rightarrow \boldsymbol{O} \tag{3.61}$$

is called an *interpretation*. The information in the model constrains the admissible elements in the segmentation object, e.g., by requirng parallel lines in the three-dimensional object model to be mapped to approximately parallel lines in the segmentation object. Finding an interpretation, generally is a search problem in which data resulting from image segmentation has to be compared to data in models. Since during search this comparison has to be done frequently, it is required for effi ciency reasons that segmentation data and model data are compatible in the sense, that a direct comparison of primitives is possible. In particular, effi cient and roboust mappings of data represented in model primitives to that represented in segmentation data have to exist. This is not the case, e.g. for CAD models as used in Section 4.3, since these models contain lines which will never be segmented in the image. At least some of the lines prepresented in the model are not present in the object; as an example we can use wire frame models. As another example, the control points in triangulated surfaces will not be detected roboustly in the image data, either. Free form surfaces represented as splines, as proposed in Section 4.6 are parametrized by the spline parameters and their control points, which usually cannot be detected by segmentation, either.

Figure 3.13 shows an example of a surface model $C = \{c_1, \ldots, c_9\}$ and an idealized segmented range image consisting of surface segments in a segmentation object $\boldsymbol{O} =$

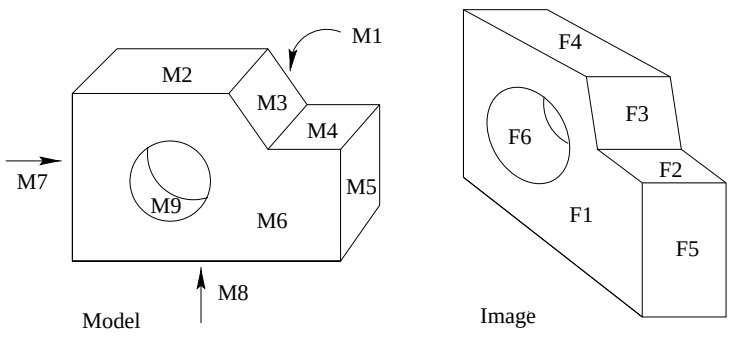

Figure 3.13. Matching of segmented features to a model.

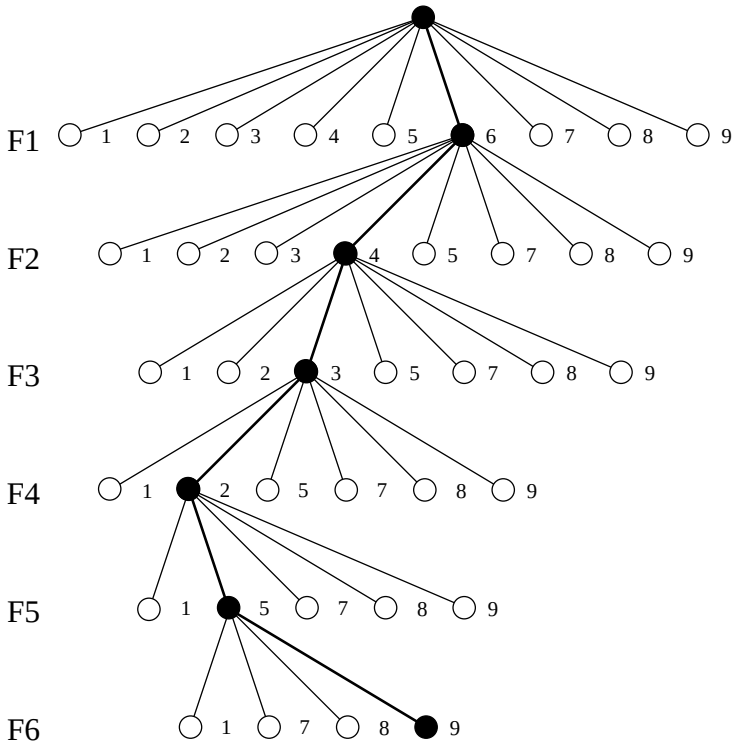

Figure 3.14. Search space for model match for the object in Figure 3.13.

$\{o_1, \ldots, o_6\}$. We constrain the matches such that L-shaped segments may only be matched to the L-shaped model elements $\{c_1, c_6\}$. A depth-first search of a match in Figure 3.14 left results in an incorrect interpretation assignment. In a subsequent verification by backprojection of the hypothesized model parameters to the image and comparison of the features it is required to find the correct match which is shown as a dotted in Figure 3.14 right. The general problem is that features are only compared locally. Since real image segmentation is inaccurate and not complete, a perfect match will almost never be found for real data. One solution is to add 'wildcards' to the segmentation and to the model when processing the interpretation tree. These are nodes which can be matched to any segment or model; a wildcard model is matched to a superfluous segment node; the same technique is used for missing segments [708]. More elaborate matching strategies for semantic models are outlined in Section 3.3.

