Real-Time Pose Estimation of 3-D Objects from Camera Images Using Neural Networks

P. Wunsch, S. Winkler and G. Hirzinger

German Aerospace Research Establishment - DLR
Institute for Robotics and System Dynamics
82230 Wessling, Germany
E-mail: Patrick.Wunsch@dlr.de

Abstract — *This paper deals with the problem of obtaining a rough estimate of three dimensional object position and orientation from a single two dimensional camera image. Such an estimate is required by most 3-D to 2-D registration and tracking methods that can efficiently refine an initial value by numerical optimization to precisely recover 3-D pose. However, the analytic computation of an initial pose guess requires the solution of an extremely complex* correspondence problem *that is due to the large number of topologically distinct aspects that arise when a three dimensional opaque object is imaged by a camera. Hence general analytic methods fail to achieve real-time performance and most tracking and registration systems are initialized interactively or by* ad hoc *heuristics. To overcome these limitations we present a novel method for approximate object pose estimation that is based on a neural net and that can easily be implemented in real-time. 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 idea proposed here is to choose network topology in accordance with 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 81% of all cases. The current implementation operates at 10Hz on real world images and a more advanced use of available image processors is certain to yield results at video frame rate.*

1 Introduction

Estimating the three dimensional position and orientation of known objects from a single or a sequence of 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 analytically solve for correspondences (eg. [3, 7]) fail to achieve real-time performance, which is necessary in most robotic applications.

An approach that seems more promising in terms of computational efficiency is to first compute a rough pose estimate based on fast image classification, which is then refined by efficient numerical registration [12] or tracking algorithms [2]. Under this paradigm the first estimation step is not required to yield optimal accuracy, but rather only needs to ensure that the inital estimate is *within the range of convergence* of the subsequent numerical procedure. Typically, deviations of about $\pm 25^o$ in each rotational degree of freedom and up 20% of the object size in translation can be compensated [12].

As it is still very hard to find even an approximate pose solution in an analytic fashion, a learning approach is 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 (ie. the inverse camera projection) is not available, a neural network-based approach seems most suitable.

2 Previous Work

Although there has been a lot of research on object recognition by neural networks, there exist relatively few neural network-based approaches to the problem of 3-D

pose estimation. As spatial translation can usually be easily determined once the object attitude has been found, most work has focused on recovering object orientation, as we will also do in this contribution. However, most approaches reported in the literature simplify the task by either limiting the range of admissible object orientations, by ignoring self-occlusions 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 [9] propose a feed-forward radial basis function network that uses an ordered vector of object vertices in image coordinates as input. The network is trained with about 100 views generated by uniformly sampling a viewing sphere enclosing the object. The network output are the longitude and latitude on the viewing sphere that correspond to the presented view. Maggioni and Wirtz [6] use the same representation, but pose estimation is done by a self-organizing feature map with cubical topology. Satisfactory estimation accuracy is obtained if the range of presented views is limited to about one eighth of the viewing sphere, but for larger ranges performance quickly degrades. Furthermore the input representation poses the difficult correspondence problem of relating vertices of an input image to the vertices of the training set. In addition, inevitable self-occlusion is not addressed at all.

Park and Cannon [8] explicitly address the last two problems. Occlusions are accounted for by removing hidden lines and surfaces in the sample views, and a parametric contour description represents the pictorial input. Thus the correspondence problems is reduced to the unique determination of the parametrization starting point, which, however, turns out to be quite unrobust in practice.

Kothanzad and Liou [4] totally avoid such a correspondence problem by deriving a set of Zernike moments from the binarized image contour. This representation is not only independent of the way the contour is sampled but is also invariant with respect to planar rotation and translation, which reduces the dimensionality of the pose estimation task. With a two stage network architecture a registration accuracy of better than 3^o is reported for undisturbed computer generated views. Although based on a different network architecture the approach proposed in [5] uses similar contour representations. The advantage of these is that the correspondence problem is avoided and occlusions are implicitly considered. However, the extraction of a closed object contour from noisy images taken in unfavorable illumation conditions is a difficult problem, and artifacts are likely to occur, which is worsened by the fact that moment invariants are very noise sensitve. A merely pictorial input representation that does not require any feature extraction at all should be more robust in practical applications.

