A Feature Map Approach to Pose Estimation Based on Quaternions

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

German Aerospace Research Establishment – DLR
Institute of Robotics and System Dynamics
82230 Wessling, Germany

Abstract. This paper proposes a novel solution to the problem of pose estimation of three-dimensional objects using feature maps. Our approach relies on *quaternions* as the mathematical representation of object orientation. We introduce the *rigid map*, which is derived from Kohonen's self-organizing feature map. Its topology is fixed and chosen in accordance with the quaternion representation. The map is trained with computer-generated object views such that it responds to a preprocessed input image with one or more sets of object orientation parameters. Experimental results demonstrate that a pose estimate within the accuracy requirements can be found in more than 90% of all cases. Our current implementation operates at near frame rate on real input images.

1 Introduction

Determining the three-dimensional position and orientation of objects is an essential 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 or tracking algorithms [4, 12] that are able to iteratively refine an initial estimate of the object's pose to a high precision. This initial pose estimate need not be very accurate, it must only lie *within the range of convergence* of the subsequent registration procedure. Typically, deviations of up to $\pm 25^\circ$ in each rotational degree of freedom and 20% of the object size in translation can be compensated for [12].

2 Feature Map Approach

As it is still very hard to determine even the approximate pose analytically in a reasonable amount of time, a learning approach seems more appropriate. There has been some previous work on the subject [3, 5–8]. However, the approaches reported in these papers simplify the task by either siginificantly limiting the range of admissible object orientations, by ignoring self-occlusions, or by assuming that correspondences between image and model features are known. Furthermore, none of the presented methods have been validated on real input images.

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

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

- The difficult correspondence problem 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 it is awkward to provide a large set of different camera views along with the precise orientation parameters.

These requirements favor a purely pictorial input representation that does not require sophisticated feature extraction. Such a representation also exhibits graceful degradation in the presence of noise and varying illumination. The basic idea is to systematically train a neural network with different object views such that it responds to a pictorial input pattern with the approximate orientation parameters of the presented view. Ritter [9] uses such a representation for a pose estimation network in order to demonstrate his local linear map architecture. The response of the network are the object's roll-pitch-yaw angles of rotation. Despite the low input resolution and the small network, an average estimation error of 5° is achieved if the range of rotation is limited to 90° about each axis.

3 Quaternion Representation of Orientation

Unfortunately, Ritter's approach cannot be extended to wider ranges in a straightforward fashion [11]. For ranges greater than 90°, the network fails to self-organize even if the number of neurons is increased, and the estimation accuracy quickly degrades. Two major reasons have been found for this problem:

- The roll-pitch-yaw angle representation of spatial orientation exhibits 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 orientation space very well, i.e. there exists no natural distance metric between different poses.

In order to alleviate these problems, we conducted experiments with a variety of possible representations of orientation, including orthonormal rotation matrices, roll-pitch-yaw angles, extended spherical coordinates, and quaternions [11]. Of these, the quaternion representation of orientation proved to be the most suitable. Quaternions [2] are an extension of complex numbers to four dimensions, usually written as $a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$. Multiplication is governed by Hamilton's fundamental rule $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1$. In a more compact notation, a quaternion can be regarded also as a combination of the scalar a and the vector $\mathbf{v} = [b, c, d]$: $\boldsymbol{q} = [a, \mathbf{v}]$, which simplifies formulas dramatically.

The set of all possible orientations fits into this algebraic structure of quaternions [10]: A rotation by the angle ϕ about the unit vector $\mathbf{u}$ can be represented by the unit quaternion $\boldsymbol{q} = [\cos\phi/2, \mathbf{u}\sin\phi/2]$. Performing successive rotations corresponds to multiplying unit quaternions.

Naturally, these unit quaternions come to lie on the unit 3-sphere in 4-space (i.e. the surface of the unit sphere in four-dimensional space). The only ambiguity occurs between a rotation about $\mathbf{u}$ by ϕ and a rotation about $-\mathbf{u}$ by $-\phi$; the corresponding quaternions are antipodes on the unit 3-sphere. This problem can be circumvented by restricting all quaternions to one 3-hemisphere. Then the distance $\delta(\boldsymbol{q}_1, \boldsymbol{q}_2)$ between two unit quaternions (two poses) $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ becomes

$$\delta(\boldsymbol{q}_1, \boldsymbol{q}_2) = \min\{\arccos(\boldsymbol{q}_1 \cdot \boldsymbol{q}_2), \pi - \arccos(\boldsymbol{q}_1 \cdot \boldsymbol{q}_2)\}. \quad (1)$$

Thus, the maximum distance in quaternion measure is $\pi/2$ or 90°.