The pose recognition problem for geometric models and matching of geometric features is strongly related to the problems of 3D reconstruction (see Chapter 4) and camera calibration (see Section 2.1). As noted in Section 3.4.1, the problem is to determine $\boldsymbol{R}$ and $\boldsymbol{t}$ for an object Ω_κ. Assuming a perfect match, we have a set of points

$$\{\boldsymbol{x}_1^{(w)}, \boldsymbol{x}_2^{(w)}, \ldots, \boldsymbol{x}_{N_\mathrm{m}}^{(w)}\} \tag{3.62}$$

in world coordinates corresponding to points $\{\boldsymbol{x}_1^{(C)}, \boldsymbol{x}_2^{(C)}, \ldots, \boldsymbol{x}_{N_\mathrm{m}}^{(C)}\}$ in camera coordinates related by

$$\boldsymbol{x}_i^{(C)} = \boldsymbol{R}\boldsymbol{x}_i^{(W)} + \boldsymbol{t} \quad . \tag{3.63}$$

What is observed are the image coordinates of the corresponding projected points $\{\boldsymbol{x}_1^{(I)}, \boldsymbol{x}_2^{(I)}, \ldots, \boldsymbol{x}_{N_\mathrm{m}}^{(I)}\}$. Assuming that we are dealing with 3D world coordinates and 2D images, the complete transformation including the projection step can be expressed a 4×3 matrix $\boldsymbol{E}$ if points are written in homogeneous coordinates [219]. Depending on whether the projection parameters are known from calibration or not, and whether the images are registered (see Section 5.1), several strategies exist for the computation of this matrix, see e.g. in [61, 593].

3.4.3 Appearance Based Object Recognition

Appearance based object recognition uses non-geometric models representing the intensities in the projected image. Rather than using an abstract model of geometries and geometric relations, images of an object taken from different viewpoints and under different lighting conditions are used as object representation. Figure 3.15 shows a set of such images. To beat the curse of dimenstionality, the images used for object representation are transformed to lower-dimensional feature vectors. This overcomes several problems related to standard approaches as, for example, the geometric modeling of fairly complex objects and the required feature segmentation. Comparative studies prove the power and the competitiveness of appearance based approaches to solve recognition problems [551]. Well-known and classical pattern recognition algorithms can be used for computer vision purposes, if appearance based methods are applied: feature selection methods [572, 65], feature transforms [572, 708], or even more recent results from statistical learning theory [723].

As the given image and the model share the same representation, the choice of the distance function for matching images with models is simpler than for geometric models. We rearrange the image pixels $f_{i,j}$ in an image vector

$$\boldsymbol{f}' = (f_{1,1}, \ldots, f_{1,N_x}, f_{2,1}, \ldots, f_{2,N_x}, \ldots, f_{N_y,1}, \ldots, f_{N_y,N_x})^T \tag{3.64}$$

The comparison of two images $\boldsymbol{f}_1'$ and $\boldsymbol{f}_2'$ by correlation simply reduces to the dot product of the image vectors $\boldsymbol{f}_1'$ and $\boldsymbol{f}_2'$

$$s = \boldsymbol{f}_1'^T \cdot \boldsymbol{f}_2' \quad ; \tag{3.65}$$

the bigger s gets, the more similar are the images $\boldsymbol{f}_1'$ and $\boldsymbol{f}_2'$.

Obviously, high dimensional feature vectors such this image vector will not allow the implementation of effi cient recognition algorithms [507]. The vectors have to be transformed to lower dimensions. Commonly used transforms are the principal component analysis [494, 376, 120] or in more recent publications the Fisher transform [50]. In the following we motivate a linear tranformation Φ which maps the image vector $\boldsymbol{f}' \in \mathbb{R}^{N_x \cdot N_y}$ to a feature vector $\boldsymbol{b} \in \mathbb{R}^{L_a}$ with $L_a \ll N_x \cdot N_y$ by

$$\boldsymbol{b} = \Phi \boldsymbol{f}' \quad . \tag{3.66}$$

If we choose Φ such that the distance of all features is maximized, this reduces to a problem of eigenvalue computation. From N_a given images written as vectors $\boldsymbol{f}'_1, \ldots \boldsymbol{f}'_{N_a}$ of an object we compute the mean vector

$$\mu = \frac{1}{N_a} \sum_{i=1}^{N_a} \boldsymbol{f}'_i \tag{3.67}$$

and from this we create a matrix $\boldsymbol{V}$ whose columns are the image vectors

$$\boldsymbol{V} = \left[(\boldsymbol{f}'_1 - \mu) | \ldots | (\boldsymbol{f}'_{N_a} - \mu) \right] \quad . \tag{3.68}$$

