A Feature Map Approach to Real-Time 3-D Object Pose Estimation from Single 2-D Perspective Views

S. Winkler*, P. Wunsch and G. Hirzinger

Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V. – DLR
Institut für Robotik und Systemdynamik
82230 Weßling

Abstract. A novel approach to the computation of an approximate estimate of spatial object pose from camera images is proposed. The method is based on a neural network that generates pose hypotheses in real time, which can be refined by registration or tracking systems. A modification of *Kohonen's self-organizing feature map* is systematically trained with computer generated object views such that it responds to a preprocessed image with one or more sets of object orientation parameters. The key concepts proposed are representations of spatial orientation that result in continuous distance measures, and the choice of a fixed network topology that is best suited to the representation of 3-D orientation. Experimental results from both simulated and real images demonstrate that a pose estimate within the accuracy requirements can be found in more than 90% of all cases. The current implementation operates at near frame rate on real world images.

1 Introduction

Estimating the three-dimensional position and orientation of known objects from camera images is an important sensory skill of intelligent robots, with applications such as online path or task planning, world model update and visual servoing. However, model-based pose estimation requires to explicitly or implicitly solve the extremely difficult correspondence problem of relating image features to corresponding features in the model description. The difficulty is mainly due to the large number of topologically distinct aspects that arise when opaque 3-D objects are imaged by a camera. Therefore, solutions that solve for correspondences analytically fail to achieve the real-time performance necessary in many applications. On the other hand, there exist efficient feature-based numerical registration algorithms [3, 11] that can iteratively refine an initial estimate of the object's pose. This estimate need not be very accurate, it must only lie *within the range of convergence* of the subsequent numerical procedure. Typically, deviations of about $\pm 25°$ in each rotational degree of freedom and up to 20% of the object size in translation can be compensated for [11].

As it is still very hard to find even an approximate pose solution analytically, a learning approach seems more appropriate. The basic idea is to systematically train a classifier with different object views such that it responds to a pictorial input pattern with the approximate orientation parameters of the presented view. As a large number of sample views may be necessary, training should be done entirely on images that can be automatically generated from a CAD-like object model, rather than requiring an operator to present a large set of images with ground truth to the system. Since an analytical model of the mapping to be learned (i.e. the inverse camera projection) is not available, a neural network-based approach appears most suitable. As spatial translation can usually be determined easily once the object attitude has been found, we will focus on recovering object orientation only.

* now with the Signal Processing Lab of the Swiss Federal Institute of Technology in Lausanne, Switzerland.

2 Previous Research

Most approaches reported in the literature simplify the task by either limiting the range of admissible object orientations, by ignoring self-occlusion, or by assuming that correspondences between image and model features are known. Furthermore, hardly any results on real image data have been reported, most methods have only been validated on computer generated images.

Poggio and Edelman [7] as well as Maggioni and Wirtz [5] both use an ordered vector of object vertex coordinates as input to their respective networks. This input representation poses the difficult correspondence problem of relating vertices of an input image to the vertices of the training set. Furthermore, inevitable self-occlusion is not considered at all.

Park and Cannon [6] explicitly address these problems. Hidden lines are removed in their sample views, and a parametric contour description is used as input. Thus the correspondence problem is reduced to the unique determination of the starting point of the parameterization, which, however, turns out to be quite unrobust in practice.

Kothanzad and Liou [2] as well as Lu et al. [4] avoid such correspondence problems by working with moment invariants derived form a contour representation of the image. However, the extraction of closed object contours from noisy images taken in unfavorable illumination conditions is difficult, and artifacts are likely to occur.

A purely pictorial input representation that does not require any feature extraction is more robust in practice. Ritter [8] uses such a representation for a pose estimation network. The network architecture is a local linear map, an extension of Kohonen's self-organizing feature map, with a cubical topology of only $2\times2\times2$ neurons. The input vector to the network is a heavily subsampled version of the gradient filtered input image. The response of the network are the three angles of rotation about the object's principal axes. Despite the low input resolution and the small network, an average error of 5° is achieved if the range of object rotation is limited to 90° about each axis.

