ISSN 1361 - 6161

THE UNIVERSITY of MANCHESTER

Computer Science
University of Manchester

Selecting Landmarks from Sonar Scans for Mobile Robot Navigation

Stephen Marsland and Ulrich Nehmzow

Department of Computer Science

University of Manchester

Technical Report Series

UMCS-00-8-1

Selecting Landmarks from Sonar Scans for Mobile Robot Navigation*

Stephen Marsland and Ulrich Nehmzow

Department of Computer Science
University of Manchester
Oxford Road, Manchester, UK.
{smarsland,ulrich}@cs.man.ac.uk

25.8.2000

*Refereed by: Andreas Bühlmeier

Abstract

Navigation using landmarks is one way in which researchers have tried to overcome the inaccuracies of using a robot's internal odometry to localise the robot. Systems which use landmarks often take regular readings of an environment, and store these perceptions in a map. Unfortunately, this method means that the most recognisable landmarks may be missed because they are between sampling points. In addition, many of the perceptions which are added to the map are very similar if not identical. This is a waste of map space.

A collaborative project between Manchester and Bremen Universities has been investigating methods of automatically selecting suitable landmarks to aid navigation.

In this report, the problem of landmark selection is tackled using neural network techniques. A new method of selecting landmarks is described, and the results compared with an alternative technique developed earlier. The experiments undertaken show that the same landmarks are predicted consistently over several runs in the same environment.

1 Introduction

To minimise the use of dead-reckoning in robot navigation, which is unreliable due to incorrigible wheel slippage, many navigation systems for mobile robots use perceptual landmarks to generate maps and to localise the robot with respect to these maps. One question is how these landmarks should be selected. Some systems [10, 2] take sonar scans at regular intervals, with each sonar scan being a landmark which is put into the map. In many environments, such as corridors, a large number of perceptions are identical, and so the map is filled up with information which does not aid navigation. In addition, because landmarks are taken at discrete intervals, conspicuous landmarks could be missed because they lie between sampling points.

Other researchers have investigated systems which navigate using user-defined landmarks such as doors, or line segments taken from camera images [5]. However, because the perceptions of the robot are very different to those of a human, these systems can be prone to failure.

This paper investigates ways of autonomously determining suitable landmarks for autonomous navigation by a mobile robot. These landmarks are perceptual snapshots of the environment taken by the robot at various points as it explores an environment.

We argue that in order to be useful for navigation, a landmark should be perceived in the same place consistently. Selecting suitable landmarks for navigation is a non-trivial task, as we need to pick those features which are recognised on *each* traversal of an environment. For the landmarks to be useful to the robot, they need to be chosen by the robot, so that human assumptions about the perceptions of the environment are not involved.

The argument underlying the approach presented in this paper is that a landmark should be both conspicuous and reliably perceived. In order to identify those sensory perceptions that "stick out", i.e., are conspicuous, our robot first learns a model of the usual temporal relationships between the signals perceived by its various sensors (the model acquisition phase). This model is later used to make a prediction of the most likely sensory perception at the next time step.

In the landmark selection phase, those locations where predicted and actual perception differ considerably are selected as landmarks, on the grounds that perceptions are so "unusual" at that location that they were not predictable by the model of "normal" perceptions.

In both the model acquisition phase and the landmark selection phase a number of different methods were tried, to determine which ones produce the most consistent selections. All these methods are discussed later in this report.

2 Experimental Approach

2.1 Data Collection

The perceptions of the robot as it explores an environment consist of the returns from sonar sensors placed around the periphery around the robot. The responses of these sonar sensors were collected by a robot as it travelled in a number of environments. The Bremen robot, *Rolland*, shown in figure 1, is a semi-autonomous wheelchair made by the German company Meyra. It has 27 sonar sensors arranged

Figure 1: The Bremen semi-autonomous wheelchair, *Rolland*.

Environment	Description	Approx. dist. to wall (cm)	Approximate length (m)
A	Lab corridor, from lab to fire doors	82	10
B	Lab corridor, from fire doors to lab	82	10
C	By lifts, from Ladies WC to Mens WC	87	10
D	By lifts, from Mens WC to Ladies WC	87	10
E	Opposite lifts, from Room 8180 to 8220	65	10
F	Opposite lifts, from Room 8220 to 8180	65	10
G	In kitchen corridor	82	6

Table 1: A summary of the environments used to collect data. All the environments are on the 8th floor of the MZH building at Bremen University, shown in figure 2.

around the robot, as can be seen in the figure. These sonar sensors return a reading between 14 cm and 175 cm, where anything closer that 14 cm returns 14, and anything further away returns 175 cm. Control software interprets steering and motion commands from the user given via a joystick and performs the commands, providing that they can be carried out safely.

The data collection procedure consisted of driving the wheelchair in a straight line, parallel to a wall, through an environment. Every 20 cm, the wheelchair stopped and polled the sonar sensors. 20 runs were made in each environment, with each run starting in the same place, as accurately as could be controlled.

Table 1 shows the different environments in which data was collected, all of which are on the 8th floor of the MZH building at Bremen University, shown in figure 2. All environments were sections of corridor. The sonar sensors polled were those on the side of the robot which faced the side of the corridor which has doorways on it. This is because the other side was generally just blank wall, or, in the case of environments C, D, E and F, empty space.

The data collected by the robot is intrinsically noisy, with noise coming from such sources as measurement and odometry error, crosstalk between sensors and specular reflection. The measurement errors are particularly noticeable in this case because of limitations in the accuracy of the odometry used to measure the 20 cm between stopping places, the lack of fine control in positioning and the difficulties of maintaining a straight wall-following trajectory. However, for a landmark prediction mechanism to be useful, it must be able to deal satisfactorily with problems like these - no robot control program can ever be perfectly accurate.

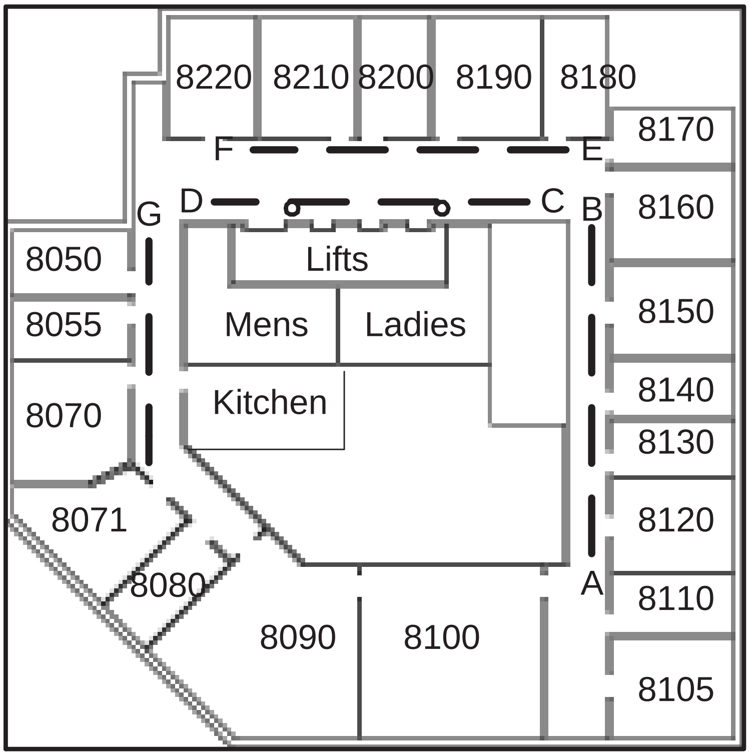

Figure 2: A map of the 8th floor of the MZH building at Bremen University, where the data was collected. The letters show where that data run started, and the lines show the intended robot trajectory.