Eigenvalue analysis of the matrix $\boldsymbol{K} = \boldsymbol{V}^T \boldsymbol{V}$ yields the eigenvectors $\boldsymbol{v}_1, \ldots \boldsymbol{v}_{N_a}$ sorted by magnitude of the corresponding eigenvalues. A fundamental fact from linear algebra states that an image vector $\boldsymbol{f}'_j$ can be written as a linear combination of the mean image vector and the eigenvectors as

$$\boldsymbol{f}'_j = \mu + \sum_{i=1}^{N_a} b_i^{(j)} \boldsymbol{v}_i \quad . \tag{3.69}$$

An approximation of $\boldsymbol{f}'$ can be obtained if instead of N_a eigenvectors we select only the first $L_a \leq N_a$ vectors. The image vector $\boldsymbol{f}'_j$ is represented by

$$\boldsymbol{b}^{(j)} = (b_1^{(j)}, \ldots, b_{L_a}^{(j)})^T = \Phi \left(\boldsymbol{f}'_j - \mu \right) \tag{3.70}$$

and the columns of the matrix Φ are the first L_a eigenvectors $\boldsymbol{v}_1, \ldots \boldsymbol{v}_{L_a}$.

Typically for $N_a = 100$ images we choose only the first $L_a = 15$ eigenvectors. For each object class κ we now record images from different viewpoints and under changing lighting conditions, perform the transformation to eigenspace to obtain

$$\left\{ \boldsymbol{b}^{(\kappa,j)}; j = 1 \ldots N_{a\kappa} \right\} \tag{3.71}$$

for $N_{a\kappa}$ images captured. The vectors $\boldsymbol{b}^{(\kappa,j)}$ of an object of class κ are a manifold in eigenspace. They are used and stored as the object model. The processing steps of this approach are exemplified in Figure 3.15.

The correlation of two normalized images $\boldsymbol{f}'_i$ and $\boldsymbol{f}'_j$ with $||\boldsymbol{f}'_i|| = 1$ can now be approximated by the Euclidian distance of two weight vectors $\boldsymbol{b}^{(i)}$ and $\boldsymbol{b}^{(j)}$ which yields a huge gain in computation speed:

$$||\boldsymbol{f}_i'^T \boldsymbol{f}'_j|| \approx 1 - 0.5||\boldsymbol{b}^{(i)} - \boldsymbol{b}^{(j)}|| \quad . \tag{3.72}$$

For object recognition of a given image we compute its eigenspace representation to create a vector $\boldsymbol{b}$ using Equation 3.70. From the manifolds representing the objects we choose the one which has minimal distance to the computed vector $\boldsymbol{b}$. Object recognition is thus reduced to the problem of finding the minimum distance between

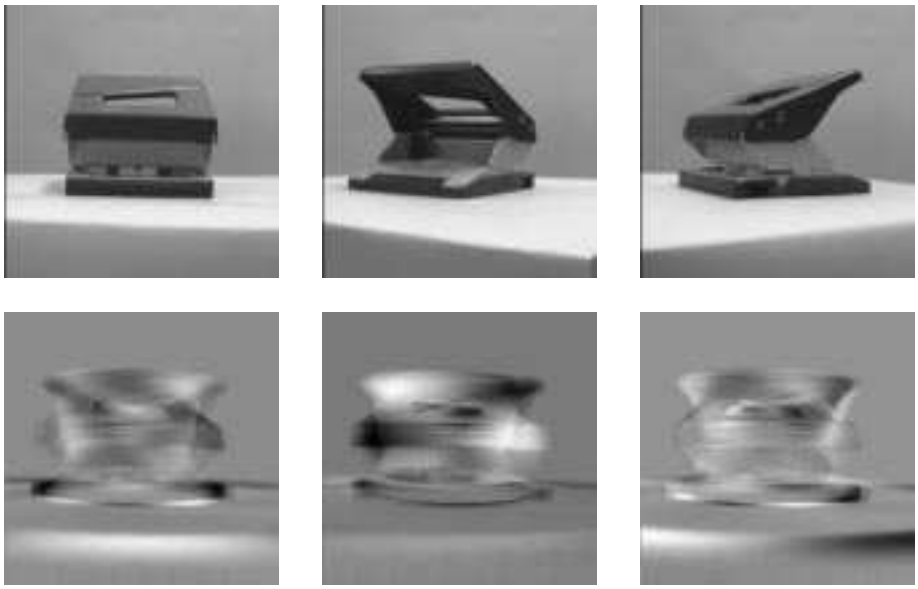

Figure 3.15. Three different views of an object (upper row), mean vector, and eigenvectors $\boldsymbol{v}_0, \boldsymbol{v}_{15}$ (lower row). For the computation 72 views and 360^o rotation in 5^o-steps were used. The images are taken from the COIL-data set.

