Visual Memory-Based Learning for Mobile Robot Navigation

Daniel Nikovski

Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
daniel.nikovski@cs.cmu.edu

Abstract

The paper describes an efficient memory-based learning scheme for the localization of a door in a visual scene, estimation of the distance to it from a mobile robot, and steering the robot through the door. Digital signal processing techniques are used to speed up the run-time processing and achieve performance much faster than other learning techniques would permit. Reported are results from experiments with several varieties of memory-based learning on two mobile robots.

1 Introduction

Navigating from point to point in a working environment is the main task of a mobile robot. A number of techniques are used for practical applications: metric and topological maps, sonar-derived evidence grids, sensor-based planning, etc. These techniques solve the problem of high-level navigation and usually rely on low-level primitives for navigating between close points. In a typical environment such as an office building, low-level navigation in the hallways can be done reliably by means of sonar-based maps and local obstacle avoidance routines. Entering and exiting rooms, however, presents a more challenging task. Local obstacle avoidance can be used for navigating through any narrow passage in a local occupance grid, but the robot cannot be sure that it has successfully exited the room, because any gap between two pieces of furniture in the room can be mistaken for the door. For successful exiting and entering rooms, it is necessary to first recognize where the door is.

Recognizing doors in visual scenes is not different from recognizing any other objects. A significant amount of work has been done on finding faces [5, 6] and cars [2] in images. Recent systems rely almost exclusively on learning techniques instead of on model based ones, because it is very difficult to create a model (either analytical ot other) of faces, cars, doors, and other non-elementary objects. Using learning techniques results in a more autonomous operation and is much preferable for mobile robots.

Most of the existing learning systems for image recognition use some type of neural network (NN). The approach reported in this paper uses a different scheme — memory-based learning (MBL) [3]. While having similar learning power to that of NN, MBL has an important advantage over NN — it does not require any training, as opposed to the sometimes prohibitively long training time of NN systems. Instead, the computational load is shifted to run-time. This, however, is not necessarily a problem — digital signal processing techniques (DSP) can be used for speeding up the computation, and as a result, faster execution than that of NN is possible even at run-time.

Section 2 describes the task of finding a door in a visual scene and estimating the distance to it, as well as how the task would be solved by an NN-based system. Section 3 presents our approach to the problem with an MBL system and the DSP techniques that can be used for speeding it up. In section 4, experimental results of the precision of the schemes on real robots are reported. Section 5 describes how the estimate of the position of a door, found by MBL is used to steer a robot through the door. Section 6 lists several directions for future work, and section 7 concludes.

2 Using Vision for Estimation of Direction and Distance

The specific task at hand is to steer the robots Xavier and Amelia [7] through a typical office door. These robots have a round mobile base of diameter 60cm and 50cm, respectively. Since a typical door is about 80cm wide, the tolerances are quite small and navigation has to be very careful, especially for Xavier. The precise estimation of the position of the door is thus essential for successful navigation.

A convenient representation of the position of the door are the polar coordinates of the midpoint of the line that connects the two door posts. The robot needs to find the direction to the door (the angle between the current direction of the robot and the line that connects its center with the midpoint of the door), as well as the distance from the center of the robot to that midpoint.

These two coordinates can be found independently by means of MBL. The first step is to acquire a

wide panoramic image of the surrounding scene. Since the cameras of the experimental robots have a 55° field of view, we chose to take an image every 20°. For a whole panorama of 360°, 18 images are acquired, with about 7° overlap between images. The second step is to find the door in the panorama, and the third is to estimate the distance to the door.

Before presenting our approach, let's first consider how a typical NN-based system would solve the second step, listed above — for example, a face-detection system. Such a system learns to match an image window to the probability that this window contains a face. At run-time, a window is slid over the acquired unknown image, and the NN recognizer is run for each such window. If the output of the NN is above certain threshold, a face is detected at that position of the window. The complexity of the whole operation is $O(KhMNd^2)$ for a panoramic image of height N and width M, $M > N$ and a square window of size d, $d < N$; K is the number of different scales, for which testing is done. Note that d varies for different scales and can go up to N. Finally, h is the number of units in the hidden layer of the neural net.

The complexity of this method for the problem of door detection would indeed be $O(KhMN^3)$, because doors are large objects and d is close to N. This complexity is considerable. It would be desirable to have faster computation at run-time. This can be achieved by the MBL scheme described in the next section. Furthermore, as is typical of all MBL systems, it has zero training time.