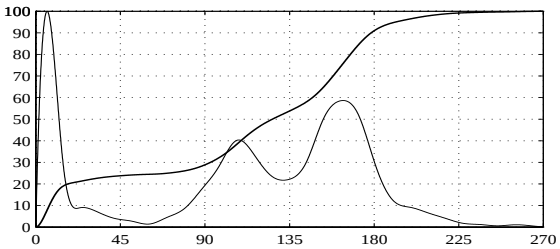

Figure 1. RMS orientation estimation error histogram and cumulative distribution for an implementation of [10]. The network has been trained with object views drawn from a range of 360^o in all rotational degrees of freedom. Only for about 20% of all test samples estimation accuracy is with in tolerable limits. Note that the histogram has been scaled in size.

Ritter [10] 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 $4{\times}4{\times}4$ neurons. The input vector to the network is simply a subsampled version of the input image, that has been gradient filtered in horizontal and vertical direction. The target resolution is $8{\times}8$ resulting in an input vector of 64 elements. The response of the network are the three angles of rotation about the object's principal axes. Despite the low input resolution an average error of only 3^o is achieved if the range of object rotation is limited to 90^o in each degree of freedom.

Unfortunately, Ritter's approach cannot be extended to larger ranges in a straighforward fashion as figure 1 shows. For ranges greater than 90^o the network fails to self-organize even if the number of neurons is increased. Two major reasons have been found for these problems: 1) 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, and 2) a cubical network topology does not very well reflect global neighborhood relations in 3-D orientation space. In the next sections we describe some pose representations and pertinent network topologies that are better suited for pose estimation.

3 Pose Representation and Neural Topology

As has become obvious from our experiments with a roll-pitch-yaw based parametrization of spatial orientation, a representation that avoids discontinuities and ambiguous configurations seems necessary to improve pose estimation performance of a feature map type neural network. The competitve learning approach that is usually employed to train such networks determines a winning node

based on the weight vector that best matches the input pattern. The winning node and all neighboring nodes are then adapted to the input according to Kohonen's neighborhood rule. Depending on the measure defined to compute distances between neighboring nodes, neurons tend to get placed according to the topology of the subspace that they are adapted to. This property is generally known as *self-organization*. As in our case neurons represent samples in the space of three dimensional orientation, it is obvious that a cubical network topology imposed by a Euclidean distance metric, cannot accurately reflect global neighborhood relationships in this space. Therefore a cubical network fails to self-organize if the range of trained object orientations is extended beyond some 90^o, and estimation accuracy quickly degrades.

In the pose estimation problem each neuron represents a point in orientation space, and for a given parametrization of rotation a uniform distribution of neurons in that space – and hence an optimal network topology – is known in advance. Therefore, it turns out that for pose estimation self-organization of the network is not necessary, and we can modify the training algorithm such that an a-priori defined topology is preserved during training. This leads to the concept of a *rigid map* [11]. As each neuron represents a three dimensional rotation, network topology is chosen such that the neurons are uniformly distributed in rotation space. Adaptation during learning also starts out by the determining the winner neuron. However, the winning node is chosen not due to similarity of input and weight vectors, but merely based on the proximity of a neuron to the object orientation presented in the training sample. As before, the winning node and its neighbors are then adapted according to Kohonen's rule. This procedure keeps the neuron topology fixed, hence the term rigid map. For details of the modified learning algorithm refer to [11].

Given the new learning algorithm a suitable representation of spatial orientation must meet the following conditions: 1) The representation must be unique and not exhibit any discontinuites. 2) A distance metric must be available that accurately reflects neighborhood relations in orientation space, and 3) an algorithm must be available to uniformly distribute nodes in the particular orientation space. The following two sections introduce such distance measures with the pertinent network topologies.

3.1 Extended Spherical Coordinates and 2-1-Spherical Topology

In order to be able to represent all possible rotations of an object, spherical coordinates (cf. figure 2) 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. This representation reduces the ambiguities of the roll-pitch-yaw representation to only two points,

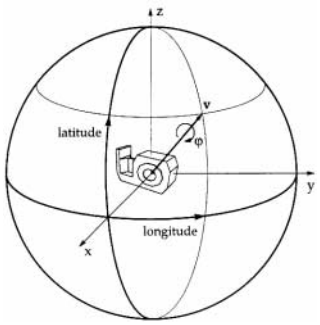

Figure 2. Extended spherical coordinates consist of a viewing vector $\boldsymbol{\nu}$ and a rotation angle ϕ about $\boldsymbol{\nu}$.