an object and a model. In order to generate an image from a vector $\boldsymbol{b}$, we use the pseudo-inverse Φ^+ of Φ

$$\Phi^+ = \Phi^T \left(\Phi \, \Phi^T\right)^{-1} \tag{3.73}$$

to create

$$\boldsymbol{f'} = \Phi^+ \boldsymbol{b} \quad . \tag{3.74}$$

The key to success in this approach is not to create the matrix

$$\boldsymbol{K} = \boldsymbol{V} \ \boldsymbol{V}^T \tag{3.75}$$

explicitly when the eigenvectors are computed. For a typical image f of size $N_x = 256$ and $N_y = 256$, the image vector $\boldsymbol{f'}$ has length 2^{16}; for $N_a = 100$ images, the matrix $\boldsymbol{V}$ has size $2^{16} \times 100$; the matrix $\boldsymbol{K}$ would thus be of size $2^{16} \times 2^{16}$ and computation of the eigenvectors would be unfeasible. Instead, we use either iterative methods to compute the eigenvectors [517] or we use a result from singular value decomposition. We compute the eigenvalues λ_i and eigenvectors $\boldsymbol{v}'_j$ of the so-called implicit matrix

$$\boldsymbol{K'} = \boldsymbol{V}^T \ \boldsymbol{V} \tag{3.76}$$

which is much smaller than $\boldsymbol{K}$. In our example, the size would be 100×100. We note that

$$\boldsymbol{K'} \boldsymbol{v}'_j = \boldsymbol{V}^T \left(\boldsymbol{V} \boldsymbol{v}'_j\right) = \lambda_j \boldsymbol{v}'_j \quad . \tag{3.77}$$

We multiply Equation 3.77 from left by $\boldsymbol{V}$ and get

$$\boldsymbol{V}\left(\boldsymbol{V}^T\,\boldsymbol{V}\right)\boldsymbol{v}'_j = \left(\boldsymbol{V}\;\boldsymbol{V}^T\right)\boldsymbol{V}\,\boldsymbol{v}'_j = \lambda_j\left(\boldsymbol{V}\,\boldsymbol{v}'_j\right) \quad . \tag{3.78}$$

which shows that the eigenvalues of σ' are also eigenvalues of σ and that the eigenvectors are related by $\boldsymbol{V}$. We use these results to compute the eigenvectors for $\boldsymbol{K}$.

3.4.4 Probabilistic Object Recognition

Probabilistic methods as presented in Section 3.2 are used in several object recognition systems. A Bayesian framework for 3D object recognition requires that the appearance of objects in the image plane is characterized using probability density functions. These densities have to incorporate prior knowledge on objects, rotation and translation, self-occlusion, projection to the image space, the assignments of image and model features as well as the statistical modeling of errors and inaccuracies caused by varying illumination, sensor noise or segmentation errors [335]. We call these densities *model densities*.. The structure of these models can vary: It can be a single multivariate Gaussian density, a hidden Markov model or some other type of density (see Section 3.2).

In [337] we assume N_{K} possible object classes and a set of observations $\boldsymbol{O}$ consisting of feature vectors o_k in a segmentation object $\boldsymbol{O} = \{o_k \in \mathbb{R}^2 | 1 \leq k \leq N_{\mathrm{m}}\}$ where the number N_{m} of observed features like corners or vertices varies for different images. Appearance and position of features in the image show a probabilistic behavior. The statistical description of an object belonging to class Ω_κ consists of a model density $p(\boldsymbol{O}|\boldsymbol{B}_\kappa, \boldsymbol{R}, \boldsymbol{t})$ (see Section 3.2) combined with discrete priors $p(\Omega_\kappa)$ $1 \leq \kappa \leq N_{\mathrm{K}}$ for the probability of an object of class Ω_κ to appear in the scene. The priors are estimated by relative frequencies of objects in the training samples. The set $\boldsymbol{B}_\kappa$ contains the model-specific parameters for the behavior of features as well as the parameters for the assignment of image and model features.

