Pose Estimation From Image Data Without Explicit Object Models

G. Dudek* and **C. Zhang**
Centre for Intelligent Machines
McGill University
Montreal, Que, Canada H3A 2A7
email: {dudek,zhangc}@cim.mcgill.edu

Abstract

We consider the problem of locating a robot in an initially-unfamiliar environment from visual input. The robot is not given a map of the environment, but it does have access to a limited set of training examples each of which specifies the video image observed when the robot is at a particular location and orientation. Such data might be acquired using dead reckoning the first time the robot entered an unfamiliar region (using some simple mechanism such as sonar to avoid collisions). In this paper, we address a specific variant of this problem for experimental and expository purposes: how to estimate a robot's orientation (pan and tilt) from sensor data.

Performing the requisite scene reconstruction needed to construct a metric map of the environment using only video images is difficult. We avoid this by using an approach in which the robot learns to convert a set of image measurements into a representation of its pose (position and orientation). This provides a *local* metric description of the robots relationship to a portion of a larger environment. A large-scale map might then be constructed from a collection of such local maps. These maps express the statistical relationship between the image measurements and camera pose. The conversion from visual data to camera pose is implemented using multi-layer neural network that is trained using backpropagation. For extended environments, a separate network can be trained for each local region. The experimental data reported in this paper for orientation information (pan and tilt) suggests the accuracy of the technique is good while the on-line computational cost is very low.

1 Introduction

In many cases, a complete and accurate map of an environment is a useful tool for navigating within it. In general, however, it is difficult to acquire and maintain a complete metric map of the entire environment of an active agent such as a person or a robot. In many cases *a priori* maps are unavailable and they are almost never complete. Most simple devices for measuring position and distance are relative measurement tools (e.g. odometers). Furthermore, unavoidable errors in estimates of orientation, distance accumulate with successive motions and make the general problem of maintaining an accurate absolute coordinate system very difficult (Dav86).

In order to avoid the problems of maintaining a globally accurate metric map of the world, we suggest mapping the world as a connected set of *local* regions for which metric spatial information is available. In some instances it may be possible to produce a map which is completely covered by such "well-mapped" regions. In other environments, there may be substantial portions of the environment for which only partial or qualitative metric information is available. It is often sufficient to model an environment in terms of familiar local regions and the interconnections between them (KB87; DJMW91; BD90; Har87). If the robot is able to sense its location accurately within a familiar region, it can

*Gregory Dudek is also faculty with the School of Computer Science. We gratefully acknowledge the financial support of NSERC (through the operating grants programme). The Xerion simulator developed at the University of Toronto was used for neural network simulation. We are grateful to Geoffrey Hinton for his comments on the network architecture, his suggestion re. online learning and some portions of the presentation.

compensate for the cumulative errors that result from uncertainties in its movements as it travels within and between regions. For example, in an environment composed of tiled offices linked by carpeted hallways, it may be impossible to accurately measure distances along the hallway due to slippage of the a robots wheels. Nevertheless, it may be possible to navigate between offices using simple control strategies such as wall following. Thus, the offices might permit locally accurate spatial estimation within each one independently while the hallway linking them would have to be represented only at a topological or graph-theoretic level (see figure 1).

Even where regions with local metric structure are of little utility in and of themselves, they can be or great utility for qualitative exploration. Each one can provide positional re-calibration that would allow a robot using simple control strategies to avoid straying too far from a desired path. Kuipers referred to such re-calibration in general terms as rehearsal procedures that would move a robot to a reference position from a nearby starting point (and suggested they could be accomplished by local functional minimization, similar to perceptual serving) (KB87). Other work on graph mapping pre-supposes that from specific locations in the world, appropriate selections of which direction or doorway or path to move along can be determined (DJMW91; BD90). The ability to locally define a metric coordinate space is a somewhat more general solution to this problem of finding reference points.