2.2 Sonar Signal Preprocessing

Noise appears in the sonar readings in many forms. Some of them, such as crosstalk between sensors, and specular reflection, where a sonar pulse rebounds from an object away from the robot, which both appear as sudden spikes in the readings, can be removed by using preprocessing techniques. Unfortunately there is little that can be done for the random fluctuations in readings which also occur.

One way to remove this sort of noise is through the use of a median filter. The median filter replaces the sonar reading at each time t, with the median average of the previous n. In this case n was chosen to be five, so that for each sensor input, the median of the readings for that sensor at times $t, t-1, \ldots, t-4$ was presented. This has the effect of smoothing any large differences between perceptions taken at similar times, although at the cost of slightly delaying and blurring the onset of any abrupt changes in the readings. As this is a consistent effect between runs, it is not a problem for the application considered here. The effects of using a median filter as preprocessing for the neural network were tested. The results are shown in figure 9.

3 First Phase: Acquiring the Model of the Relationship Between Subsequent Sensory Perceptions

The technique of selecting landmarks by detecting places where the relationship between subsequent sensory perceptions does not conform to a a general model, requires that a sufficiently detailed general model of the temporal relationship between sensory perceptions is available. One way to acquire such a model is through the use of a neural network. In effect, the network was learning to predict one step ahead in a time series, given the current sensor reading.

The network architecture used was that described in [3]. This consists of a Perceptron, but with the connections between the input and output nodes being reduced in such a way that output nodes (which predict the reading of a particular sensor at the next time step) only receive input signals from sensors that are "ahead", i.e., sensors that have already perceived environmental features that will be perceived later by the sensor being modelled. The reading of each sonar is also used in the prediction of its own reading at the next step. The resulting architecture is shown in figure 3. A bias input, permanently fixed at -1, and connected to all of the output nodes was also used. This network is referred to as the "modified Perceptron" in this report.

Besides the modified Perceptron described above, two further neural networks were also investigated. Both are based on the Perceptron with the modified connections described above, but included temporal learning.

These two methods were the lagged Perceptron, shown in figure 4, where the inputs to the network are the perceptions of the robot at both the current time, t, and the previous time, $t-1$, and the recurrent Perceptron, shown in figure 5, where the outputs of the network at time $t-1$ are used as inputs at time t, together with the perceptions are time t. Initial experiments demonstrated that adding information from further back in time did not benefit the predictions of the network, and considerably more training was required.

To train these networks, early stopping [1] was used to decide how many learning iterations should be applied. Early stopping monitors the error of the network, evaluated on an independent validation data set, and stops the training of the network when the prediction error of the validation set stops decreasing. This suggested that about 20 complete runs through the data, were sufficient with the learning rate of $\eta = 0.2$.

For updating the networks' weights the Perceptron learning rule (equation 1) was used.

$$\vec{w}_i(t+1) = \vec{w}_i(t) + \eta(\tau_i - o_i) \cdot \vec{\imath}, \tag{1}$$

where $\vec{\imath}$ is the input vector to the network, $\vec{w}_i$ the weight vector of output unit i, τ_i the target value for output unit i, η the learning rate (0.2 in our experiments) and o_i the output of output unit i given by equation 2.

$$o_i = f(\vec{w}_i \cdot \vec{\imath}), \tag{2}$$

where f is the transfer function. In our experiments this was the sigmoid function given by equation 3.

$$f(x) = \frac{1}{1 + e^{-2x}}. \tag{3}$$

Six sonar readings were used for the neural network training. The sensors used were those on the side of the robot that faced the wall that was being followed. In each of the environments there was only one wall which contained changeable features, the one with doorways and other features on it. This was the wall which was followed. The side of the robot that faced this depended on whether the robot was going along the corridor using a right-hand or left-hand wall follow. Figure 6 shows the arrangement of the sonar sensors on the robot and marks which sonars were used for data collection.

4 Second Phase: Landmark Selection

Once the general model of the relationship between subsequent sensory perceptions has been acquired, current sensory perceptions can be used in conjunction with the model to predict future sensory perceptions. The perception at the current time is used as input, and the output of the network, produced using equation 2, is taken as a prediction of the next perception. The network error can then be calculated as the difference between the predicted and actual perceptions.

In the landmark selection technique presented here, we use the places where the prediction of the general model is bad — that is, the places where the error is high — as landmarks. The "quality criterion" for suitable landmarks here is that they are selected *consistently*, and our theory is that the places where the model breaks down (resulting in bad predictions) will be the same on every run. The problem is to detect these peaks in the error curve which are above the level of noise in the data. Two ways of doing this are described below.

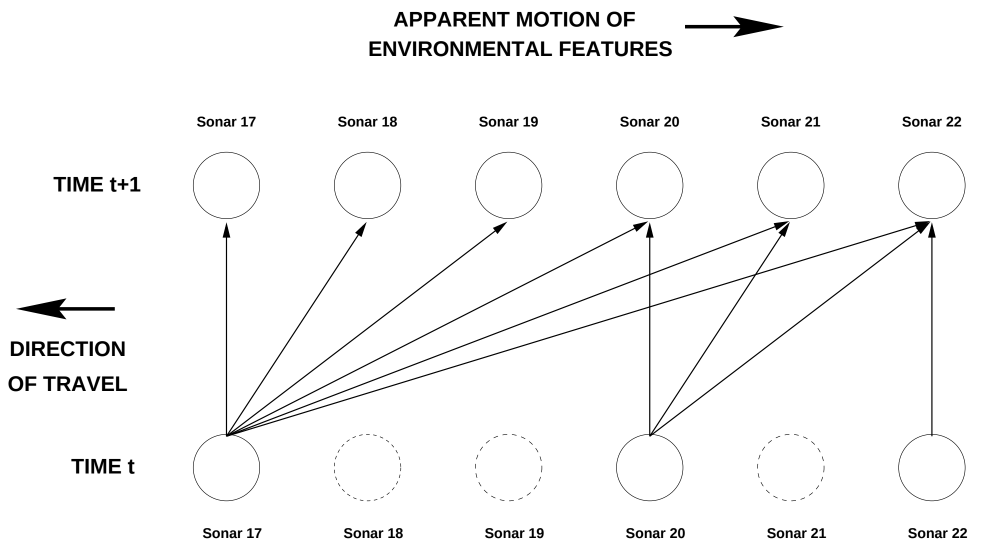

Figure 3: The neural network architecture of the Perceptron. For clarity, the connections are only shown for a subset of the neurons, 17, 20 and 22 in this case. A bias input, which connected to all output units was also used (not shown). (Taken from [3]).