For the explicit definition of $p(\boldsymbol{O}|\boldsymbol{B}_\kappa, \boldsymbol{R}, \boldsymbol{t})$ we use the observed feature set $\boldsymbol{O}$ and the corresponding features $C_\kappa = \{c_{\kappa,1}, c_{\kappa,2}, \ldots, c_{\kappa,n_\kappa}\}$ in the model where in general $n_\kappa \neq N_{\mathrm{m}}$ due to segmentation errors and occlusion. Let the parametric density of the model feature c_{κ,l_k} corresponding to o_k be given by $p(c_{\kappa,l_k}|\boldsymbol{a}_{\kappa,l_k})$, where $\boldsymbol{a}_{\kappa,l}$ $(l = 1, \ldots, n_\kappa)$ characterize model features. For a normally distributed 3D point, for instance, $\boldsymbol{a}_{\kappa,l_k}$ denotes the mean vector and the covariance matrix. A standard density transform results in the density $p(o_k|\boldsymbol{a}_{\kappa,l_k}, \boldsymbol{R}, \boldsymbol{t})$, which characterizes the statistical behavior of the feature o_k in the image plane dependent on the object's pose parameters.

As described in Section 3.2, the probabilistic modeling of the assignment from image to model features is based on discrete random vectors $\zeta_\kappa = (\zeta_\kappa(o_1), \zeta_\kappa(o_2), \ldots, \zeta_\kappa(o_{N_{\mathrm{m}}}))^T$, where ζ_κ defines a discrete mapping from an observed feature o_k to the index $l_k \in \{1, 2, \ldots, n_\kappa\}$ of the corresponding model feature c_{κ,l_k}, i.e., $\zeta_\kappa(o_k) = l_k$. The non-observable assignment is eliminated by the marginalization as shown in Equation 3.51. The probabilistic modeling of the assignment from image to model features is based on discrete random vectors. An assignment function ζ_κ defines a discrete mapping from an observed feature o_k to the index $l_k \in \{1, 2, \ldots, n_\kappa\}$ of the corre-

sponding model feature c_{κ,l_k}, i.e., $\zeta_\kappa(o_k) = l_k$. A set of observed features can thus be associated with the assignment random vector $\zeta_\kappa = (\zeta_\kappa(o_1), \zeta_\kappa(o_2), \ldots, \zeta_\kappa(o_{N_\mathrm{m}}))^T$ which is related to the discrete probability $p(\zeta_\kappa)$, i.e., the matching problem is also modelled statistically. The discrete probability of $p(\zeta_\kappa)$ extents the probability density function for observing the set of features $\boldsymbol{O}$. Due to the statistical interpretation of ζ_κ, the non-observable assignment can be eliminated by the following marginalization:

$$p(\boldsymbol{O}|\boldsymbol{B}_\kappa, \boldsymbol{R}, \boldsymbol{t}) \quad = \quad \sum_{\zeta_\kappa} p(\zeta_\kappa) \prod_{k=1}^{N_\mathrm{m}} p(o_k|\boldsymbol{a}_{\kappa,\zeta_\kappa(o_k)}, \boldsymbol{R}, \boldsymbol{t}) \quad . \tag{3.79}$$

If the structure of the model density (i.e., the number of model features and the dependency structure of single assignments) is known, algorithms for the estimation of the parameter set $\boldsymbol{B}_\kappa$ exist [335]. The computation of $\boldsymbol{B}_\kappa$ for each object class Ω_κ, $\kappa = 1, 2, \ldots, N_\mathrm{K}$ requires $p(\zeta_\kappa)$ and $\{\boldsymbol{a}_{\kappa,l}\}$. Due to the projection of the 3D world to the 2D image plane, the range information is lost. Furthermore, the assignment of image and model features is not a component of the observations. The calculation of $\boldsymbol{B}_\kappa$ thus corresponds to an incomplete data estimation problem which can be solved using the Expectation Maximization algorithm (EM algorithm, cmp. Section 3.2).

The framework introduced so far requires a minor modification of the standard Bayesian decision rule, since a segmentation object $\boldsymbol{O}$ is given instead of a single vector, and the unknown pose parameters are part of the probability density. The modified Bayesian decision rule for the statistical classification of objects is:

$$\lambda = \mathrm{argmax}_\kappa \; p(\Omega_\kappa|\boldsymbol{O}) \quad = \quad \mathrm{argmax}_\kappa \; p(\Omega_\kappa) p(\boldsymbol{O}|\boldsymbol{B}_\kappa, \boldsymbol{R}, \boldsymbol{t}) \quad . \tag{3.80}$$

The a posteriori probabilities $p(\Omega_\kappa|\boldsymbol{O})$ cannot be evaluated explicitly. The pose estimation stage has to compute the best orientation and position $\boldsymbol{R}$, $\boldsymbol{t}$ before the class decision is possible. This corresponds to the maximization problem

$$\{\widehat{\boldsymbol{R}}, \widehat{\boldsymbol{t}}\} = \mathrm{argmax}_{\boldsymbol{R},\boldsymbol{t}} p(\boldsymbol{O}|\boldsymbol{B}_\kappa, \boldsymbol{R}, \boldsymbol{t}) \quad , \tag{3.81}$$

which requires a global optimization of a multimodal likelihood function.