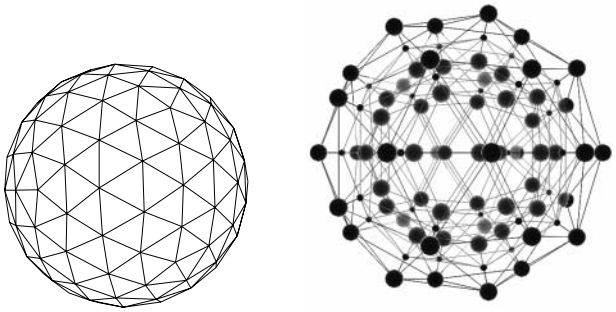

Figure 3. A uniform node distribution in spherical coordinated is derived from a 320-patch Gaussian sphere (left). The corresponding network topology is shown on the right. The size of a node is proportional to ϕ, while the node location is derived from the Gaussian sphere and represents the vector part $\boldsymbol{\nu}$. Note that only a subset of the nodes is visualized.

namely the poles of the sphere.

An orientation is hence represented by a unit vector $\boldsymbol{\nu}$ and an angle ϕ. The distance $\delta(\mathbf{p}_1, \mathbf{p}_2)$ between two poses $p_1 = [\boldsymbol{\nu}_1, \phi_1]$ and $p_2 = [\boldsymbol{\nu}_2, \phi_2]$ 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.

$$\delta(\mathbf{p}_1, \mathbf{p}_2) \;=\; \sqrt{\delta_S(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2)^2 + \delta_C(\phi_1, \phi_2)^2} \qquad (1)$$

where

$$\delta_S(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2) = acos(\boldsymbol{\nu}_1 \cdot \boldsymbol{\nu}_2)$$

is the spherical distance along the great arc between the two poses $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$, while

$$\delta_C(\phi_1, \phi_2) = \phi_1 - \phi_2$$

is the circular distance along the viewing axis.

In order to distribute a certain number of nodes as uniformly as possible in orientation space, a uniform tesselation of the unit sphere is needed, which can be obtained by inscribing a regular polyhedron. However, the highest-order regular polyhedron is the ikosahedron, which has only 20 vertices. In order to achieve finer resolution the triangular faces of the icosahedron are recursively subdivided into four new triangles whose vertecies are projected onto the sphere. This leads to the well-known triangulation of the Gaussian sphere, which ensures a quasi-uniform tesselation. Figure 3 shows on the left a 320-patch sphere which is the basis for the distribution of neurons in extended spherical coordinate space. Thus one neuron represents a patch of about $\pm 8^o$ in both longitude and latitude, which also imposes a limit on the pose estimation accuracy that can be expected.

Based on these representations training is done in the following way: Starting from a tesselation of the Gaussian sphere at a desired resolution, a set of neurons that represent the scalar part ϕ of the extended spherical coordinates is placed a the center of each patch[1] of the Gaussian sphere (figure 3 right). Then, for each training sample, the node that is closest to the orientation in the presented view is selected and the corresponding weight vector and its neighbors are updated according to Kohonen's rule. For online pose estimation, an input pattern is presented to the network and the node with the greatest response indicates the orientation of the imaged object. This recall procedure resembles the well-known method of geometric hashing, however neighborhood updating guarantees a continuous mapping unlike hash tables that are usually discrete.

3.2 Unit Quaternions and 3-Hemispherical Topology

Extended spherical coordinates still exhibit ambiguities at the poles, which can cause problems (cf. section 4.2). The set of all possible rotations also fits into the algebraic structure of quaternions. A rotation by the angle ϕ about the unit vector $\mathbf{u}$ can be represented by a unit quaternion

$$\mathbf{q} = [\cos\frac{\phi}{2}, \mathbf{u}\sin\frac{\phi}{2}].$$

The main advantage of unit quaternions over the extended spherical representation 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 $-\phi$. The corresponding quaternions are antipodes on the 3-sphere. This problem can be circumvented by restricting all quaternions to the positive hemisphere.

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 bewteen the two, considering that the distance between antipodes is 0.

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

Thus the maximun distance is $\delta_{max} = \pi/2$.

For the quaternion representation, neurons must be distributed evenly on the three dimensional unit 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, although resulting neuron arrangements can hardly be visualized.