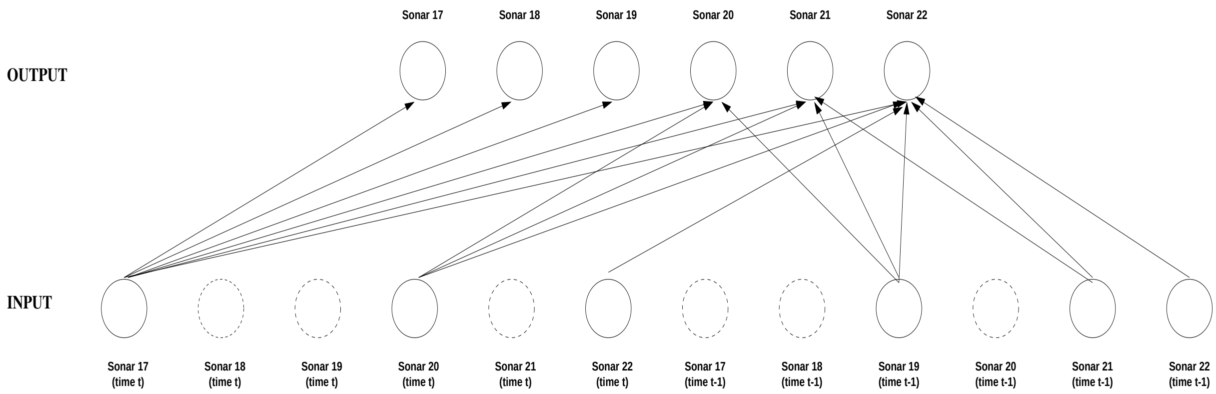

Figure 4: The lagged Perceptron. The inputs to the network comprise the sonar readings at times t and $t-1$, with the reduced connectivity used for both sets of inputs. For clarity, the connections are only shown for a subset of the neurons.

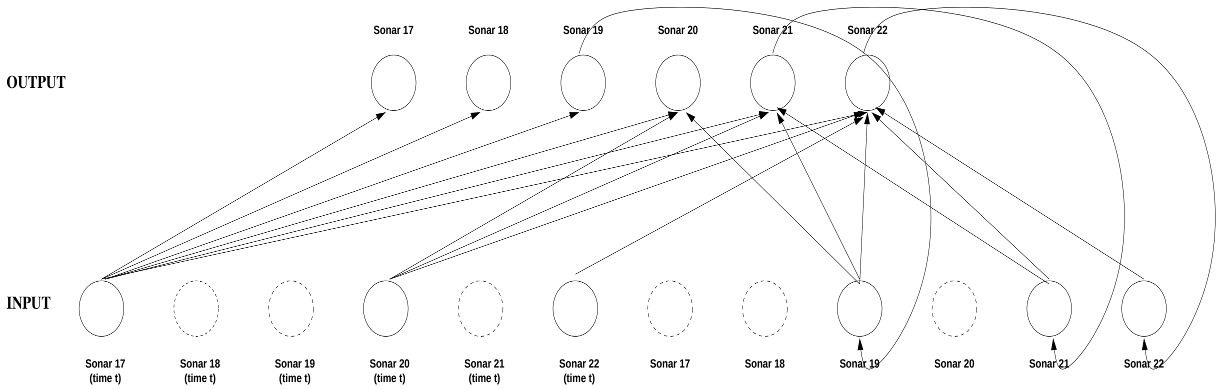

Figure 5: The recurrent Perceptron. The outputs of the network, the predicted perception at the previous time step, are used as inputs as well as the current perceptions of the robot. Again, for clarity, the connections for only some of the neurons are shown.

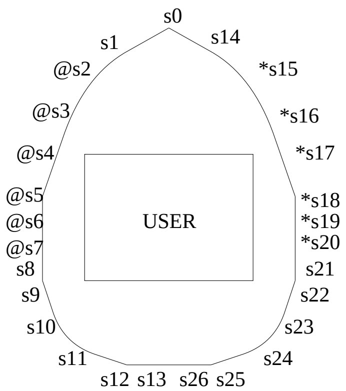

Figure 6: The layout of sonar sensors on the wheelchair. The two sets of sonar sensors used are marked with '@' and '∗'. Only one of the two sets of sensors were used at any time, the set selected being the one which faced the wall being followed.

4.1 Selection Method 1: Peak Detection

There are a number of ways in which peaks in the error curve, F_1, can be found. The problem is that the curve is very noisy, as can be seen in figure 7. Using a Perceptron to train the network means that the accuracy of the predictions is never particularly high, so we are looking for peaks which stand out above the residual noise level. This can be difficult as the noise level is high.

The peak detection itself can be done by looking for crossing points of the first derivative of F_1. This is approximated by considering points where the sign of $\frac{\Delta t}{\Delta F_1}$ changes. Typical error curves are shown in figure 7. It can be seen that, while there are a number of significant peaks, there are also several minor peaks within the noise level of the data. These would also be detected by a peak detection algorithm, which is undesirable.

To select only major peaks, therefore, a number of filtering steps were carried out before the peak detection. These filters are designed to smooth the curve, reducing the number of peaks, so that only significant ones remain. A summary of the filter stages follows, more details are given in [3, 8].

Summation The current error value, $F_1(t)$, was summed with the previous 9 to yield F_2 (equation 4), providing an accumulated value of the last 10 values.

$$F_2(t) = \sum_{i=0}^{9} F_1(t-i). \tag{4}$$

Median Filter The last nine values of F_2 were median filtered in order to remove major noise peaks from the curve. This provides a value $F_3(t)$ for each time t.

Smoothing The curve was then smoothed, removing any small peaks by using a momentum term from previous time steps (equation 5).

$$F_4(t) = \lambda F_3(t) + (1-\lambda) F_4(t-1), \tag{5}$$

where a value of $\lambda = 0.2$ was used. This means that the filter uses information about the previous value to smooth out the curve. The drawback of this technique is that it also pushes the position of the peak back, so that the robot does not perceive the peak in the place where the readings were taken, but some way further along.