In this paper we assumed that the robot knows which local region it is in and we focus on the problem of converting the current visual input into an estimate of the local pose (position and orientation). We ignore the issue of how a large environment is divided into local regions, though there is relevant recent work on neural networks that automatically decompose complicated functions into a number of simpler functions (JJNH91). We concentrate, rather, on the problem of using visual input data to recover the pose of the camera within a (local) environment that has been previously visited. This type of approach has been proven useful in the context of position from (very noisy) sonar data[1] and is somewhat related to work of model-based position estimation from image data (BR93; KMK93). Unlike model-based methods, however, it does not depend on the selection of specific robust geometric features or a good initial estimate of the camera's position.

The mobile robot which is used can carry a camera mounted on a two degree-of-freedom pan-tilt unit as shown in figure 2. Since it is laborious to collect data that is accurately labelled with position, results are reported for data collected by changing pan-tilt positions only, without any robot's movement. This allows more accurate estimates of the methods actual performance. The same technique can be also applied to estimating translational parameters, and that the results described here generalize to this closely related problem. Preliminary data for Cartesian position (translation) is promising but its analysis is more involved and is not considered here.

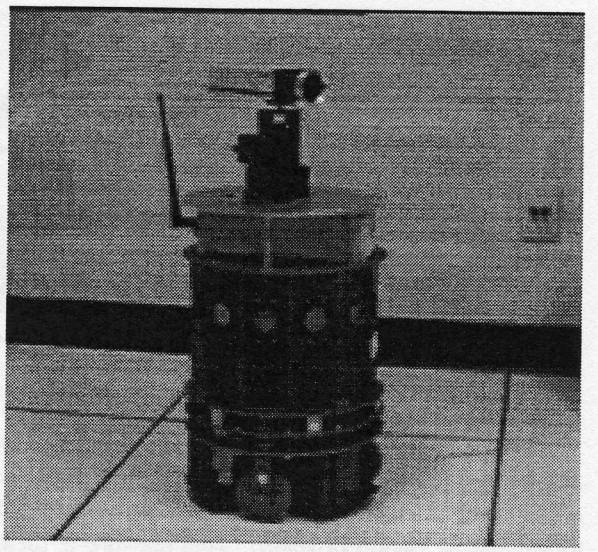

Figure 2: The mobile robot carrying a camera mounted on a two degree-of-freedom pan-tilt unit.

2 Converting images directly into locations

Correctly recovering the environmental structure using only image data is well known to be an extremely difficult and computationally costly problem. Instead, our approach extracts the robot's location from the raw measurements without using any explicit model of the locations of surfaces in the environment. The *"perceptual structure"* (as opposed to physical 3-D structure) of a local region is recorded by statistically encoding image properties as a function of position. As presented here, we accomplish this with visual input data acquired at fixed robot positions. This is performed by computing a collection of statistical descriptors from

[1] This work was carried out with the first author and G. Hinton.

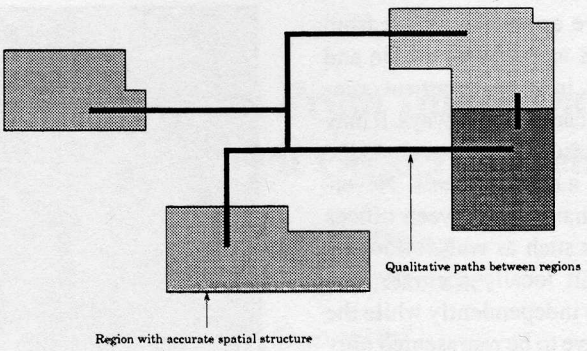

Figure 1: Local metric sub-maps linked by qualitative paths

images, i.e. global features. This collection of features is used to train an artificial neural network specialized to the current region in space. This network is then used to determine the orientation of the camera from an image *given that the robot is within the correct region of the environment*. Thus for an input feature vector $\vec{d}$ a network output

$$\vec{N}(\vec{d}) = (\theta, \phi) \qquad (1)$$

can be determined, where θ and ϕ are the pan and tilt angles of the camera. The consequence is that orientation information can be recovered without the use of explicit models that account for the solution of the scene reconstruction problem. In addition, very limited on-board computation is need to produce the position of the camera using a previously trained neural network. (Although training itself may involve substantial computation cost this can be performed off-line.)