3 Fast Visual MBL

MBL, also known as lazy or delayed learning, is a general function approximation method that matches input vectors $\mathbf{x}$ to output vectors $\mathbf{y}$ [3]. It differs from NN in the way it treats the training examples. Instead of using them in order to adjust a set of parameters (weights and biases), the example pairs $(\mathbf{x}, \mathbf{y})$ are simply stored in memory until run-time. When a query $\mathbf{x}_{new}$ is made, the learning algorithm computes the distances from $\mathbf{x}_{new}$ to the $\mathbf{x}$ vectors of each of the training examples, and uses these distances to build a model for the computation of the required value of $\mathbf{y}_{new}$. After the query is answered, the model is discarded, and a new one is built when another query arrives.

We adopted the MBL approach for our problem as follows. Images of doors are stored in memory, all with the same size — N by N pixels. They show the doors at varying scales and from varying directions, but in all of them the door is centered. In the order of $2K$ to $4K$ instances seems to be enough for successful recognition (K is the number of scale steps for a corresponding NN system). Let the number of such instances be P.

As described in section 2, the wide image of the environment is taken as a panorama of 18 images. The width M of the whole panorama is then $18N$. Accordingly, there are MN candidate points for the position of the door within the panoramic image. If we are interested only in the direction to the door on the horizontal axis, the number of candidate directions is M.

These images constitute the input vectors, $\mathbf{x}$, of the training pairs. The output vectors, $\mathbf{y}$, are different for the two subtasks, as described below. After the training examples are obtained, four additional elements have to be determined in order to do MBL ([3]): a distance metric, the number of neighbors to look at, a weighting function, and a way to fit with the local points. These elements also differ for the two subtasks of direction and distance estimation.

3.1 Direction Estimation

The output vectors $\mathbf{y}$ for the problem of door detection are the probabilities that an image $\mathbf{x}$ contains a door in its center. For images of the door (positive examples), this probability is $\mathbf{y} = 1$. For images that do not contain the door (negative examples), the probability is $\mathbf{y} = 0$. Obtaining positive images does not present any difficulties; however, obtaining negative ones is a hard problem, pervasive to all object recognition systems [5, 6]. This is so because the space of negative instances (all images that do not contain doors, in our case) is much bigger than the space of positive instances (images that do contain a door). Since representative sampling of the negative space is not easy, we have adopted an approach that does not need negative instances under certain assumptions.

The distance measure used is based on the correlation function $Q_{\mathbf{xu}}$ between the template $\mathbf{x}$ and the unknown image $\mathbf{u}$:

$$Q_{\mathbf{xu}}[p,q] = \sum_{m=1}^{N}\sum_{n=1}^{N} \mathbf{x}[m,n]\mathbf{u}[(m-p)\%N, (n-q)\%N] \tag{1}$$

Here $\mathbf{x}$ and $\mathbf{u}$ are both square N by N images, and p, q, m, and n range from 1 to N. The modulo operators % serve to keep the differences $(m-p)$ and $(n-q)$ within this range too. When the images $\mathbf{x}$ and $\mathbf{u}$ are normalized to unit length, the two images are vectors on the unit hypersphere in N^2-dimensional Euclidean space, and the values of the correlation function $Q_{\mathbf{xu}}[p,q]$ are equal to the cosine ρ of the angle between the two vectors. The value ρ is thus a valid similarity measure between the two vectors, although not a proper distance measure. To turn it into one, several different expressions can be used, such as:

$$d_1(\mathbf{x},\mathbf{y}) = |\arccos\rho|$$

$$d_2(\mathbf{x},\mathbf{y}) = (\rho-1)^r,$$

where r is an even number. Computationally, we can obtain the cross-correlation function between two images very efficiently by means of the Fast Fourier

Transform (FFT)[1]. For two images $\mathbf{x}$ and $\mathbf{u}$, the cross-correlation function $Q_{\mathbf{xu}}$ is given by

$$Q_{\mathbf{xu}} = \mathcal{F}^{-1}[\mathcal{F}(\mathbf{x})\mathcal{F}(\mathbf{u})^*],$$

where $\mathcal{F}$ denotes Fourier transform, $\mathcal{F}^{-1}$ inverse Fourier transform, and $*$ denotes complex conjugate. That is, we multiply the 2D spectrum of $\mathbf{x}$ by the complex conjugate of the 2D spectrum of $\mathbf{u}$, and take the inverse Fourier transform of the product to obtain the cross-correlation function of the two images in an N by N array. If FFTs are used, this would produce N^2 correlation coefficients with $O(N^2 \log N)$ computations, instead of the $O(N^4)$ for the straightforward application of formula (1).