For using the quaternion representation with the rigid map (see below), the neurons should be distributed as uniformly as possible on the unit 3-hemisphere in 4-space. Since good solutions to the related problem in the three-dimensional case were obtained from regular polyhedra [11], whose vertices are distributed uniformly on the surface of the enclosing sphere, their n-dimensional equivalents, called regular *polytopes* [1], can serve the same purpose. Of the six regular polytopes in 4-space, we use the one with the most vertices (600) as the basis for distributing larger numbers of points on the 3-sphere. Because antipodes represent equivalent orientations, only the points in one 3-hemisphere are considered. Combining the above polytope with its reciprocal under this restriction leaves 360 nodes for the network used in our experiments.

4 The Rigid Map

The competitive learning approach usually employed to train feature maps determines a winning node w based on the weight vector $\mathbf{w}_w$ that best matches the input pattern $\hat{\mathbf{w}}$. The winning node and its neighbors are then adapted to the input according to Kohonen's neighborhood rule (see table 1). Depending on the metric 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 this space can be determined (cf. above), 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* [11]. 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 $\hat{\boldsymbol{p}}$ presented with the training sample. Only the weight vectors $\mathbf{w}_i$ of the winning node and its neighbors are then adapted according to Kohonen's rule. This procedure

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

Table 1. Comparison of Self-Organizing Map and Rigid Map. δ denotes the distance metric 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.

keeps the neuron topology fixed, hence the term rigid map. After training, i.e. for online pose estimation, 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, $\boldsymbol{p}_w$, then represent the response of the network. Table 1 summarizes the rules for both map types.

5 Experimental Results

As a large number of sample views may be necessary, training is done on images that are generated automatically from CAD-like object models, rather than requiring an operator to present a large set of images with ground truth to the system. A training image is generated by projecting a 3-D wire-frame model of the object in a random pose. Hidden lines are removed to account for self-occlusion. The resulting image is subsampled within its bounding rectangle to a much lower target resolution, for instance 8×8 pixels. Finally, the resulting pattern is normalized to reduce intensity dependence (figure 1).

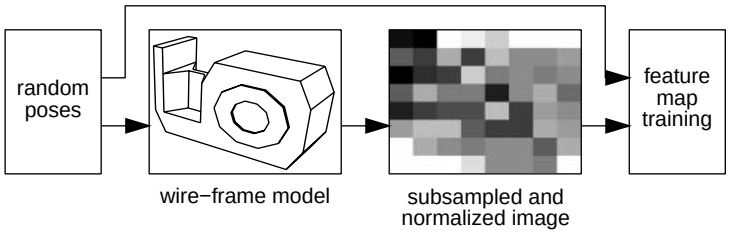

Fig. 1. Training of the feature map: Projections of a 3-D wire-frame model of the object in random poses are subsampled and normalized to form the input to the network. The pose information is also used directly for supervised learning.

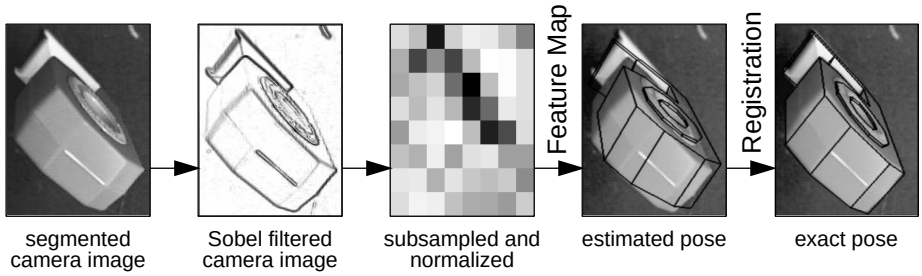

Fig. 2. The feature map at work: The segmented camera image is Sobel filtered, subsampled and normalized. This yields the input for the feature map, which estimates the object's pose (shown as a wire-frame model overlay). The exact pose can then be determined by applying a registration method to edge segments.

Figure 2 shows the application phase of the quaternion net. Note that the network has been trained entirely with synthetically generated images. After rectification of the camera image, a bounding rectangle of the object is determined by color segmentation. In analogy to the training phase, the intensity image within the bounding rectangle is Sobel-filtered with a 7×7 Gaussian derivative kernel and then subsampled and normalized just like the training images. These preprocessing steps can be implemented easily on standard image processing hardware such as a Datacube MV200, where processing a 384×287 image takes

about 20 ms. Subsequent computation of the pose estimate by the neural network takes less than a millisecond on a standard workstation. The resulting pose can then be passed on to the registration algorithm described in [12] as an initial pose estimate. The algorithm extracts edge segments from the raw pixel data and refines the map's pose estimate by iterative matching of image and model edges. Hence, the exact pose is obtained in a fully automatic way.