Any $n+1$ points which do not lie in an $(n-1)$-space are the vertices of an n-dimensional simplex, the simplest polytope. When its vertices are mutually equidistant, the simplex is said to be regular [1] . Of the six regular polytopes in 4-space, the one with 120 vertices and 600 tetrahedral cells as well as its reciprocal with 600 vertices and 120 dodecahedral cells are the most promising for distributing larger numbers of points on the 3-sphere. Even more points could be obtained by subdividing the regular polytopes with an algorithm similar to the triangulation of the Gaussian sphere. Because antipodes represent equivalent orientations, only the points in the upper hemisphere are considered, the other half is ignored. Vertices on the 2-spherical boundary of the 3-hemisphere that happen to have antipodes are only considered once. Considering these restrictions, a combination of the vertices of the two regular polytopes mentioned above leads to a network with 360 nodes, which was used in our experiments [11].

4 Experimental Results

4.1 Image Preprocessing

Figure 4 shows the results of image preprocessing for both simulated and real images. A training image is generated by projecting a 3-D wireframe model of the object from a randomly chosen viewpoint. Hidden lines are removed to account for self occlusion. The resulting image is subsampled within its bounding rectangle to a target resolution of 8×8 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 [11]). Finally the resulting 64-element input vector is normalized to reduce intensity dependance. For a real camera image we first compute the bounding rectangle of the imaged object based on color segmentation. The intensity image is then Sobel-filtered

[1] to reduce dimensionality only a subset of these may be used.

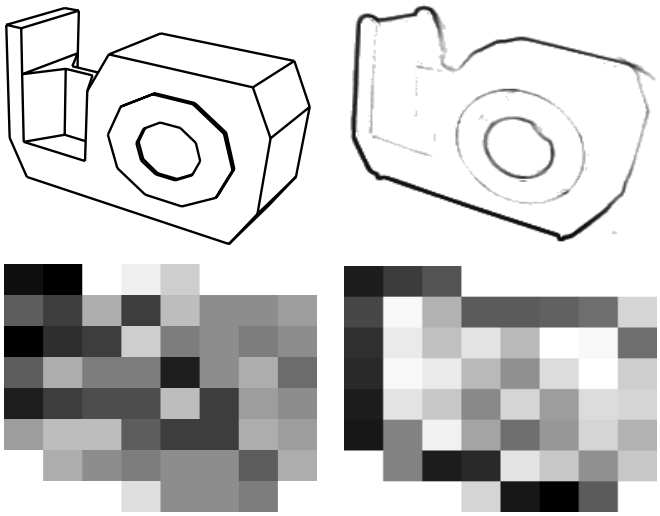

Figure 4. Left: Training image of a tape dispenser generated from the objects's wireframe model (top) and training pattern obtained by subsampling (bottom). Right: Sobel filtered camera image (top) and corresponding input pattern (bottom).

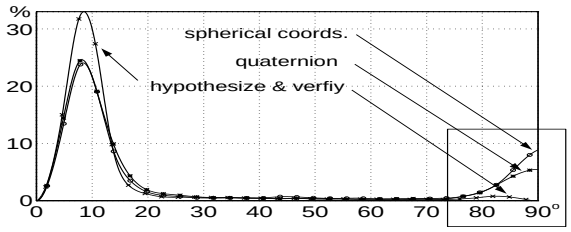

Figure 5. Histogram of the orientation error for different representations and network topologies. The high maximum corresponds to the sample interval in orientation space ($\approx 8^o$), the others are due to symmetries and singularities in representation.

Table 1. Estimation error quantiles

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

with a 7x7 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.

As can be seen in figure 4, due to illumination quite a few edges that are visible in the simulated image exhibit no contrast in the real image and the resulting input patterns seem quite different to a human observer. However, up to a certain degree the network is robust with respect to such distortions. The view presented in figure 4 is, for instance, correctly classified.

4.2 Simulation Results

In order to quantitatively compare different pose representations and network topologies an extensive series of simulations were conducted. Figure 5 summarizes the results. The plots show the error histograms generated from classifing 10,000 randomly generated test views. Training took about 15 minutes on a standard Sillicon Graphics INDY workstation for both network types, each requiring about 400KByte of storage. 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 a about 8^o which roughly corresponds to the sampling interval in rotation space. Ideally this should be the only maximum of the curve, but in practice we find significant tails at about 90^o of error. There are two major sources for these error tails. The first are symmetries of the object (here an ordinary tape dispenser), which cannot be resolved from a single camera image. The second are the ambiguities in pose represenation. An important conclusion that can be drawn from figure 5 is that the performance of the quaternion-based network is notably better than that of spherical coordinate network. Compared to the results obtained with a roll-pitch-yaw based cubical network (figure 1) both network architectures exhibit substantial improvements.