4.2 Selection Method 2: The Kalman filter

The peak detection algorithm discussed in section 4.1 has a major problem. The filters which are needed to clean up the data sufficiently have a number of side effects to do with delaying the appearance of peaks in the data and smearing them across the data.

For this reason, an alternative approach was considered, using a Kalman filter [4] to maintain a model of the error curve and looking for places where the error did not fit the model. The Kalman filter makes continuously updated estimates of the state of a linear process, as well as maintaining an error model of the estimates. By using the filter to keep a model of the error curve, and detecting when the model predictions are poor, we can detect outliers to the data at the current time. These outliers are then selected as landmarks.

The implementation used in this work was a simplified version of the Kalman filter. For a more complete description, see [7]. The Kalman filter maintains an estimate of the covariance of the data (in this simple, one-dimensional case, the variance, v) and uses that to give error bounds to the predictions, using the Kalman gain, K.

At each time step, a number of equations are applied. The first, equation 6, calculates the Kalman gain for the new iteration, based on the estimate of the variance at the previous time step. R is an estimate of the variance in the measurement error, that is, the error which is introduced in the reading of the state. Then, in equation 7, the new estimate of the state of the process, $y(t)$, is calculated. This includes a (noisy) measurement of the process taken at the current time, $x(t)$. Finally, equation 8 shows how the estimate of the variance is updated.

$$K(t) = \frac{v(t-1)}{v(t-1)+R}, \tag{6}$$

$$y(t) = K[x(t) - y(t-1)] + y(t-1), \tag{7}$$

$$v(t) = v(t-1) - Kv(t-1). \tag{8}$$

The value of R, the variance in the measurement noise, and the initial value of v, the variance in the data, were estimated from the training data which was used to train the neural network, using the normal definition of the standard deviation.

The Kalman filter was used to keep a continuously updated estimate of the state of the error curve. The variance, v, also suggests bounds on the error of the prediction. These bounds were used to define landmarks — a landmark was placed wherever the actual observation differed from the prediction by more than four standard deviations.

5 Assessing the Performance of the Landmark Selection Mechanisms

This work investigates three different neural network architectures (the modified Perceptron, the lagged Perceptron and the recurrent Perceptron) for model acquisition, all with and without median filtered inputs, and two methods for selecting landmarks from the error curves of these networks.

In order to be able to evaluate the performance of all these variations, it is therefore necessary to have some method for evaluating performance. Two different assessment methods are introduced in this section. Both work by comparing pairs of runs in one environment.

Where we have 10 test runs in an environment, we compare the 1st run with the 2nd, 3rd and so on, the 2nd with the 3rd and 4th, etc., so that we obtain 45 pairs for comparison ($9+8+7+\ldots+1=45$).

The first evaluation method investigates how well two error curves obtained in different runs match, by calculating the sum-squared difference between the two curves. The second evaluates the consistency of the landmark selections, where a landmark is considered to be consistent if it is selected repeatedly on successive runs.

5.1 Assessment by Sum-of-squares Error

If a trained network is used to predict the sensor perceptions over several different runs in the same environment, we would expect that good performance would mean that the error curves would be similar every time. One way of comparing two curves is to compute the squared distance between the two curves at every point. This gives a measure of the correlation between them. The measure increases rapidly when the distance between the two curves is large, for instance when a spike appears in one data set, but not the other. A peak in only one error curve, of course, means a landmark is selected only in that one runs, and not in the other.

5.2 Assessment by Measuring the Consistency of Landmark Selection

To measure how consistent the landmark selection is, the following method, referred to as LandMark Consistency, LMC, is proposed.

For each run in a particular environment, a vector is made, containing entries for each sensor sample point in the environment. Each element of the vector contains information about whether or not that place was selected as a landmark. If it is, the value $1/\sqrt{2}$ is used, if the place is adjacent to a landmark the value 1/2 is used, and otherwise a 0 is put into the element. This produces a vector with as many elements as there were samples in the environment, a typical section of which may look like

$$(\ldots 0, 0, \frac{1}{2}, \frac{1}{\sqrt{2}}, \frac{1}{2}, 0, 0, \ldots) \tag{9}$$

The scalar product of two such vectors, obtained in two separate runs through the same environment, is then taken as the LMC performance measure. In order to obtain a value between 0 and 1, the scalar product is normalised by dividing it by the total number of landmarks selected (equation 10).

$$LMC(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\max(N_a, N_b)}, \tag{10}$$

where N_a is the number of landmarks in vector **a**, N_b is the number of landmarks in vector **b**, and LMC is the LandMark Consistency performance measure.

The LMC performance measure is designed such that if the number of landmarks is not the same in both runs, the value of LMC is lower than 1. The values $1/\sqrt{2}$ and $1/2$ were chosen so that a perfect match between a landmark in two different runs gives a value of 1, and close misses a value of $1/\sqrt{2}$. The information about a landmark being present was spread across three elements of the vector to allow for error in the measurement of the robot's odometry, which could change the position of the landmark. In the theoretically possible event that two consecutive positions were recognised as landmarks, both landmarks would be marked as $1/\sqrt{2}$. In that case, a perfect match with a second run would result in a LMC performance measure greater than 1. This has not happened in any of the testing.

Using this measure, each comparison between two runs yields a value between 0 and 1, (provided that landmarks never appear in consecutive elements), where a 0 means that no landmarks correlated, and 1 means that all the landmarks matched perfectly. The two possible different overlaps (of two elements and one element) produce values of $1/\sqrt{2}$ and $1/4$ respectively.

6 Experimental Results

The technique of early stopping, discussed in section 3, was used to decide when network training should be terminated. This was done by obtaining data from 10 runs through an environment, and using five of them as training data and the other five runs as validation data. Using this method, it was established that 20 training steps was sufficient; the data of those 10 runs was then discarded.

For the experiments proper, we obtained data through another 20 data collection runs in each of the seven environments. Of these 20, 10 were used as training data, and the remaining 10 as test data.

The model which is learned should be general to the perceptions of the robot, not specific to particular environments. For this reason we experimented with using training data taken from three environments chosen at random, and also with using all the data from the seven environments.

Figure 7 shows the types of error curves which the networks produce on a typical data run. It can be seen that the curves have peaks above a baseline produced by the noise in the system, and that the several data runs have produced curves which sit fairly well on top of each other.

6.1 Evaluation Using the Sum-of-Squares Error

The different trained networks were all evaluated over the 10 test runs in each environment. The sum-squared difference between each pair of curves was then evaluated, leading to 45 $(9 + 8 + \ldots + 1)$ values for each network in each environment. These numbers were averaged to produce an average error for each network in each environment. Figure 8 shows the average, best, and worst error values for each of the network architectures tried. In addition, table 2 shows the mean sum-squared error and standard deviation, averaged across the different test environments.