In order to quantitatively compare different pose representations and network topologies, an extensive series of simulations was conducted [11]. Figure 5 summarizes the most important results for the quaternion map, which was trained with 50,000 views in about 15 minutes on a standard SGI Indigo^2 workstation. The plots show the error histograms generated from classifying 100,000 randomly generated test views of the tape dispenser shown in the previous figures. That particular object is quite demanding in terms of pose estimation as it is non-convex with few symmetries and exhibits curved features (approximated with polygons in the object model).

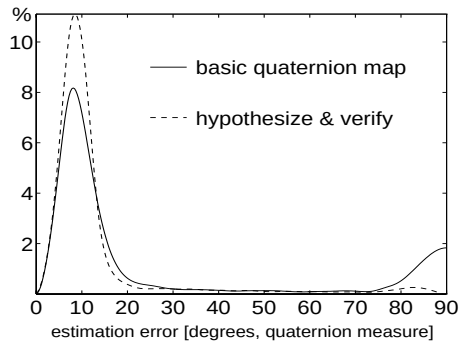

Fig. 3. Histograms of the network's pose estimation error for the tape dispenser. The tail of the histogram is due to object symmetries and disappears for three hypotheses.

The maximum of the histograms at about 8° corresponds to the sampling interval of orientation space. Ideally this should be the only maximum of the curve, but in practice we find a "tail" near the maximum 90° of error, which disappears when testing a small number of the network's best pose hypotheses and choosing the correct response (the application in mind being the hypothesize & verify paradigm). This indicates that the major source for the error tail are symmetries of the object that cannot be resolved from a single camera image. Depending on the object, the orientation estimation error is less than 25° in 80% to 90% of all cases, which is already accurate enough to ensure convergence of numerical registration methods [12]. These numbers can be improved even further when two or three hypotheses are tested, as shown in figure 5.

The rigorous reduction of the input dimension by subsampling inevitably leads to a tremendous loss of information. In particular, the direction of edges within a single subsampling block is lost almost completely. In order to circumvent this problem, oriented-edge filtering was used before subsampling: The gradient of the image was multiplied by Gaussian functions centered at the desired angles, giving a set of input images, each representing image edges at a certain orientation [11]. However, we found that this procedure yields only minor improvements. Similarly, increasing the resolution beyond 8×8 pixels only marginally improves accuracy. On the other hand, decreasing the resolution of the subsampled image to below 6×6 pixels leads to a significant deterioration of estimation accuracy.

Further experiments have demonstrated the robustness of the network with respect to typical image artifacts. Estimation accuracy 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, for example, 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 relatively robust to these changes of aspect [11].

Considering the significant difference in quality of the simulated training data (the wire-frame model in figure 1) and the input computed from camera images (the Sobel filtered image in figure 2), the network performance on real images is surprisingly good. Varying illumination conditions represent the main problem, because they strongly influence edge contrast. In this respect, the object model is quite crude and could be made more realistic. 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.

6 Conclusions

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

References

1. H. S. M. Coxeter: *Regular Polytopes*. Dover, 1973.
2. W. R. Hamilton: *Elements of Quaternions*. Chelsea, 1969.
3. A. Khotanzad, J. Liou: "Recognition and pose estimation of 3-D objects from a single 2-D perspective view by banks of neural networks." in *Proc. Artifical Neural Networks in Engineering Conference*, pp. 479–484, ASME Press, 1991.
4. D. G. Lowe: "Fitting parametrized three-dimensional models to images." in *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 60, no. 5, pp. 441–450, May 1991.
5. M.-C. Lu, C.-H. Lo, H.-S. Don: "A neural network approach to 3-D object identification and pose estimation." in *Proc. Int. Conf. on Artificial Neural Networks*, pp. 2600–2605, 1991.
6. C. Maggioni, B. Wirtz: "A neural net approach to 3-D pose estimation." in *Proc. Int. Conf. on Artificial Neural Networks*, pp. 75–80, 1991.
7. K. Park, D. J. Cannon: "Recognition and localization of a 3-D polyhedral object using a neural network." in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3613–3618, 1996.
8. T. Poggio, S. Edelman: "A network that learns to recognize three-dimensional objects." in *Nature*, vol. 343, pp. 263–266, 1991.
9. H. J. Ritter: "Learning with the self-organizing map." in *Proc. Int. Conf. on Artificial Neural Networks*, pp. 379–384, 1991.
10. K. Shoemake: "Animating rotation with quaternion curves." in *Computer Graphics*, vol. 19, no. 3, pp. 245–254, July 1985.
11. S. Winkler: *Model-Based Pose Estimation of 3-D Objects from Camera Images Using Neural Networks*. Technical Report 515-96-12, German Aerospace Research Establishment – DLR. Master's Thesis, TU Vienna, Austria, 1996.
12. P. Wunsch, G. Hirzinger: "Registration of CAD-models to images by iterative inverse perspective matching." in *Proc. Int. Conf. on Pattern Recognition*, pp. 78–83, 1996.