3 Neural Network Design and Pose Representation

In order to make a neural network based solution to the pose estimation problem viable for practical applications, the following requirements should be met:

- An explicit solution of difficult correspondence problems should be avoided. For instance, a feature vector composed of the object's vertices necessitates relating each of the vertices derived from the image at hand with the corresponding vertices in the training set.
- The network should be designed such that it can be trained on computer generated images. As the space of possible object orientation may have to be densely sampled during training, it is quite cumbersome to provide a complete set of real camera images along with the precise orientation parameters.
- Given a single input image, the network should simultaneously provide several pose hypotheses together with some quality measure. These hypotheses can then be either verified or rejected by the subsequent registration procedure.

The first two points favor a pictorial input representation such as the one proposed in [8]. An additional advantage of representing the input by a filtered and subsampled version of the original image is that such a representation exhibits graceful degradation in the presence of noise and varying illumination.

3.1 The Rigid Map

The competitive learning approach usually employed to train feature maps determines a winning node w based on the weight vector $\boldsymbol{w}_w$ that best matches the input pattern $\hat{\boldsymbol{w}}$. The winning node and its neighbors are then adapted to the input according

	Self-Organizing Map	Rigid Map
Training input	weight vector $\hat{\boldsymbol{w}}$ and pose $\hat{\mathbf{p}}$	
Training winner	$\lVert \boldsymbol{w}_w - \hat{\boldsymbol{w}} \rVert \leq \lVert \boldsymbol{w}_i - \hat{\boldsymbol{w}} \rVert \;\; \forall i$	$\delta(\mathbf{p}_w, \hat{\mathbf{p}}) \leq \delta(\mathbf{p}_i, \hat{\mathbf{p}}) \;\; \forall i$
Weight update	$\varDelta \boldsymbol{w}_i = \nu(\mathbf{p}_w, \mathbf{p}_i, t) \cdot \lambda(t) \cdot (\hat{\boldsymbol{w}} - \boldsymbol{w}_i)$	
Pose update	$\varDelta \mathbf{p}_i = \nu'(\mathbf{p}_w, \mathbf{p}_i, t) \cdot \lambda'(t) \cdot \delta(\hat{\mathbf{p}}, \mathbf{p}_i)$	—
Online input	weight vector $\hat{\boldsymbol{w}}$	
Online winner	$\lVert \boldsymbol{w}_w - \hat{\boldsymbol{w}} \rVert \leq \lVert \boldsymbol{w}_i - \hat{\boldsymbol{w}} \rVert \;\; \forall i$	

Table 1. Comparison of the self-organizing map and the rigid map. δ denotes the distance measure between poses, λ is the learning rate. The neighborhood function ν ensures that the nodes are updated in proportion to their distance from the current winner w.

to Kohonen's neighborhood rule (table 1). Depending on the measure defined to compute distances between neighboring nodes, neurons tend to get placed according to the topology of the input space. This property is generally known as *self-organization*.

In the pose estimation problem, each neuron represents a point in an inherently three-dimensional orientation space. For a given parameterization of rotation, the optimal network topology is known in advance. If a uniform distribution of neurons in this space can be determined in advance (see below), self-organization of the network is not necessary for pose estimation, and the training algorithm can be modified such that an a-priori defined topology is preserved during training. This leads to the concept of the *rigid map* [10].

The network topology is chosen such that the neurons are uniformly distributed in rotation space. During training, the winner is *not* selected according to the similarity of input and weight vectors, but based on the proximity of a neuron to the object orientation $\hat{\mathbf{p}}$ presented with the training sample. Only the weight vectors $\boldsymbol{w}_i$ of the winning node and its neighbors are adapted according to Kohonen's rule. This procedure keeps the neuron topology fixed, hence the term rigid map. For pose estimation after training the winner w is of course determined based on the similarity of input and weight vectors. The orientation parameters stored in the best-matching node, $\mathbf{p}_w$, then represent the response of the network. Table 1 summarizes the rules for each type of map.

Unfortunately, Ritter's approach cannot be extended to wider ranges in a straightforward fashion: For ranges greater than $90°$, the network fails to self-organize even if the number of neurons is increased [10]. Two major reasons have been found for this problem:

- Roll-pitch-yaw angles that represent spatial orientation exhibit a variety of discontinuities and ambiguous configurations which violate the implicit continuity assumptions of the learning algorithm.
- A cubical network topology does not reflect global neighborhood relations in 3-D orientation space very well.

With the rigid map in mind, a better suited representation of spatial orientation should therefore meet the following requirements:

- The representation should be unique and not exhibit any discontinuities.
- A distance metric must be available that accurately reflects neighborhood relations in orientation space.
- An algorithm must be available to uniformly distribute nodes in the particular orientation space.