This framework was used for the recognition of 3D objects based on 2D images [335]: we assume that each input image (e.g. Figure 3.16) is transformed into a segmentation object of 2D feature vectors $\boldsymbol{O} = \{o_k \in \mathbb{R}^2 | 1 \leq k \leq N_\mathrm{m}\}$. The

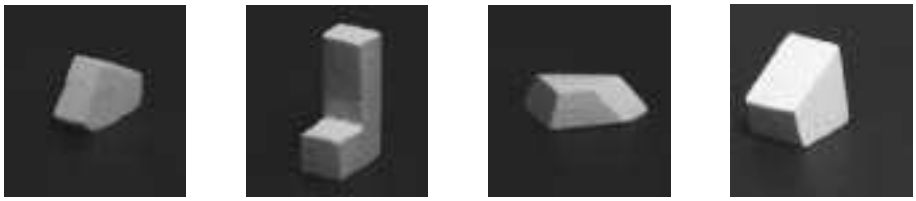

Figure 3.16. Simple polyhedral 3D objects ($\Omega_1, \Omega_2, \Omega_3, \Omega_4$), used in the experiments in [335].

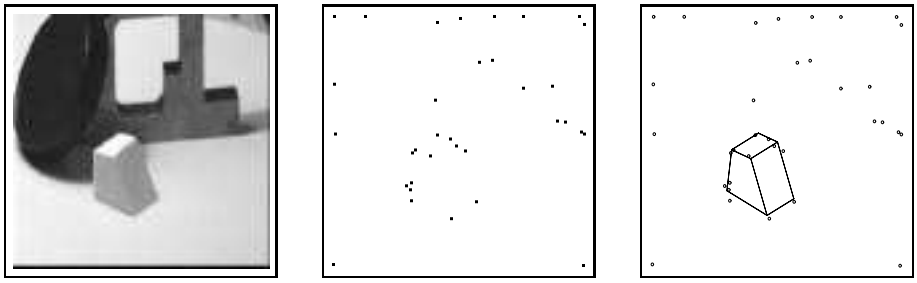

Figure 3.17. Experiment for object recognition with heterogeneous background.

elements o_k may be points (e.g. corners or vertices) or lines, which can be detected by several combinations of segmentation operators which all result in the uniform segmentation object (see Section 3.1).

For segmented 2D point features, $\boldsymbol{B}_\kappa$ provides the parameters characterizing the assignments as well as the accuracy and stability of the object points. Closed form iteration formulas can be found for normally distributed point features, which allow the estimation of mean vectors from projections without knowing corresponding features of different views [335].

This flexible formalism of model densities can also be extended to use multiple views for pose estimation or classification [335] which remarkably improves recognition rates.

A combination of appearance based methods with probabilistic approaches is also feasible. In [554], wavelet features in scale space are used for object representation, requireing no image segmentation. These features are modeled statistically and localization as well as classification is again reduced to a Bayesian decision and statistical parameter estimation problem.

3.4.5 Features and Invariants for Object Recognition

In Section 3.4.2, segmented features are matched to model features. The match is constrained by knowledge on the features, such as, e.g., parallelism. Such information can be provided by segmentation algorithms and can be represented in the relation-slot of a segmentation object. Although appearance based methods use the image directly and do not compute segmented features, they do require that the object figure is separated from the ground. This is usually achieve by image segmentation as well.

If we can find features of an object that are invariant with respect to a certain class of transformations, e.g., illumination changes, changes in size, viewpoint changes, etc., we can use these features to index into a set of models. Clearly, these features must have at the same time invariant properties as well as the discriminative power to separate object classes.

Several intuitive invariant features can be defined on 2D lines and contours. To give an example, the area of a region divided by its contour length is invariant to rotation, scaling, and translation. More complex invariant features can be found which yield, e.g., affine invariant descriptors.

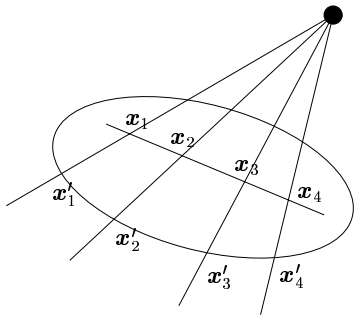

Figure 3.18. Circle skewed to an ellipse under projective transformation and its invariant properties using the cross ratio.