2.1 Perceptual structure

Images were input to the network encoded as a small collection of global statistical measurements. The measurement features were derived from edge images (computed using a Canny's edge operator (Can86)) to minimize the effects of luminance variations. Global statistical descriptors were used to minimize the dependence of the algorithm on either any specific object in the scene or any specific assumptions about image or scene structure; The perceptual structure associated with a position in space hence consists of the following measurements:

- Edge densities in the whole image and selected sub-regions of the image;
- Centroid of the edge distribution;
- Standard deviation of edge distributions at several scales;
- Mean edge orientations in the whole image and selected sub-regions;
- Densities of lines at several orientations (sampling orientation space).

where the selected sub-regions are referred to the areas shown in the figure 4. There is little doubt that some of these features are more useful than others. The questions of what features are most appropriate are currently being examined. Note that using an excessive number of such features (or too many hidden units in the network) has undesirable consequence for the number of weights within the network as well as for the number of training example required (or, in general, the Vapnik-Chernovenkis dimensionality of the problem (VC71; BEHW86)).

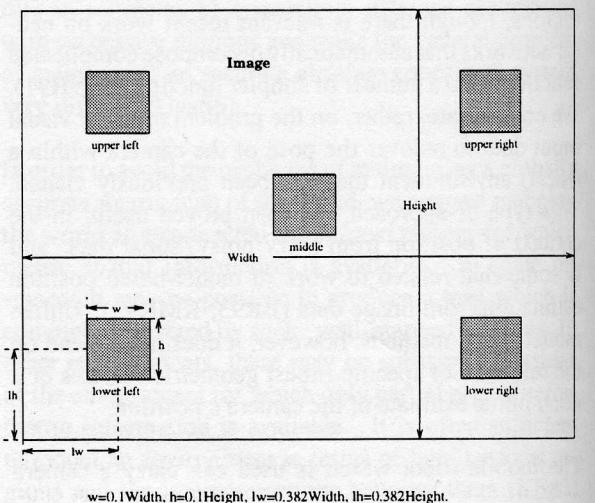

Figure 4: Sub-windows used in feature extraction.