These results show that the modified Perceptron generally produces the highest errors. This makes sense since the modified Perceptron receives less input than the other networks. Huge errors are caused by places where a landmark is predicted in some runs, but not in others.

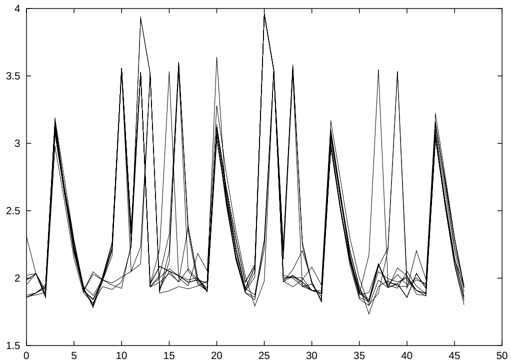

Figure 7: Some typical prediction error curves obtained over several runs in one environment, and superimposed. The way the error periodically rises above a baseline can be seen, as can the way the different curves match well at some points, but less so at others.

Model	Training Environments	Mean Error	Standard Deviation
Modified Perceptron	ABC	16.28	10.61
	ABE	25.62	16.48
	ACD	7.63	4.28
	BDE	25.68	16.45
	DEF	25.61	16.48
	ABCDEFG	25.64	16.47
Recurrent Perceptron	ABC	0.44	0.28
	ABE	0.64	0.36
	ACD	0.40	0.25
	BDE	0.32	0.32
	DEF	0.45	0.29
	ABCDEFG	0.45	0.28
Lagged Perceptron	ABC	0.53	0.36
	ABE	1.21	0.72
	ACD	52.89	33.24
	BDE	1.17	0.75
	DEF	1.21	0.72
	ABCDEFG	1.13	0.73

Table 2: The mean sum-squared error and the standard deviation, averaged over the testing environments, for the different network training algorithms.

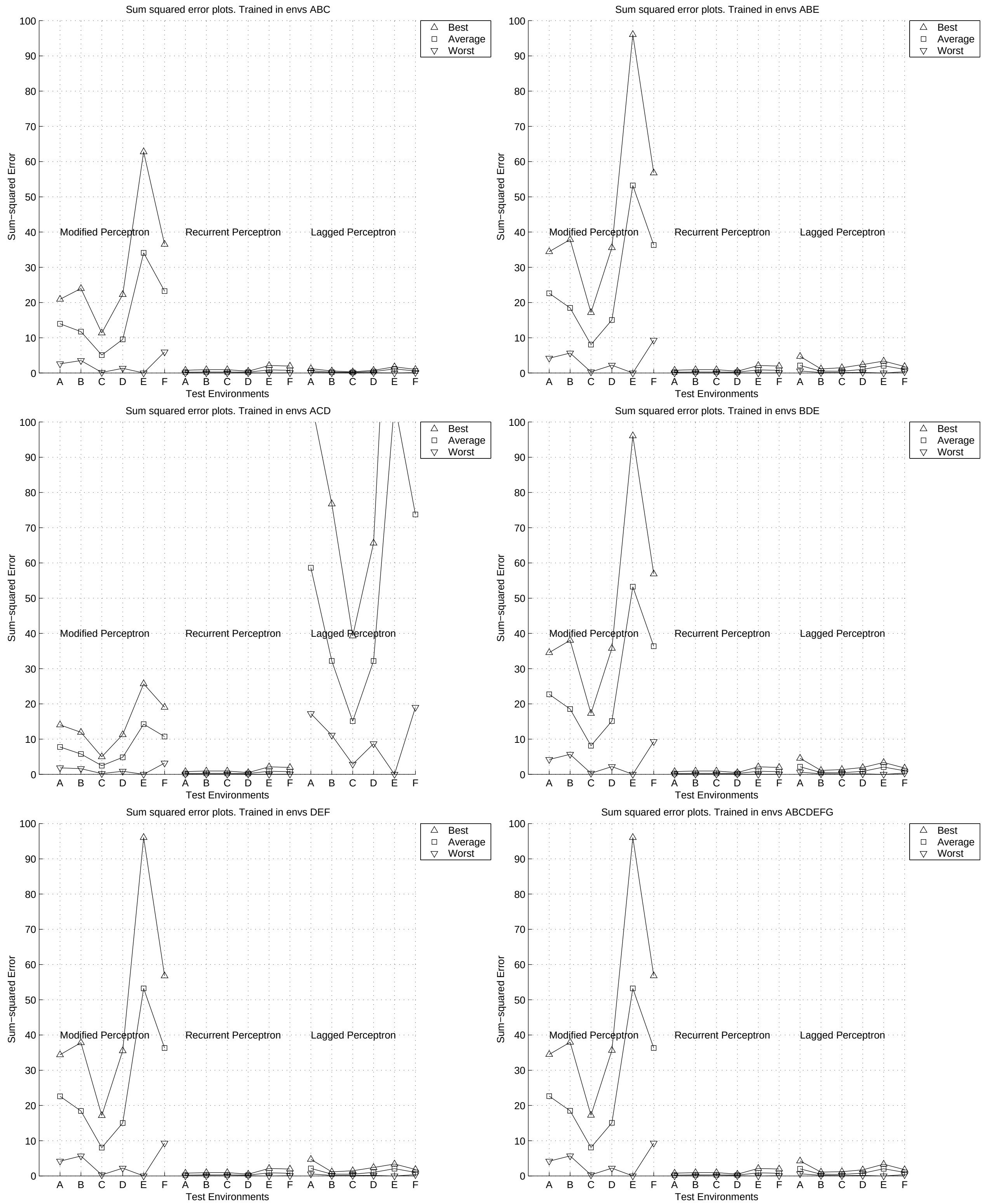

Figure 8: The results of evaluating the sum-of-squares error for each of the different neural networks. The error is shown for each network, tested in each of environments A to F (the six data points for each network). The 6 pictures show the results after training in different sequences of environments (ABC, ABE, ACD, BDE, DEF, ABCDEFG, respectively).