Using this efficient technique, we can compute the distance between all memory instances and all 18 images of the panoramic scene. The complexity of the whole computation is $O(PMN \log N)$, compared to $O(KhMN^3)$ for an NN. But P is $O(Kh)$ — the number of memory instances P is comparable to the product of the number of tested scales K and the number of hidden units h. This means that the proposed MBL method achieves a speedup in the order of $N^2/\log N$ with respect to a NN-based system. For a typical value such as $N = 64$, the speedup is more than 500 times.

Once the distances are known, several options are possible for producing an estimate of the position of the door in the scene. The simplest one is to find the maximal ρ across all correlation arrays, and assume that the door is at the corresponding location (p', q') in the image. Since the pan angle at which the image was taken and the angle covered by a single pixel are known, it is easy to compute from (p', q') the angle α to the midpoint of the detected door in the local coordinate system of the robot.

This method would work unless there exist objects that look very much like doors, but are not doors. In such case, it would be necessary to store memory instances of images of these objects, assigning to them output values $\mathbf{y} = 0$ and perform either nearest neighbor, kernel regression, or locally weighted regression (as described in the following subsection for distance estimation). We found that the simple method of finding the maximum correlation coefficient worked reliably for our purposes, and used the more advanced MBL methods for distance estimation only. This solution eliminates the need for negative examples under the discussed assumption.

3.2 MBL for distance estimation

Once the position of the door in the panoramic view is found, the distance to it can be estimated with the help of the already computed distance measures. At this step, only the images from the panorama that correspond to the detected direction are considered. Because of the overlap of 7°, two such images might exist — that is, pixels (p', q') of the best matching image $\mathbf{u}'$ might correspond to the same physical direction as pixels (p'', q'') in a neighboring image $\mathbf{u}''$. The correlation coefficients for the positions (p', q') and (p'', q'') are extracted from the correlation arrays for all memory instances. There are either P or $2P$ such correlation coefficients ρ_i.

The output values $\mathbf{y}$ of the training examples for this task are the distances from the door to the robot. While the direction estimation task uses MBL for classification, the distance estimation one uses it for approximation of a real-valued function. Three kinds of MBL were tested on the task: k-nearest neighbor (kNN), kernel regression (KR), and locally weighted regression (LWR). In kNN, the predicted output for a new image $\mathbf{x}_{new}$ is the average of the $\mathbf{y}$-values of the k best matching memory instances. In kernel regression, a weighting coefficient is first computed for each of the coefficietns ρ_i [3]:

$$w_i = (1 + \rho_i)^k, \tag{2}$$

where latge k indicates a preference to large values of ρ_i. The predicted output value is the weighted average of all output values of the training examples:

$$\mathbf{y}_{new} = \frac{\sum w_i \mathbf{y}_i}{\sum w_i}.$$

LWR is a combination of kNN and KR: the weighting function is $w_i = 1 + \rho_i$, and the regression is done only on the k best matching examples.

4 Experimental Results

4.1 Direction estimation

We used a set of $P = 30$ memory instances, obtained in the following manner. The robot was positioned at the midpoint of the door and backed up on a straight line at $30cm$ steps so that the camera was kept foveated at the midline of the door all the time. This was repeated for two such straight lines whose orientation differs by about 20 degrees. The purpose of this procedure was to sample the door at different scales and from different directions. Note that the image is much more sensitive to changes in scale than to changes in the direction of observation — we sampled the scale at 15 values, while the observation direction at only 2.

A total of 15 queries were performed by positioning the robot at a random place in the room and giving it random orientation, which was recorded for use in the subsequent estimation of the precision of the method. For each of these 15 positions, a panoramic set of 15 images was taken by rotating the camera at 20° degree increments, spanning the surrounding panorama from $-140°$ to 140° (the pan-tilt unit of the robot cannot do a full 360° degree span.)