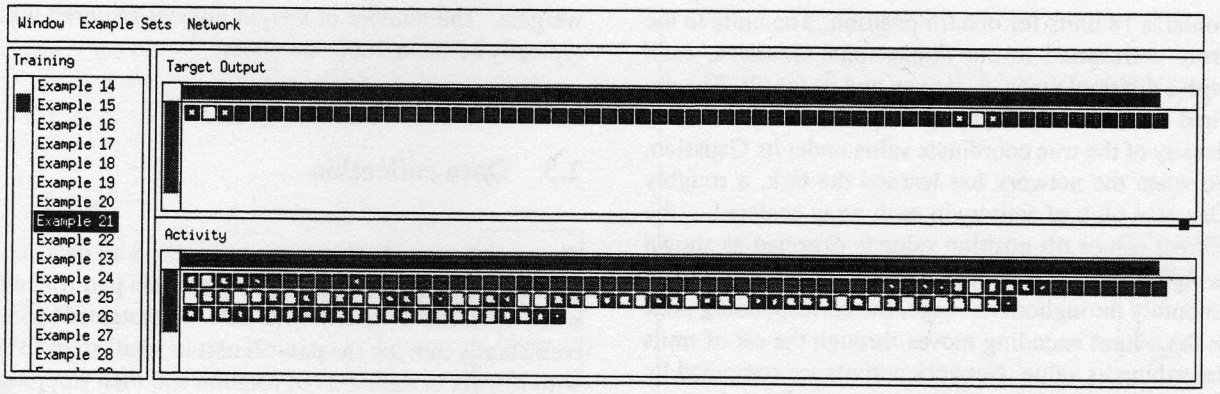

Figure 3: Vector of output units (shown as rectangles) representing true (target) pan position (first 41 units) and tilt position (next 18 units) is shown in top half of window. The lower half of the window shows 3 rows of unit activities: the first and longest row is the network's output for this input feature vector (note its closeness to the target vector). The second row is the hidden unit activities, the third is the input vector of features. Activity is coded as the area of the white blob.

Figure 5 and 6 below show an example of input image and its associated edge map.

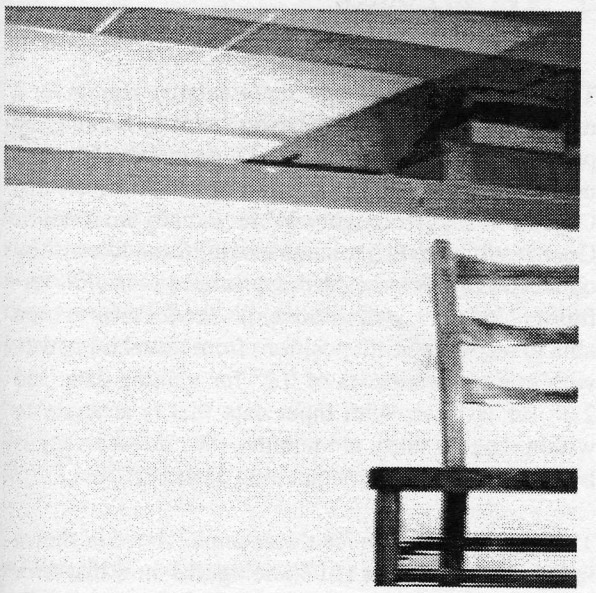

Figure 5: An image obtained from the camera.

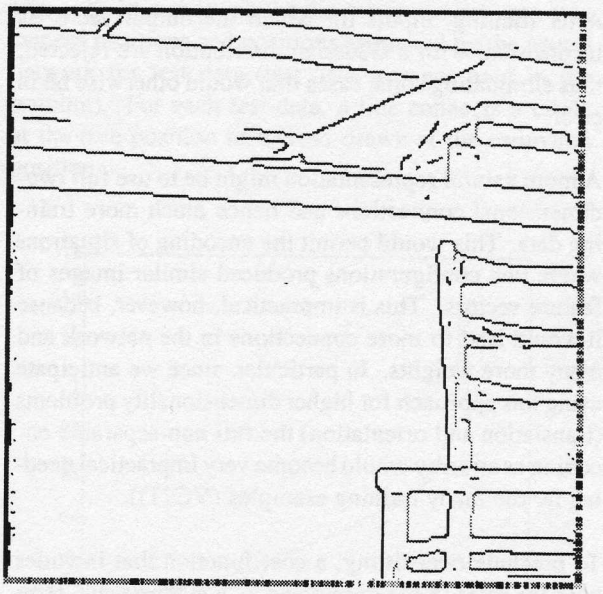

Figure 6: The edge map for the above input image.

2.2 The design of the neural network

The backpropagation algorithm (RHW86) was used to optimize the weights of a three-layer neural network that converts a visual input feature vector into an estimated camera position. The input layer of the network has 23 units whose activities represent the current visual input measurements. The hidden units in middle layer allow the network to compute non-linear functions of the inputs, they have sigmoidal activation functions, and by adapting their incoming weights they can learn to extract a set of features that are useful for estimating camera position. The output layer is composed of sigmoidal units that represent the estimated pan and tilt positions of the camera.