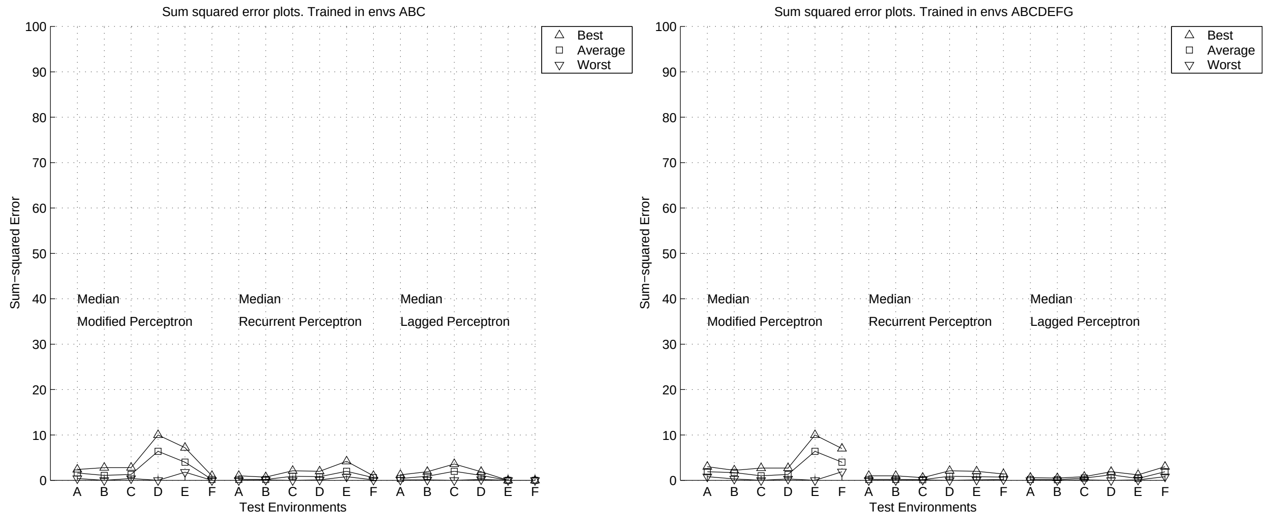

Figure 9: The results of evaluating the sum-of-squares error for the different neural networks with median filter preprocessing. By comparing with figure 8 we can see that median filtering does not make a difference for the lagged and recurrent Perceptrons. The error is shown for each network, tested in each of environments A to F (the six data points for each network). The pictures show the results after training in environments ABC and ABCDEFG respectively.

Model	Training Environments	Mean Error	Standard Deviation
Modified	ABC	2.69	2.11
Perceptron	ABCDEFG	2.42	2.32
Recurrent	ABC	0.51	0.27
Perceptron	ABCDEFG	0.78	0.66
Lagged	ABC	0.73	0.67
Perceptron	ABCDEFG	0.71	0.77

Table 3: The mean sum-squared error and the standard deviation, averaged over the testing environments, for the different network training algorithms with the network inputs (i.e., raw sonar perceptions) median filtered.

Another observation is that in one experiment the error in the lagged Perceptron is large when the error in the modified Perceptron is low ("ACD" in figure 8). This may be because for certain training sets, having information about the previous as well as the current timestep is actually a disadvantage — this point, however, requires further investigation.

By comparing the graph for training in all environments ("ABCDEFG") at the bottom right of figure 8 with the others, it can be seen that the additional training does not make much difference compared to training in only three. A paired Student t-test [9] comparing the sum-squared error after training in environments BDE and in environments ABCDEFG confirmed this ($p = 0.01$).

Figure 9 shows the sum-squared error curves generated when the inputs to the networks are first median filtered, and table 3 gives the mean and standard deviation in the measurements. The results are shown for training in three environments (A, B and C) and for training in all seven environments. One can see that this technique makes a significant difference to the modified Perceptron, but does not really affect the lagged and the recurrent Perceptron models. In fact, the results look slightly worse. These results are confirmed by the paired t-test, which suggests that the median filtered versions of the recurrent and lagged perceptions correlate highly with the non-median filtered versions, while the chance of the results from the modified Perceptron and the modified Perceptron using median filtered input coming from the same distribution is negligible.

Model	Training Environments	Mean LMC Value	Standard Deviation
Perceptron	ABC	0.13	0.04
	ABE	0.14	0.04
	ACD	0.12	0.02
	BDE	0.17	0.03
	DEF	0.13	0.06
	ABCDEFG	0.15	0.04
Recurrent Perceptron	ABC	0.15	0.04
	ABE	0.14	0.05
	ACD	0.14	0.04
	BDE	0.15	0.02
	DEF	0.17	0.03
	ABCDEFG	0.15	0.04
Lagged Perceptron	ABC	0.15	0.02
	ABE	0.14	0.05
	ACD	0.15	0.04
	BDE	0.15	0.04
	DEF	0.15	0.02
	ABCDEFG	0.14	0.04

Table 4: The mean landmark consistency and the standard deviation, averaged over the testing environments, for the peak detection.

6.2 Evaluation Using the LMC Performance Measure

Having compared the correlation between the error curves, we now compare the consistency of landmark selection in the different algorithms, using the LMC performance measure (equation 10). This was done in a very similar way to the previous sum-squared error measurements.

The results are shown in figures 10 and 11 for the three different neural network architectures and the two different methods of landmark selection (peak detection and the Kalman filter). Again, the average case together with the best and worst cases are shown. Tables 4 and 5 show the mean and standard deviation for these results, averaged over all tests.

A comparison between the two figures shows that the performance of the Kalman filter with this measure is significantly better than that of the peak detection. The paired Student t-test was used to compare the peak detection and Kalman filter algorithms for each of the neural network architectures. The results show that the probability of the LMC measures for the two algorithms coming from the same distribution, regardless of neural network used, is negligible.

Looking at figure 10, which shows the LMC measure for the peak detection, we see that while the best case performance is fairly good, in the worst case there are no matches, and the average is also very low. The performance of the algorithm does not appear to vary much between the different network architectures, nor between the different training sets. A paired t-test comparing the different network architectures shows that the performances of the lagged Perceptron and the modified Perceptron are not significantly different ($p = 0.01$), while the performance of the recurrent Perceptron is significantly different ($p = 0.01$).

Figure 11 shows the results of the LMC evaluation for the Kalman filter algorithm. It can be seen that the average landmark consistency is is considerably higher than it is for the peak detection, and the worst case has risen above zero in virtually every case. Interestingly, the Kalman filter technique appears to work best on the modified Perceptron data, which the sum-of-squared error method of the previous section found to be most noisy. Using the paired t-test to compare the effects of the different neural network architectures on the LMC measure with the Kalman filter algorithm shows that the modified Perceptron results come from a different distribution to the the lagged Perceptron and the recurrent Perceptron results.