A well-known projective invariant which is of practial interest is the cross-ratio of four 2D points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$, and $\boldsymbol{x}_4$ on a straight line

$$\frac{||\boldsymbol{x}_1 - \boldsymbol{x}_3||}{||\boldsymbol{x}_1 - \boldsymbol{x}_4||} \cdot \frac{||\boldsymbol{x}_2 - \boldsymbol{x}_4||}{||\boldsymbol{x}_2 - \boldsymbol{x}_3||} \quad . \tag{3.82}$$

In fact, the cross ratio is invariant under any collineation, i.e., under every transformation which transforms straight lines into straigth lines. Straight lines can be found with reasonable reliability by image segmentation (see Section 3.1). The four points on such lines can be identifi ed, e.g., when line intersections are searched. The geometric relations for the cross ration are outlined in Figure 3.18.

Acquiring an object model using this approach is simple, as we represent an object class by a set of invariant features. We use one image of each object class to compute feature vector for each object class and store it. For recognition we detect the features in the given image and compare to the stored feature vectors.

Color in many cases provides very useful cues to identify objects. It is relatively insensitive to view point changes and can be regarded as a non-geometric feature, which is invariant with respect to minor changes in the position parameter of an object. Changes which are caused by changes in the lighting conditions have be eliminated to provide such invariance properties.

In order to identify color objects in a scene, color histograms are used in [679]. A color image $[\boldsymbol{f}_{ij}]_{1 \le i \le N_y, 1 \le j \le N_x}$ is searched for an object which is characterized by its histogram in some quantization. The color space is divided into cubes and the number of vectors is determined which fall into the cubes. This number is divided by the total number of color vectors. The histogram is denoted by $\boldsymbol{T} = [T_l]_{l=1 \ldots N_\mathrm{L}}$ where N_L denotes the number of cubes. A function ζ maps a color pixel to the index in the histogram, and permits to use arbitrary quantizations; e.g. for an RGB histogram with $4 \times 4 \times 4$ bins ($L = 64$) and for color components in the range from 0 to 255 we might choose

$$\zeta : \begin{cases} \mathbb{R}^3 & \to \{1, \ldots, L\} \\ \boldsymbol{f}_{ij} = (r_{ij}, g_{ij}, b_{ij})^T & \to \lceil r_{ij}/64 \rceil \cdot 16 + \lceil g_{ij}/64 \rceil \cdot 4 + \lceil b_{ij}/64 \rceil \end{cases} . \tag{3.83}$$

The elements of the histogram $\boldsymbol{T}$ are defined as

$$T_l = \frac{1}{N_x N_y} \left| \{(i,j) | \zeta(\boldsymbol{f}_{ij}) = l, i = 1, \ldots, N_y, j = 1, \ldots, N_x\} \right| \quad . \tag{3.84}$$

A close-up view of the object is recorded and the histogram $\boldsymbol{T}$ is computed from this image. Then, an image $[\boldsymbol{f}'_{ij}]_{1 \leq i \leq N'_y, 1 \leq j \leq N'_x}$ of the scene is recorded and a color histogram $\boldsymbol{S} = [S_l]_{l=1 \ldots L}$ of this image is computed with the same quantization. The sizes of the images may however be different. The elements of the histogram $\boldsymbol{S}$ are defined as

$$S_l = \frac{1}{N'_x N'_y} \left| \left\{ (i,j) | \zeta(\boldsymbol{f}_{ij}) = l, i = 1, \ldots, N'_y, j = 1, \ldots, N'_x \right\} \right| \ . \tag{3.85}$$

The ratio histogram $\boldsymbol{Q} = [Q_l]_{1 \leq l \leq L}$ is computed from the object histogram $\boldsymbol{T}$ and the histogram $\boldsymbol{S} = [S_l]_{1 \leq l \leq L}$ of the image $\boldsymbol{f}$ by

$$Q_l = \begin{cases} 0 & \text{if } S_l = 0 \\ \min\left\{1, \frac{T_l}{S_l}\right\} & \text{otherwise} \end{cases} \quad . \tag{3.86}$$

If for a bin with index l is empty, i.e., $S_l = 0$, then the corresponding color is not present in the scene, i.e., no pixel can be found that is mapped to this bin. Thus, this case never occurs in backprojection. In addition, the approximate size of the object in the image is needed for the algorithm; this size is represented by a mask $\boldsymbol{D} = [\boldsymbol{D}_{ij}], \boldsymbol{D}_{ij} \in \{0, 1\}$ covering the object.