Two one-dimensional array of output units were used. One array contains 41 "radial basis" units whose combined activities encode one pan position, Another array

contains 18 units for one tilt position. The units in the array correspond to one-dimensional Gaussian, each with a different mean, $\bar{\theta}_i$ for pan and $\bar{\phi}_i$ for tilt. The desired activity of a unit is proportional to the probability density of the true coordinate value under its Gaussian. So when the network has learned the task, a roughly Gaussian blob of activity in each array centered at the correct pan or tilt position value is expected as shown in figure 3. If we sample the pan (or tilt) coordinate smoothly throughout its range, the corresponding peak in the output encoding moves through the set of units describing its value. Network outputs are computed by summing the products of the means ($\bar{\theta}_i$ $\bar{\phi}_i$) and output values o_i for each output unit:

$$\theta = \Sigma_{i \in \text{pan units}} o_i \bar{\theta}_i \qquad (2)$$

and

$$\phi = \Sigma_{i \in \text{tilt units}} o_i \bar{\phi}_i. \qquad (3)$$

After training, inputs for which the output vector is inconsistent with a Gaussian distribution are rejected, thus eliminating some cases that would otherwise be in error.

A more natural representation might be to use full two-dimensional connections and hence much more training data. This would permit the encoding of situations where two configurations produced similar images of feature vectors. This is impractical, however, because it would lead to more connections in the network and many more weights. In particular, since we anticipate using this approach for higher dimensionality problems (translation and orientation) the full non-separable encoding as an array would become very impractical needing far too many training examples (VC71)).

To preclude over-fitting, a cost function that includes both the error in the output and a "weight-decay" term that penalizes large weights (HS87) was minimized.

$$C = \sum_c \sum_j (y_{j,c} - \hat{y}_{j,c})^2 + \lambda \sum_i w_i^2 \qquad (4)$$

where c is an index over training cases, j is an index over output units, $y_{j,c}$ is a desired output, $\hat{y}_{j,c}$ is an actual output, and w_i is a weight. We found that the best performance was obtained with $\lambda = 0.00001$.

Backpropagation was used to compute the derivative of C with respect to each weight in the network and a conjugate gradient method was then used to update the weights. The number of weight updates required was typically between 2000 and 6000.

2.3 Data collection

The neural network was trained and tested on data collected by moving a two degree-of-freedom pan-tilt unit which mounted a camera. Real data was obtained by incrementally moving the pan-tilt unit in small steps (3°) with respect to each axis of rotation and then stopping and collecting one or more images. The collected data was separated into a training set and a test set which was composed of the readings from every 7th position. This test set was used to assess the performance of the neural network. The largest data set used had a total of 270 images collected from 270 different pan-tilt positions.

3 Performance

The figures below illustrate representative results for a region using the real robot environment by moving the pan-tilt units (Figures 7 and 8). The number of hidden units in the network is important for accurate results. Our results suggest roughly 50 hidden units are suitable. Good results with larger number of units can likewise be obtained provided weight-decay is used to avoid over-fitting. Under these conditions, our networks have been able to recover pan-tilt positions from visual input data with average accuracies of 0.3° for training data, and 2.3° for test data with input data that is accurate to within roughly thirty arc-minutes (i.e. the accuracy is better than the sample acquisition density).

The error distribution for the example above is shown in Figure 9 and Figure 10. These figures show that error per example for cases used for training and for testing (but not training). Note that the curves are shallow and rise only near their tails; this indicates that only a small percentage of cases accounts for a disproportionately large amount of the total error. This further suggests that if these cases were detected and rejected using the *consistency* measure proposed in the next section, the actual error accrued in practice would be smaller still. It is suspected that many of the occasional large errors are due to that some non-adjacent camera positions have similar perceptual structures. In such cases, the actual output of the net would be bimodal and it could be disambiguated by using rough positional information from