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 unexpexted errors of the quaternion network are mainly due to symmetries.

Table 1 gives some cumulative statistics. For the quaternion-based network in 81% of all cases the orientation estimation error was less than 25^o which is usually accurate enough to ensure convergence of numerical registration procedures [12].

4.3 Real Data Results

Figure 6 shows the application of the quaternion net to some real images taken from a camera mounted on a robot gripper. Note that the network has been entirely trained

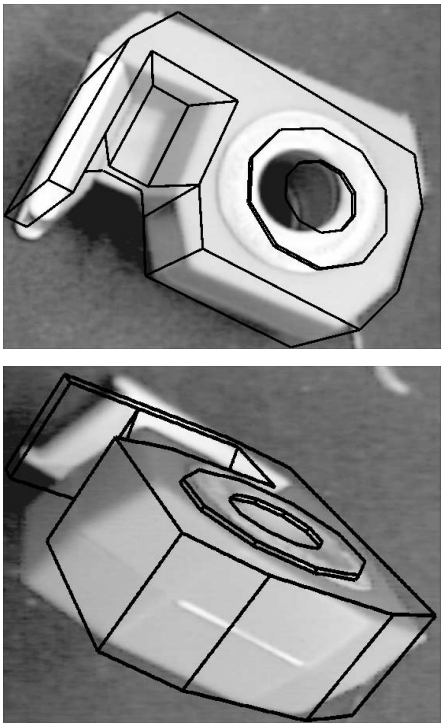

Figure 6. Real camera images of a tape dispenser with the wireframe model projected from the view determined by the quaternion network. In both cases the estimate is within the range of convergence of the registration method in [12].

on simulated images. Nevertheless performance is surprisingly good, although the error rates are somewhat higher than those achieved with simulated data. The main reason for such a degradation is illumination. As pointed out before, the contrast of an image edge strongly depends on illumination conditions. As long as most the of the edges in the trained image can be extracted from the image data, pose estimation will usually succeed. However, to further improve performance training from simulated images should only be considered as a bootstrapping phase, after which the network should be adapted online to match the illumination conditions at hand.

5 Conclusions and Future Directions

We have presented a neural network based approach to obtain a rough 3-D pose estimate from a 2-D camera image. Training is done entirely on synthetic views that are generated from a 3-D CAD-like object model. This makes the training phase fully automatic and efficent. The network architecture is of feature map type with a neuron topology that is especially tailored to the representation of spatial orientation. Experiments have shown that a network based on the unit quaternion representation is best suited for pose estimation. The online computation time is small and real-time implementations with image processors are easily feasible. Once trained, the network can also successfully classify real images.

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 devise an interpolation scheme similar to that of the local linear map that is suited to the topology of the quaternion net. This should significantly increase the accuracy of the pose estimate.

References

[1] H. S. M. Coxeter. *Regular Polytopes*. Dover, 1973.

[2] E. D. Dickmanns and V. Graefe. Dynamic monocular machine vision. *Machine Vision and Applications*, 1:223–240, 1988.

[3] P. J. Flynn and A. K. Jain. BONSAI: 3-D object recognition using constrained search. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(10):1066–1075, 1991.

[4] 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 *American Society of Mechanical Engineers*, pages 479–484, 1991.

[5] M.-C. Lu, C.-H. Lo, and H.-S. Don. A neural network approach to 3-D object identification and pose estimation. In *Proc. International Conf. on Artificial Neural Networks*, pages 2600–2605, 1991.

[6] 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.

[7] O. Munkelt. Aspect-tree: Generation and interpretation. *Computer Vision and Image Understanding*, 61(3):265–386, 1995.

[8] K. Park and D. J. Cannon. Recognition and localization of a 3-D polyhedral object using a neural network. In *Proc. IEEE International Conf. on Robotics and Automation*, pages 3613–3618, 1996.

[9] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266, 1991.

[10] H. J. Ritter. Learning with the self-organizing map. In *Proc. International Conf. on Artificial Neural Networks*, pages 379–384, 1991.

[11] S. Winkler. Model-based pose estimation of 3-D objects from camera images by using neural networks. Technical Report 515-96-12, Institute of Robotics and System Dynamics. German Aerospace Research Establishment – DLR, 1996. Master's Thesis. TU Wien, Austria.

[12] P. Wunsch and G. Hirzinger. Registration of CAD-models to images by iterative inverse perspective matching. In *Proc. International Conf. on Pattern Recognition*, pages 78–83, 1996.