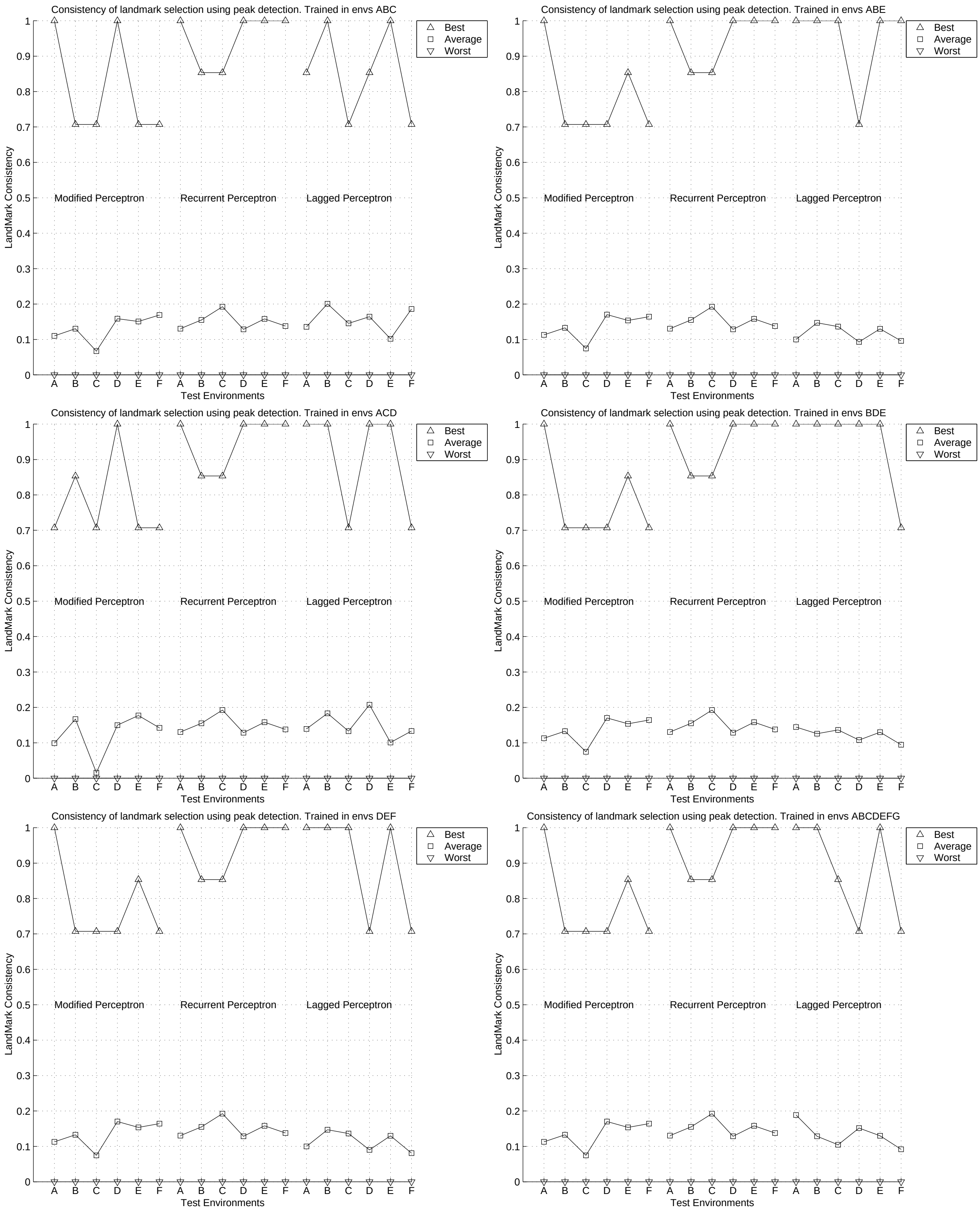

Figure 10: The results of evaluating the consistency of landmark selection (LMC) for the peak detection approach. The LMC is shown for each network, tested in each of environments A to F (the six data points for each network). The 6 pictures show the results after training in different sequences of environments (ABC, ABE, ACD, BDE, DEF, ABCDEFG, respectively).

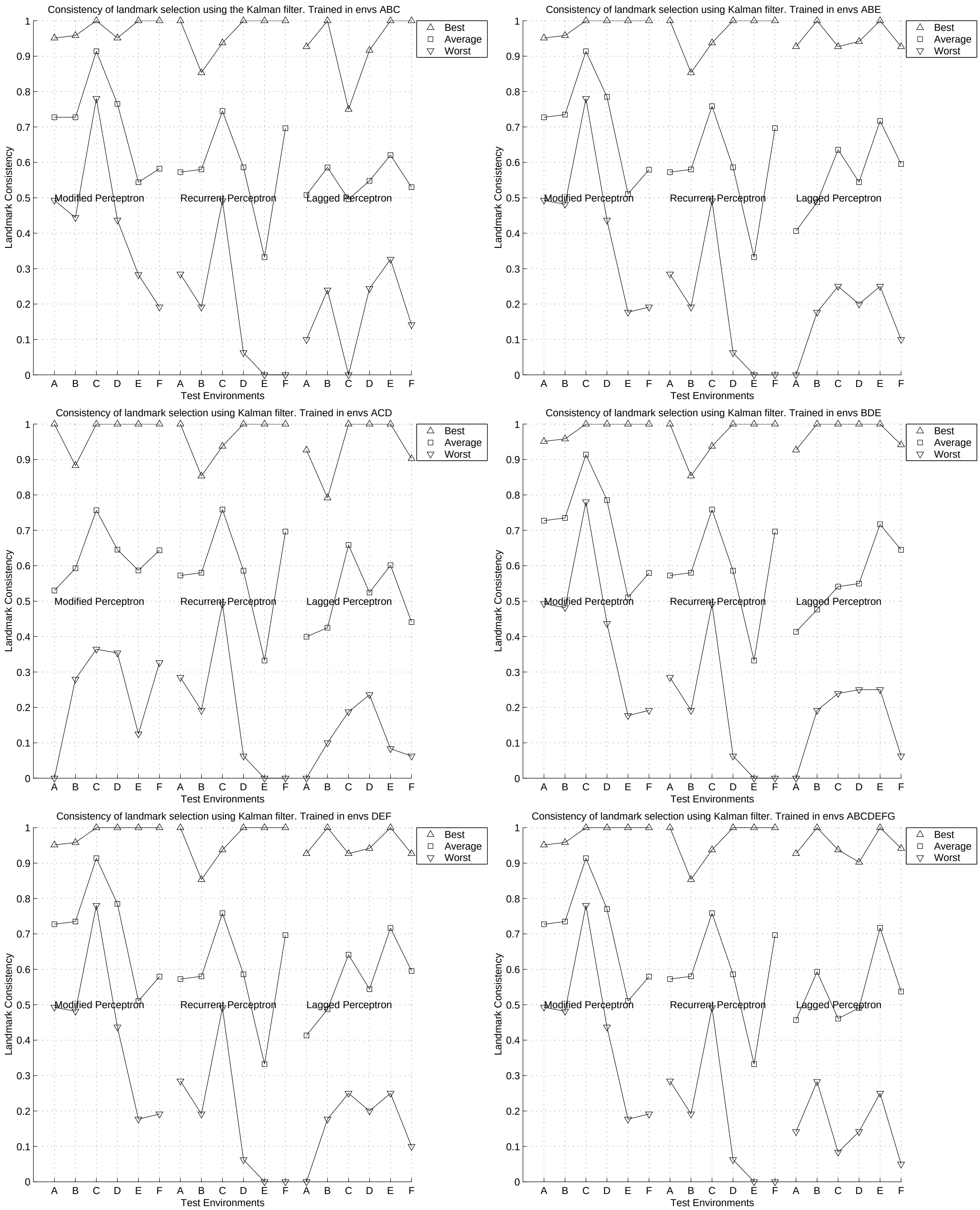

Figure 11: The results of evaluating the consistency of landmark selection (LMC) for the Kalman filter approach. The LMC is shown for each network, tested in each of environments A to F (the six data points for each network). The 6 pictures show the results after training in different sequences of environments (ABC, ABE, ACD, BDE, DEF, ABCDEFG, respectively).