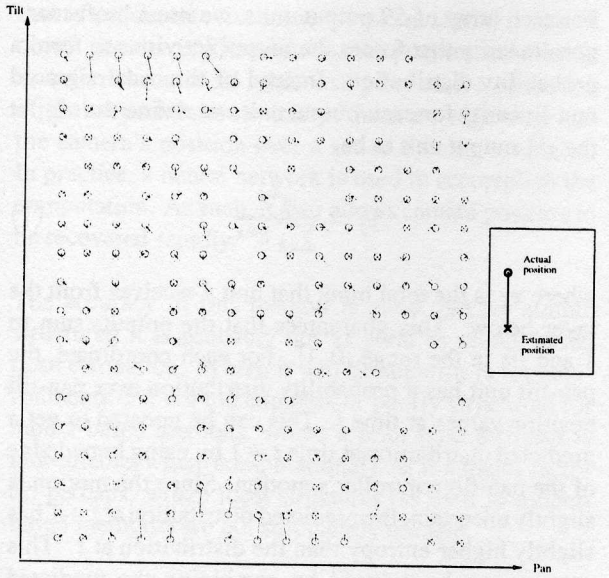

Figure 7: Plot showing error in estimated position as a function on camera position. For each image data, a line connects a circle at the true camera position to a cross drawn at the estimated position. Training data are shown.

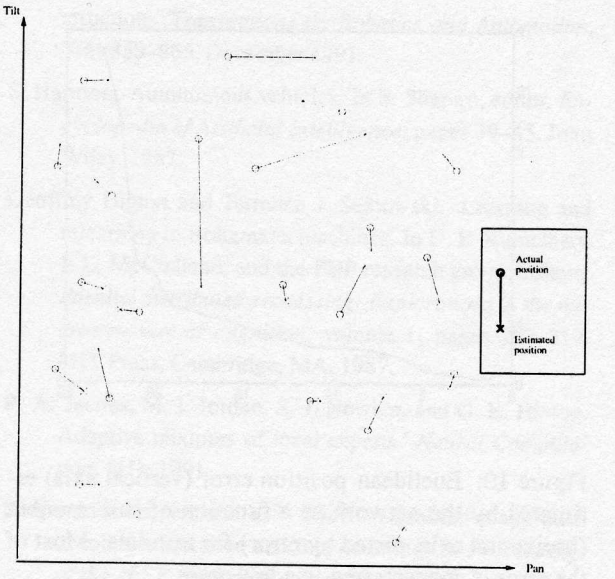

Figure 8: Plot showing the discrepancies between true camera positions and positions estimated by the neural network for test data (test data was not used in the training). For each test data, a line connects a circle at the true position to a cross drawn at the estimated position.

the pan-tilt controller's previous estimated position and its estimated motion. The ability to indicate uncertainty by giving a high entropy output distribution is a major advantage of the output representation. When using a single output unit for each pan or tilt value, the neural net minimizes its squared error by incorrectly outputting the average of the two positions that have very similar perceptual structure.

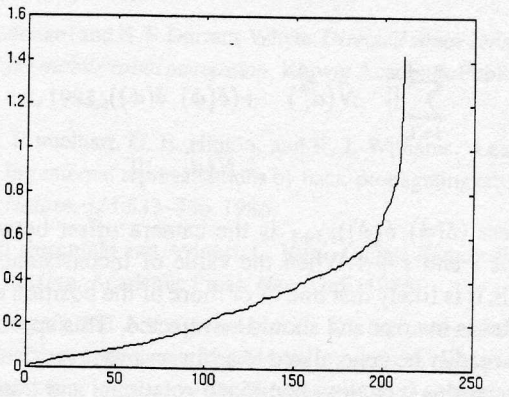

Figure 9: Euclidean position error (vertical axis) estimated by the network as a function of case number (horizontal axis, sorted by error) for training data. Most of the error is concentrated in a few cases.