The following two sections introduce such representations together with the pertinent network topologies.

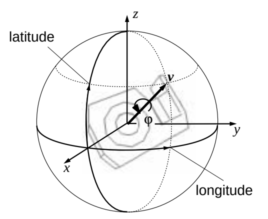

Fig. 1. Extended spherical coordinates consist of a viewing vector $\boldsymbol{v}$ and a rotation φ about $\boldsymbol{v}$.

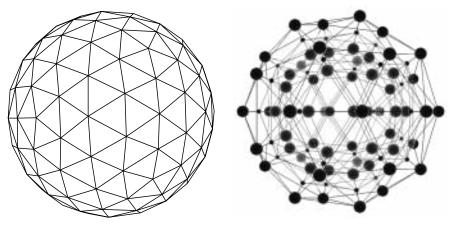

Fig. 2. A uniform node distribution in spherical coordinates can be derived from an 320-patch Gaussian sphere (left). The corresponding node arrangement is shown on the right (only a subset of the nodes is visualized).

3.2 Extended Spherical Coordinates

In order to represent all possible rotations of an object, spherical coordinates with longitude and latitude need to be extended by a third angle describing the rotation of the camera about its own axis, the line of sight (figure 1). Hence, an orientation is represented by a unit vector $\boldsymbol{v}$ and an angle φ. This representation reduces the ambiguities of the roll-pitch-yaw representation to only two points, namely the poles of the sphere. The distance $\delta(\mathbf{p}_1, \mathbf{p}_2)$ between two poses $\mathbf{p}_1 = [\boldsymbol{v}_1, \varphi_1]$ and $\mathbf{p}_2 = [\boldsymbol{v}_2, \varphi_2]$ now comprises two separate parts: the spherical distance along a great arc on the viewing sphere, and the circular distance between the two rotations about the line of sight:

$$\delta(\mathbf{p}_1, \mathbf{p}_2) \;=\; \sqrt{\arccos(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)^2 + (\varphi_1 - \varphi_2)^2} \tag{1}$$

In order to distribute a certain number of nodes as uniformly as possible in orientation space, a uniform tessellation of the unit sphere is needed, which can be obtained from regular polyhedra. In order to achieve a finer resolution, the triangular faces of the icosahedron can be recursively subdivided into four new triangles whose vertices are projected onto the sphere. This leads to the well-known triangulation of the Gaussian sphere, which ensures a quasi-uniform tessellation (cf. figure 2).

In order to allow for a greater flexibility with the number of nodes, the centers of the faces at each level of subdivision can be combined with the vertices. At the lowest level, this is the combination of the icosahedron and the dodecahedron, for example, yielding 32 points, with the nearest neighbors $37.4°$ apart. For a similar quantization of the third dimension describing the rotation φ about the line of sight, 10 nodes (36° apart) are necessary at each of the 32 points, producing a network with a total of 320 nodes.

3.3 Unit Quaternions

Extended spherical coordinates still exhibit ambiguities at the poles of the viewing sphere. However, the set of spatial rotations also fits into the algebraic structure of quaternions [9]. A rotation by the angle φ about the unit vector $\mathbf{u}$ can be represented by a unit quaternion $\mathbf{q} = [\cos\varphi/2,\ \mathbf{u}\sin\varphi/2]$.

The main advantage of unit quaternions over extended spherical coordinates is the removal of ambiguities at the poles. Furthermore, distance calculations are more straightforward, because the three parameters are no longer contained in two separate sets. The only difficulty relevant for our application that remains is an ambiguity between a rotation about the axis $\mathbf{u}$ by the angle φ and a rotation about $-\mathbf{u}$ by $-\varphi$; the corresponding quaternions are antipodes on the 3-sphere. This problem can be circumvented by restricting all quaternions to one hemisphere.

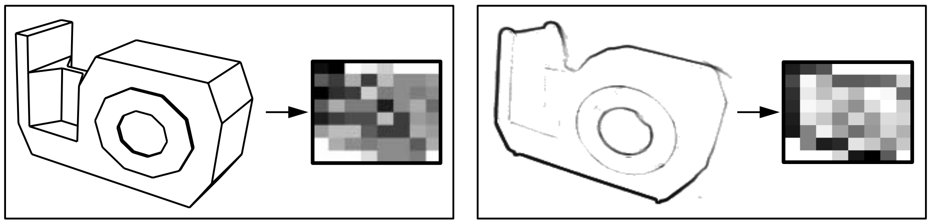

Fig. 3. Left: Training image of a tape dispenser generated from the object's wire-frame model, and the training pattern obtained by subsampling. Right: Sobel filtered camera image and corresponding input pattern. Even though simulated and real image differ significantly due to unfavorable illumination conditions and object colors, a correct pose estimate can be obtained for the view on the right.