For each of the 15 queries, we computed the predicted direction of the door and the error with respect to the true direction, which we had recorded at the time of taking the testing sequences. The measured root mean squared errors (RMSE) are shown in Table 1 together with the average correlation for the best matching image,

for several resolutions of the images. Computation times are shown in seconds for Sun Sparc 4.

Table 1. Direction estimation.

Image size $X \times Y$	Comp. time [sec]	Nearest Neighbor	
		$RMSE°$	$\bar{\rho}$
64×64	25	3.8439	0.7515
32×32	5	3.6314	0.7879
32×16	3	3.6314	0.8091
16×16	1	3.0603	0.8080

The results show that even for a 16×16 image, the average error in direction is only $3°$.

4.2 Distance estimation

In another experiment, distance estimation was tested. Thirty memory instances $\mathbf{x}_i$, $i = 1, 30$ were acquired and the corresponding distances $\mathbf{y}_i$ to the door were recorded. The distances varied in the range from $0cm$ to $420cm$, in $30cm$ increments along two lines from the door. Four test cases (not on the two lines) were recorded as well. Table 2 shows the RMSE from applying three types of MBL, with varying parameters. In this table, k denotes the number of neighbors for local regression and nearest neighbor, while for kernel regression it denotes the power in equation (2).

Table 2. Distance estimation RMSE cm.

type	1	2	3	4
KR	49.54	37.03	32.15	27.26
kNN	22.38	17.92	27.31	25.44
LWR	22.38	17.86	24.61	23.89

The best results were obtained with local regression between the two closest points. This result is consistent with the model of [4] for visual pose estimation. Furthermore, the obtained accuracy is very close to the one obtained by 2NN, which suggests that the correlations with the two best matching memory instances have very similar values and local regression practically averages the distances of the two closest examples.

5 Robot Steering

Based on the experimental results described above, we concluded that local regression between two neighbors on 16×16 images would give enough precision to steer the robots through doors. It was found experimentally that this can indeed be done repeatedly and reliably. MBL was used to estimate the position of the midpoint of the door, and the robot was rotated to head in this direction. Along this line, a goal was determined $150cm$ beyond the midpoint, and the goal was given to the local obstacle avoidance of the robot. The robot was able to successfully reach the goal within a prespecified tolerance, usually correcting only slightly its course from the chosen straight line. Since the tolerances on the larger robot, Xavier, are very tight, a speed as low as $3cm/s$ is necessary for successful navigation. On the smaller robot, Amelia, a speed of $10cm/s$ is possible.

6 Future Directions

The current implementation of the system is in an open-loop mode — the robot first determines the goal point and then uses obstacle avoidance routines to reach it. A closed-loop control system can be created to reestimate the goal position and continuously correct the navigation routine. Other possible extensions of the system include the ability to select autonomously the memory instances, as well as to recognize objects other than doors.

7 Conclusion

The paper presented a method for the visual estimation of the location of a door in the local frame of a mobile robot. Memory-based learning techniques were used, which eliminated the need for training. Of these, locally weighted regression on two nearest examples provided best results. The similarity measures necessary for MBL were obtained very efficiently by means of DSP techniques, which resulted in a fast, robust, and practical system for navigation through office doors.

References

[1] Jain, A. (1989). Fundamentals of Digital Image Processing. Englewood Cliffs: Prentice Hall.

[2] Koller, D., Weber, J., and Malik, J. (1993). Robust multiple car tracking with occlusion reasoning. CSD-93-780. University of California, Berkeley, 1993.

[3] Moore, A.W., Atkeson, C.G., and Schaal, S. (1995). Memory-based Learning for Control, CMU Robotics Institute Technical Report CMU-RI-TR-95-18, April 1995.

[4] Murase, H. and Nayar, S.K. (1995). Visual learning and recognition of 3-D objects from appearance. *Intl. Journal of Computer Vision*, vol. 14, pp. 5-24.

[5] Rowley, H.A., Baluja, S., and Kanade, T. (1996). Human face detection in visual scenes. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo (Eds.), *Neural Information Processing Systems 8*, pp. 875-881, Cambridge, MA: MIT Press.

[6] Sung, K.-K. and Poggio, T. (1994) Example-based learning for view-based human face detection. A.I. Memo 1521. CBCL Paper 112, MIT, 1994.

[7] Xavier Project, Learning Robots Lab, CMU, http://www.cs.cmu.edu/Groups/xavier/www/.