4 Discussion

Although by-and-large the quality of the results from the network is good, there seem to be occasional outliers in the position estimates that have large errors. These may be due to views that have degeneracies that makes position estimation impossible, for example view that contain insufficient features or that resemble other views. Although some of these may be detectable *a priori* (such example views with no features), there remains the possibility that incorrect estimates will sometimes be produced. This potential difficulty can be resolved by exploiting the fact that dead reckoning is accurate over small intervals in space or time, the method is related to both Kalman filtering and median filtering (LDW92; RK76).

For small motions, robotic vehicles or manipulators can usually accurately recover their motion parameters. For two images acquired with an orientation separation known from dead reckoning by $(\delta(\theta), \delta(\phi))$, we have collections of images features $\vec{d}_1$ and $\vec{d}_2$, we can estimate $\vec{N}(\vec{d}_1^{\kappa}) = (\theta_1, \phi_1)$ and $\vec{N}(\vec{d}_2^{\kappa}) = (\theta_2, \phi_2)$. Thus we can define the **consistency** of a pair of measurements

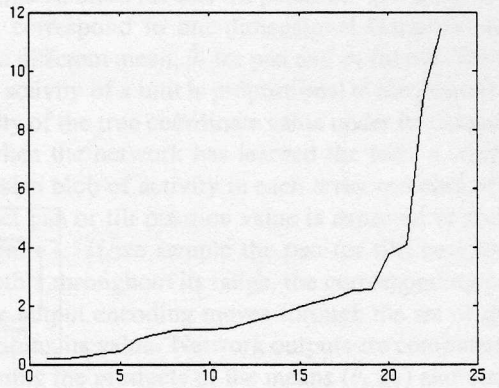

Figure 10: Euclidean position error (vertical axis) estimated by the network as a function of case number (horizontal axis, sorted by error) for test data. Most of the error is concentrated in a few cases.

to be

$$c = ||\vec{N}(\vec{d}_1) - \vec{N}(\vec{d}_2) + (\delta(\theta), \delta(\phi))||. \quad (5)$$

More generally, we can define the *inconsistency* of a *series* of measurements $i = 0..n$ as

$$\gamma = \sum_{i=1}^{n-1} || \vec{N}(\vec{d}_i^{\phi}) + (\delta(\phi), \delta(\phi))_{i,i+1} - \vec{N}(\vec{d}_{i+1})|| \quad (6)$$

where $(\delta(\phi), \delta(\phi))_{i,i+1}$ is the camera offset between steps i and $i+1$.When the value of inconsistency is high, it is likely that one of or more of the position estimates is in error and should be rejected. This approach can readily be generalized to arbitrary robot verification trajectories including combined rotational and translational motion[2]. Measurements that are not consistent can thus be discarded.

We have largely ignored the issue of how to best integrate the pan-tilt position information acquired from the current visual input with the position that can be inferred from the previous pan-tilt position and the pan-tilt controller's motion. Several approaches are possible, including the use of an *Extended Kalman filter* to integrate estimates over time (and combine odometry).

[2]A variety of more sophisticated fitting methods could also be used for computing *inconsistency*, but this does not appear necessary.

For each array of 59 output units, we use a "soft-max" non-linearity that forces the output activities to form a probability distribution. Instead of the usual sigmoid non-linearity for each output unit, we define the output the j^{th} output unit to be

$$\hat{y}_j = \frac{e^{x_j}}{\sum_k e^{x_k}} \quad (7)$$

where x_j is the total input that unit j receives from the layer below. This guarantees that the outputs sum to 1 and lie in the range $[0, 1]$. For each coordinate, the pan-tilt unit has a probability distribution over pan-tilt position values at time t. This can be updated to get a predicted distribution at time $t+1$ by using knowledge of the pan-tilt controller's motion. Since the motion is slightly uncertain, the predicted distribution at $t+1$ has slightly higher entropy than the distribution at t. This entropy can be reduced by combining the predicted distribution with a distribution based on the current visual input that is provided by the neural network. The obvious combination rule is to multiply together the corresponding probabilities in the two distributions and to re-normalize. This, however, makes unreasonable independence assumptions. For example, if the pan-tilt unit simply stays in one position, we do not want its distribution to become infinitely sharp. We therefore need to allow for the correlated errors in the position estimates at nearby positions.