The distance $\delta(\mathbf{q}_1, \mathbf{q}_2)$ between two unit quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ is measured as the angle between the two, considering that the distance between antipodes is 0:

$$\delta(\mathbf{q}_1, \mathbf{q}_2) = \min\{\arccos(\mathbf{q}_1 \cdot \mathbf{q}_2), \pi - \arccos(\mathbf{q}_1 \cdot \mathbf{q}_2)\}. \quad (2)$$

Thus the maximum distance between two nodes is $\delta_{max} = \pi/2$ or 90°.

For the quaternion representation, the neurons must be distributed evenly on the unit 3-sphere in 4-space. Since good solutions to the related problem in the three-dimensional case were obtained from regular polyhedra, their n-dimensional equivalents, called regular *polytopes*, can serve the same purpose [1]. Of the six regular polytopes in 4-space, we combine the two with the largest numbers of vertices to obtain a quasi-regular distribution of 720 points on the 3-sphere. Because antipodes represent equivalent orientations, only the points in the upper hemisphere are considered, the other half is ignored. This restriction leaves 360 nodes for the network used in our experiments [10] The minimum distance between neighboring nodes in this arrangement is approximately $15.5°$ in the quaternion measure.

4 Implementation and Experimental Results

Figure 3 shows the results of image preprocessing for both simulated and real images. A training image is generated by projecting a 3-D wire-frame model of the object in a randomly chosen pose. Hidden lines are removed to account for self-occlusion. The resulting image is subsampled within its bounding rectangle to a target resolution of $8{\times}8$ pixels. Although perspective projection is not invariant with respect to spatial translation, the subsampling step eliminates most of its effects, because visibility changes due to translation are rather small within a reasonable working area (see also [10]). Finally the resulting 64-element input vector is normalized to reduce intensity dependence. An input intensity image is Sobel-filtered with a $7{\times}7$ Gaussian derivative kernel and subsampled in the same way as the training images. All preprocessing steps can easily be implemented on standard image processors such as a Datacube MV200, which was used in the experiments reported here.

4.1 Simulation Results

In order to quantitatively compare different pose representations and network topologies, an extensive series of experiments was conducted. Figure 4 summarizes the results. The plots show the error histograms generated from classifying 100,000 randomly generated test views. Our benchmark object is the tape dispenser of figure 3. Training with 50,000 views takes about 15 minutes on a standard SGI Indigo^2 workstation for

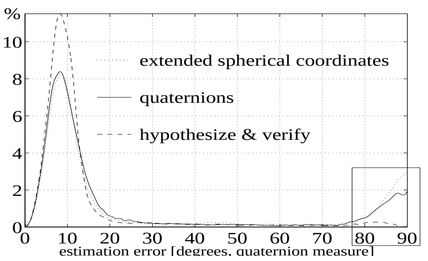

Fig. 4. Histogram of the orientation error for different representations and network topologies. The high maximum corresponds to the sampling interval in orientation space ($\approx 8^\circ$ for networks of this size), the tails near the maximum error of 90° are mainly due to symmetries of the test object.

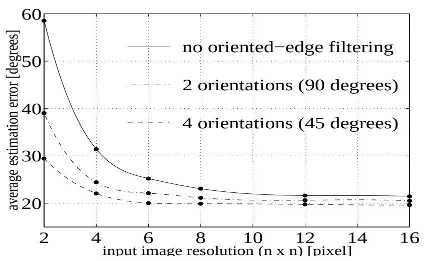

Fig. 5. Average estimation error versus the resolution of the subsampled image (resolution = $n{\times}n$) for 0, 2 and 4 gradient filter orientations. Comparing input vectors of the same dimension (e.g. 1 orientation at 8×8 pixels and 4 orientations at $4{\times}4$), the errors are slightly lower for oriented-edge filtered images.