The values of the ratio histogram are thus in the range $[0, 1] \subset \mathbb{R}$. An intermediate image $\boldsymbol{g} = [g_{ij}](1 \leq i \leq N_y, 1 \leq j \leq N_x)$ of the same dimension as the input image is computed as $g_{ij} = Q_{\zeta(\boldsymbol{f}_{ij})}$ where the function ζ is used to find the appropriate bin for the color vector $\boldsymbol{f}_i j$ at position $(i,j)^T$. The convolution of $\boldsymbol{g}$ with the object mask $\boldsymbol{D}$ yields the output image $\boldsymbol{h}$. Local maxima in the output image indicate possible positions of the object.

A disadvantage of the color backprojection method is its sensitivity to illumination changes. This can be helped by preprocessing with a color constancy algorithm.

If an object is supposed to be at position $(i,j)^T$ in the image and the size is estimated to be $a \times b$, then it is reasonable to create a local histogram at position $(i,j)^T$ for a sub-image of the estimated size and to compare this histogram to the histogram given for the close-up view of the object. The size of this mask has to be known before using this method.

A simple distance measurement is the sum of distance squares (sum of squared differences, SSD):

$$SSD(\boldsymbol{S}, \boldsymbol{T}) = \sum_{l=1}^{L} (T_l - S_l)^2. \tag{3.87}$$

The following quadratic form can be used as weighted version of L_2-norm:

$$d_A(\boldsymbol{S}, \boldsymbol{T}) = \sqrt{(\boldsymbol{S} - \boldsymbol{L})^T \boldsymbol{A} (\boldsymbol{T} - \boldsymbol{L})} \tag{3.88}$$

where $\boldsymbol{A}$ weighs the color distances in the color channels. Statistical methods can be used to compare the histograms, for example, the χ^2-test:

$$\chi^2_{\boldsymbol{S},\boldsymbol{T}} = \sum_{l=1}^{L} \frac{(S_l - T_l)^2}{T_l + S_l} \tag{3.89}$$

The number of degrees of freedom is $L - 1 - |\{l|T_l = S_l = 0\}|$. If the histograms are given in absolute frequencies and their sum is equal, then this degree is reduced by one. More information on object recognition using color histograms can also be found in [610].

3.4.6 Active Object Recognition

For real scenes and computer vision problems of practical interest, the examples described so far are too much academic. Real-world scenes rarely contain isolated objects and homogeneous backgrounds, the scenes are rather cluttered, the objects are partially occluded, relatively small, etc.

The strategy of active vision which will be introduced in Section 3.6 can be successfully applied to combine recognition modules, 3D estimation, and model-based camera actions in order to solve the object recognition task in real-world indoor scenes [6]. This system actively moves the camera and changes the parameters of the lens to create high resolution detail views. Camera actions as well as top-level knowledge are explicitly represented in a semantic network; the control algorithm of the semantic network used to guide the search is independent of the camera control; this semantic network will be introduced in Section 3.6. Objects are hypothesized using color information (see Section 3.4.5) and verifyed using geometric matching (see Section 3.4.2) based on segmented color regions (see Section 3.1). The active component is necessary, since the objects are too small to be detected in the wide-angle view robustly, using any of the above mentioned methods.

An example scene containing office objects, object hypotheses by color backprojection, and close-up views of the objects is shown in Figure 3.19 (top). In a sequence of images which is recorded when the camera is moved latereally in front of the scene, interesting points are selected by one the methods proposed in Section 3.1.4. Tracking of these points and analysis of their trajectory is used to recover the distance of these points to the camera. Figure 3.19 (bottom) shows two projections of these points. This sparse 3D information permits to estimate the size of the object mask $\boldsymbol{D}$ used in histogram backprojection Figure 3.19 (bottom right).

3.4.7 Conclusion

Most systems on object recognition reported in the literature use models which are represented explicitely. The choices for object models are geometric descriptions, appearance-based models or statistical models. None of the approaches guarantees satisfactory results for general problems, if it is used in an isolated module. Instead, a combination of modules is required.

Figure 3.19. Office scene (top left), close up views used for hypotheses (top right), two projections of 3D points (bottom left), color backprojection for red punch (bottom right). The red punch searched for is shown in the close-up views on the top right.

Traditionally, geometric approaches were dominant e.g., in [333]. Recently, statistical methods are gaining more and more interest, e.g., in [262, 335, 406]. Support vector machines as proposed in [723, 708] promise a further step into this direction.

Although the object recognition problem is a central task in almost any computer vision system, it is still not completely solved and there is still room for improvement.

3.5 IMAGE UNDERSTANDING

H. NIEMANN

The term model-based image understanding is used in different meanings, of which we mention in particular:

- The interpretation of images or image sequences using a semantic model in the sense of Section 3.2.

  Examples of this approach are [473, 509, 562, 410]; a main source of knowledge are the geometric and task-specifi c properties of objects and events; its usage depends