It is likely that more sophisticated models for relating the robot pose to the statistical features could be developed and implemented within this framework. A further extension might be the use of "online bootstrap learning" to accomplish the image-to-position learning without many training examples. This would have the additional advantage of allowing the robot to adapt to changes in the environment without requiring the collection of additional training data This ability would stem from the fact that the knowledge of the motion provides a consistency constraint that must be satisfied by the position estimates at successive times.

5 Conclusions

We have described a technique for computing camera position from image data. The example we have described is based on recovering the orientation parameters of a camera. This approach to estimating camera position has the advantage of not requiring either an *a priori* model of the environment nor an environment

that is consistent with some simple model of scene or object geometry. The basis for the position estimate is an inferred statistical relationship between a collection of global parameters extracted from an edge map and the camera's position over a set of training examples. In practice, a neural network is used to accomplish the computation. As such, it also allows camera position to be recovered rapidly.

The accuracy of this method is good, but for simple structured it is probably not as good as model-based pose estimation. Where it will be superior is for environments that are too complex or poorly structured for model-based methods, where a prior pose estimate is unavailable, where computational cost is a major factor or, perhaps, where local structures (as used in model-based methods) are not sufficiently reliable.

In continuing work, it appears that this approach is extensible to the estimation of robot translation as well as orientation. Advantages over landmark-based methods are the reduced reliance on any single landmark and the system's ability to automatically learn the relationship between its percepts and the environment.

References

Kenneth Basye and Thomas Dean. Map learning with indistinguishable locations. In M. Henrion L. N. Kanal J. F. Lemmer, editor, *Uncertainty in Artificial Intelligence 5*, pages 331–340. Elsevier Science Publishers, 1990.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the vapnik-chervonenkis dimension. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 273–282, Berkeley 1986, 1986. ACM, Salem.

Ronen Basri and Ehud Rivlin. Homing using combinations of model views. In *Proceedings of the International Joint Conference of Artificial Intelligence*, pages 1656–1591, Chambery, France, August 1993. Morgan Kaufman Publishers.

John F. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-8:679–698, 1986.

Ernest Davis. *Representing and Acquiring Geographic Knowledge*. Pitman and Morgan Kaufmann Publishers, Inc., London and Los Altos, California, 1986.

Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. Robotic exploration as graph construction. *Transactions on Robotics and Automation*, 7(6):859–865, December 1991.

S. Harmon. Autonomous vehicles. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 39–45. John Wiley, 1987.

Geoffrey Hinton and Terrence J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart, J. L. McClelland, and the PDP research group, editors, *Parallel distributed processing: Explorations in the microstructure of cognition*, volume 1, pages 282–317. MIT Press, Cambridge, MA, 1987.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1), 1991.

Benjamin J. Kuipers and Y. T. Byun. A qualitative approach to robot exploration and map-learning. In *Proceedings of the IEEE workshop on spatial reasoning and multi-sensor fusion*, pages 390–404, Los Altos, CA, 1987. IEEE.

Akio Kosaka, Min Meng, and A. C. Kak. Vision-guided mobile robot navigation using retroactive updating of position uncertainty. In *Proceedings of the International Conference of Robotics and Automation, Volume 2*, pages 1–7, Atlanta, GA, May 1993. IEEE Computer Society Press.

J. J. Leonard and H. F. Durrant-Whyte. *Directed sonar sensing for mobile robot navigation*. Kluwer Academic Publishers, 1992.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by back-propagating errors. *Nature*, *323*:533–536, 1986.

Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing*. Academic Press, New York, 1976.

V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.