both network types. The pose estimation error is defined as the unit quaternion distance based on equation 2, converted to degrees. The maximum of each histogram is located at about 8°, which roughly corresponds to half the sampling interval in rotation space. Ideally this should be the only maximum of the curve, but we find significant tails at about 90° of error. There are two major sources for these error tails: The first are symmetries of the object, which cannot be resolved from a single camera image. The second are the poles in the extended spherical coordinate representation. However, both of the proposed network architectures successfully solve the pose estimation task even if the range of admissible orientation is unrestricted.

representation	$e \leq 15^\circ$	$e \leq 25^\circ$
roll-pitch-yaw	20%	24%
spherical coords.	65%	74%
quaternions	73%	81%
3 hypotheses	85%	92%

Table 2. Estimation error quantiles

One conclusion that can be drawn from figure 4 is that the performance of the quaternion-based network is better than that of spherical coordinate network. For reference purposes we have also included the error histogram for the three highest network responses, from which we manually chose the best. The application in mind is a hypothesize-and-verify paradigm, where a small number of pose hypotheses is tested for correctness such that symmetries may be resolved. In this case, the tail in the error distribution nearly disappears, indicating that the errors of the quaternion network are mainly due to the symmetries of the object.

Table 2 gives some cumulative statistics. For the quaternion-based network, the orientation estimation error is less than 25° in 81% of all cases, which is usually accurate enough to ensure convergence of numerical registration procedures [11].

The reduction of the input dimension by subsampling leads to a tremendous loss of information. In particular, the direction of short edges that come to lie within a single subsampling block is lost completely. In order to circumvent this, oriented edge filtering can be employed to emphasize certain angles of the gradient and suppress others by multiplying the gradient image with Gaussian functions centered at the desired angles. This gives a set of image feature vectors, each representing image edges at a certain orientation. The results reported in figure 5 were obtained with oriented-edge filtering at steps of both 90° and 45°, corresponding to 2 or 4 sets of feature vectors, respectively. For a fair assessment, the classification results for equal input dimension should be compared. Two conclusions can be drawn from figure 5: A higher resolution of the subsampled image can increase the estimation accuracy only up to a certain point, and oriented-edge filtering yields only minor improvements for higher resolutions.

Further experiments have demonstrated the robustness of the network with respect to typical image artefacts. Estimation accuracy is degrades gracefully with noise in both edge localization and intensity. Another typical variation with images obtained by perspective projection is the abrupt appearance or disappearance of lines (and surfaces) when the object is translated with respect to the camera, e.g. when images are recorded at varying distances. This is one of the major problems of multi-view representations. Our experiments have shown that a neural network trained with views taken from a fixed distance is robust to such changes of aspect within a reasonable working range.

4.2 Real Data Results

Figure 6 shows the application of the quaternion net to some real images. A bounding rectangle of the object is determined by color segmentation (the object colors being red and light blue). The gradient filtered intensity image within the bounding rectangle is then subsampled, and the resulting 64-element feature vector is normalized and presented to the quaternion map (figure 3). In this example the network consists of 360 nodes and has been entirely trained on synthetically generated views.

In order to visualize the pose estimate, the model wire-frame is placed at the pose relative to the camera that is represented by the winning node. The resulting pose is also passed as an initial pose estimate to the registration algorithm described in [11], which refines the network's estimation. The algorithm extracts edge segments from the raw pixel data and computes the pose by iterative matching of image and model edges. The resulting pose is shown in figure 6 on the right. It takes less than a millisecond to compute the network response on an SGI Indigo^2 workstation, while image preprocessing accounts for about 20 msec on a 384×287 image.

Considering the significant difference between the simulated training data and the test input computed from images (cf. figure 3), the network performance on real images is surprisingly good. The error rates are of course somewhat higher than those obtained with simulated test samples, which were reported in the previous section. As it is quite cumbersome to generate a large real image test set with precise ground truth, we cannot report any statistically significant results here. However, from the experiments conducted, we found that in more than 70% of all cases the quaternion network yielded a pose estimate for our benchmark object that was within the range of convergence of the registration algorithm. The main reason for such a degradation is illumination, as the contrast of an image edge strongly depends on the number and position of light sources. However, as long as most the of the edges in the trained image can also be extracted from the image data, pose estimation usually succeeds.