Model	Training Environments	Mean LMC value	Standard Deviation
Modified Perceptron	ABC	0.71	0.13
	ABE	0.71	0.15
	ACD	0.62	0.08
	BDE	0.71	0.14
	DEF	0.71	0.14
	ABCDEFG	0.71	0.13
Recurrent Perceptron	ABC	0.59	0.14
	ABE	0.59	0.15
	ACD	0.61	0.15
	BDE	0.61	0.15
	DEF	0.59	0.15
	ABCDEFG	0.61	0.14
Lagged Perceptron	ABC	0.54	0.05
	ABE	0.56	0.11
	ACD	0.51	0.10
	BDE	0.56	0.11
	DEF	0.56	0.11
	ABCDEFG	0.54	0.09

Table 5: The mean landmark consistency and the standard deviation, averaged over the testing environments, for the Kalman filter.

6.3 Number of Landmarks

Table 6 shows the average number of landmarks which are selected by each of the algorithms for different test sets. It can be seen that the Kalman filter approach finds considerably more landmarks, and that the standard deviation in the number of landmarks found is similar for both algorithms. Although the goal of this project is to *select* (and therefore to limit) landmarks consistently on every run in an environment, there still needs to be a certain number of landmarks to allow self-localisation and navigation. The results shown in table 6 show that the choice of landmark selection method will influence the number of landmarks selected, a property that can be useful for real world robotics applications.

Selection Algorithm	Test Environment	Average Number of Landmarks	Standard Deviation
Peak Detection	A	2.1	0.5
	B	2.5	1.0
	C	1.8	0.7
	D	1.0	0.0
	E	1.4	0.5
	F	1.9	0.6
Kalman Filter	A	4.5	1.1
	B	5.5	0.5
	C	3.4	0.9
	D	5.0	0.0
	E	4.9	0.5
	F	3.4	0.6

Table 6: The average number of landmarks found in each environment by the two different landmark detection algorithms.

7 The Novelty Filter

Another possibility which was considered as a landmark selection mechanism was the use of the online novelty filter described in [6]. This is a self-organising neural network which evaluates the novelty of each input with respect to a model created as the network learns. Since the network builds a model of normality, where normality is defined as things which have been seen several times before, and then detects deviations from this model, we investigated whether this would be useful for this problem.

Evaluation of the network showed that the novelty filter's performance was between the peak detection and the Kalman filter. The principal problem is that after training in an environment, the model of that environment is generally very good and hence nothing in that environment was considered novel. That is, the network learns a very detailed model rather than the very simple and general model that the modified Perceptron learns. Hence the only features that are highlighted as landmarks are those that are seen very infrequently, and thus not recognised by the network.

8 Conclusions

The research presented in this report is a continuation of the work described in [8]. In addition to the technique of detecting peaks in the error curve to select landmarks, a new method using a Kalman filter is proposed, and the two selection methods are compared, using two different types of performance measure.

Three different neural network architectures to model the relationship between subsequent sensor readings are investigated. Performance of the three models, using raw sonar input and median filtered sonar input, is also assessed using the two performance measures.

The results show that the filtering algorithm which is used is particularly important for the consistency of the landmarks that are selected. The LandMark Consistency (LMC) results show that the Kalman filter approach to landmark selection, together with the modified Perceptron as the learning mechanism to generate the model of the data, performs best. The Kalman filter selection method has a second advantage. By changing the value at which a point becomes an outlier (four standard deviations in the current work), it is possible to make landmark selection more or less frequent, thus influencing the number of landmarks selected in a particular environment. This means that the algorithm could be used even in environments where there is very little difference between positions, and still produce landmarks from what data there was.

Before the system can be used, a model of perception has to be acquired by network training. The results of this paper have shown that this is quickly achieved, and that training in a few representative environments is sufficient for good landmark prediction. Once trained, the network weights can be left untouched for all exploration and landmark selection.

Further work will investigate how well the system performs when the perceptions of the robot are taken continually, rather than at 20 cm intervals. They will also examine the effects of the algorithms when the robot does not travel in a straight line parallel to a wall, but follows paths which include turns.

Acknowledgements

We thank Jason Fleischer and Jonathan Shapiro for helpful discussions, and Tom Duckett for many emails discussing the performance measures and algorithms applied. We are grateful to Bernd Krieg-Brückner's group at the Bremen Institute of Safe Systems, who made us welcome, and especially Axel Lankenau, Rolf Müller and Thomas Röfer for assisting in the use of the wheelchair.

The support of the British Council and the German Academic Exchange Service DAAD under ARC grant 878 is gratefully acknowledged.

References

[1] C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, England, 1995.

[2] Tom Duckett and Ulrich Nehmzow. Mobile robot self-localisation and measurement of performance in middle scale environments. *Robotics and Autonomous Systems*, 24(1–2):57–69, 1998.

[3] Tom Duckett and Ulrich Nehmzow. Learning to predict sonar readings for mobile robot landmark selection, 1999. Internal report, University of Manchester.

[4] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, March 1960.

[5] David Kortenkamp and Terry Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelth National Conference on Artificial Intelligence (AAAI'94)*, pages 979–984, Seattle, Washington, 1994.

[6] Stephen Marsland, Ulrich Nehmzow, and Jonathan Shapiro. Novelty detection on a mobile robot using habituation. In *From Animals to Animats: The 6th International Conference on Simulation of Adaptive Behaviour (SAB'2000)*. MIT Press, 2000.

[7] Peter S. Maybeck. The Kalman filter: An introduction to concepts. In I. Cox and P. Wilfong, editors, *AI-based Mobile Robots: Case Studies of Successful Robot Systems*, pages 193–204. Springer, Berlin, 1990.

[8] Ulrich Nehmzow, David Gelder, and Tom Duckett. Automatic selection of landmarks for mobile robot navigation. Technical Report UMCS-00-7-1, Department of Computer Science, University of Manchester, 2000.

[9] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, England, 2nd edition, 1992.

[10] Brian Yamauchi and Randall Beer. Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man and Cybernetics Section B*, 26(3), 1996.