5 Conclusions and Future Directions

We have presented a feature map approach to the problem of obtaining 3-D pose estimates from single 2-D perspective views. Experiments have shown that a network based on the unit quaternion representation is best suited for this task. The feature map's topology is tailored to this representation of spatial orientation and is kept fixed during training. Training is done entirely on synthetic views generated from a CAD-like 3-D object model, which makes the training phase fully automatic and efficient. Once trained, the network can successfully classify real images. The online computation is very fast, and frame-rate implementations with image processors are easily feasible.

Future research aims at both theoretical and practical aspects. In order to include the effects of illumination, training on synthetic images should only be done to get an initial set of neural weights that are adapted to a given setup by online retraining with real data. An interesting theoretical problem is to adapt the interpolation scheme of Ritter's local linear map to the topology of the quaternion net. This should significantly increase the accuracy of the pose estimate.

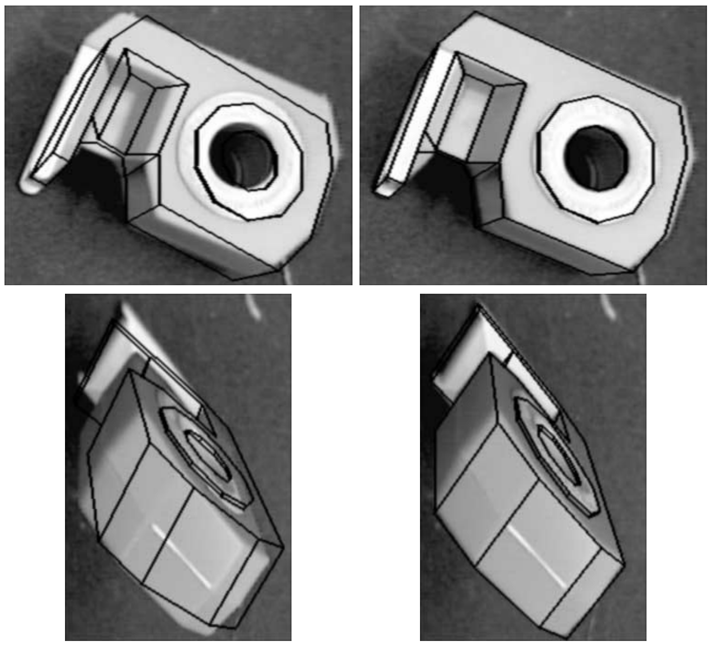

Fig. 6. Left: Real camera images of a tape dispenser with the wire-frame model projected from the view determined by the quaternion network. Image processing and evaluation is confined to the object's bounding rectangle, which is determined by color segmentation. Right: Result of applying the registration method described in [11] to edge segments extracted from the image with the initial pose shown on the left.

References

1. H. S. M. Coxeter. *Regular Polytopes*. Dover, 1973.
2. A. Khotanzad and J. Liou. Recognition and pose estimation of 3-D objects from a single 2-D perspective view by banks of neural networks. In *Proc. of the Artificial Neural Networks in Engineering Conference*, pages 479–484, 1991.
3. D. G. Lowe. Fitting parametrized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.
4. M.-C. Lu, C.-H. Lo, and H.-S. Don. A neural network approach to 3-D object identification and pose estimation. In *Int. Conf. on Artificial Neural Networks*, pages 2600–2605, 1991.
5. C. Maggioni and B. Wirtz. A neural net approach to 3-D pose estimation. In *Proc. International Conf. on Artificial Neural Networks*, pages 75–80, 1991.
6. K. Park and D. J. Cannon. Recognition and localization of a 3-D polyhedral object using a neural network. In *Int. Conf. on Robotics and Automation*, pages 3613–3618, 1996.
7. T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266, 1991.
8. H. J. Ritter. Learning with the self-organizing map. In *Proc. International Conf. on Artificial Neural Networks*, pages 379–384, 1991.
9. K. Shoemake. Animating rotations with quaternion curves. *Computer Graphics*, 19(3):245–254, July 1985.
10. S. Winkler. Model-based pose estimation of 3-D objects from camera images using neural networks. Technical Report IB 515-96-12, Institut für Robotik und Systemdynamik. Deutsche Forschungsanstalt für Luft- und Raumfahrt, 1996. Diplomarbeit. Institut für Nachrichten- und Hochfrequenztechnik. Technische Universität Wien.
11. P. Wunsch and G. Hirzinger. Registration of CAD-models to images by iterative inverse perspective matching. In *Int. Conf. on Pattern Recognition*, pages 78–83, 1996.