ISSN 1361 - 6161

THE UNIVERSITY of MANCHESTER

Computer Science
University of Manchester

Learning by Tuition, Experiments with the Manchester “FortyTwo”

Andreas Bühlmeier, Holger Dürer, Jürgen Monnerjahn, Manfred Nölte and Ulrich Nehmzow

Department of Computer Science

University of Manchester

Technical Report Series

UMCS-96-1-2

Learning by Tuition, Experiments with the Manchester "FortyTwo"*†

Andreas Bühlmeier‡
andreas@informatik.uni-bremen.de

Holger Dürer‡
duerer@uni-bremen.de

Jürgen Monnerjahn‡
monn@informatik.uni-bremen.de

Manfred Nölte‡
mn@pc-labor.uni-bremen.de

Ulrich Nehmzow§
ulrich@cs.man.ac.uk

January 25, 1996

Recent technical reports issued by the Department of Computer Science, Manchester University, are available by anonymous ftp from `ftp.cs.man.ac.uk` in the directory `pub/TR`. The files are stored as PostScript, in compressed form, with the report number as filename. They can also be obtained on WWW via
URL `http://www.cs.man.ac.uk/csonly/cstechrep/index.html`. Alternatively, all reports are available by post from The Computer Library, Department of Computer Science, The University, Oxford Road, Manchester M13 9PL, UK.

*Refereed by: Alan Murta

†This report also appears as ZKW-Bericht 2/96, Zentrum für Kognitionswissenschaften, Universität Bremen, Germany

‡Universität Bremen / FB 3, Mathematik und Informatik / 28334 Bremen / Germany

§University of Manchester / Department of Computer Science / Oxford Road / Manchester / UK.

Abstract

This report describes the results of a short–term research stay of a group of PhD-students from Bremen University, Germany at the University of Manchester robotics lab. The goal of the research was to investigate the feasibility of teaching direct sensor to motor relations to evolve a complex task-solving behaviour without explicitly programming it. Results and consequences for directions of further research are reported.

1 Introduction

Typically, robots have been programmed to fulfill tasks by explicit commands like "move 10mm in x-direction". Such classical controllers often allow some kind of 'teaching': the operator can steer a robot towards points and label them. The program may then contain commands to go from one 'taught' label to the other.

One problem with this approach is that of generalization. This 'teaching' only provides taught points and the actions between them. In situations which were not taught, this kind of knowledge does not supply a strategy of what to do.

To learn a generalization we need a different understanding of learning—learning as finding a model that fits the taught data and that allows appropriate behaviour in other situations. This will allow the robot to acquire knowledge about the environment and eventually to move about autonomously. It does, however, create the problem of deciding which of a possibly large set of explanations (i.e. generalizations) is the most appropriate. Here prior knowledge about the task and the environment are needed to make this decision.

A second problem with the memorization approach is that of exact positioning and position detection. This is not a big problem with a robot arm and on a mobile robot can also be done using odometry. However, due to inevitable wheel slippage the mobile robot's position estimation will become increasingly incorrect. The estimated position could be recalibrated using reference points, which must be detected by sensors. In normal environments this necessitates a complex algorithm to identify objects or places in the surrounding world and to unify these with the internal model.

On the other hand, sensor readings can be directly transformed into motor commands without explicitly recognizing landmarks or other methods of matching the outer world to the internal model. The choice of how to approximate the function that maps sensor data to actions influences how much and what generalization is achievable. Possible choices are a list of taught points combined with a nearest-neighbour approximation[1], neural nets or any function approximator known in statistics.

The work described here builds on experiments conducted before (Martin & Nehmzow, 1995). The intention is to enhance the quality of existing software and to analyze the system's success and failure.

This report is structured as follows: In section 2 related work is reviewed. Section 3 describes the test tasks selected to investigate the developed mechanisms, which are

[1] As a result, outputs depend on the taught and on similar points.

outlined in sections 4 and 5. The sections 6 and 7 describe the various experiments and their result. We close with a discussion and outlook.

2 Related Work

Supervised learning, i.e. adapting a controller by providing examples of desired input to output mappings, is often used in artificial neural networks, e.g. for classification tasks. The problem of adaptive robot control is very similar: sensory representations of the environment are to be classified not to classes of objects but to actions. Since the back-propagation algorithm (Rumelhardt *et al.*, 1986) is a general tool for supervised learning it is straightforward to use it for robot control, if a teacher is available. Pommerleau (1989) demonstrated the feasibility of this approach, when a multilayer neural network learned to associate preprocessed CCD-camera inputs (32×32 pixels) to a human driver's steering. After learning the network replaced the human driver. The main problem was to achieve generalization of situations the system did not encounter, for example being too close to the road's edges. Therefore, additional training examples were generated from the images by shifting them to the left and right. A training set of 100 exemplars was trained, to prevent the network from forgetting old situations. One interesting result is the extraction of important features, like road edges, in the hidden layer. See Fäustle et al. (1994) for similar work with an indoor mobile robot.

The same principle can be applied to recurrent neural networks (Tani, 1995). By this, ambiguous situations (where different places give similar or the same sensor signals) can be disambiguated by past sensory signals stored in feedback loops. Tani's robot samples 193 examples and learns off-line.

In many cases in mobile robot control a multilayer network is not necessary. Martin and Nehmzow (1995) demonstrated how a Perceptron-like pattern associator can be used for tuition of obstacle avoidance, wall following, box pushing and route learning. Using a rather simple network enables to perform learning directly without the need of storing input to output mappings, providing very fast on-line learning.

Another learning scheme is competitive Hebbian learning as applied by Rao and Fuentes (1995) controlling a simulated mobile robot. In a competitive learning scheme input data is categorized to given output data. During a learning step the input to output mapping is changed only for the current input data. Therefore, no storage of a collection of input-output patterns is needed to prevent the forgetting of old patterns when new ones are learnt, like in a backpropagation setup.

In all cases there is a trade-off between generalization and accuracy. Over-generalization will cause the same action in situation which should be treated differently. If generalization is too weak the system has to be trained explicitly for almost all situations that might occur.

3 The tasks

In the experiments reported here, we used the Nomad 200 mobile robot[2] of the Manchester robotics lab, called 'forty–two' due to its serial number. This robot (see fig. 1) has, in addition to a black-and-white camera, 16 infra-red (IR), 16 ultrasonic (sonar) and 20 touch sensors as well as wheel encoders which integrate the performed actions over time to estimate the current robot position. It is controlled by an intel 486 IBM-compatible computer running DOS using self-developed programs in C. The video data are collected via a VideoBlaster adapter.

Figure 1: The Nomad 200 robot

We considered two different tasks:

3.1 Box–Pushing

Here the robot has to learn to wander around in its environment avoiding walls. When a box is detected by the IR and sonar sensors[3], it is to turn towards it and start pushing it, i.e. move ahead turning only to keep the box from slipping away. This is to go on until one of the bumpers is activated which only occurs when the box has hit a wall (or some other fixed obstacle) and finally gives enough resistance to activate the touch sensors. In that situation, the robot is to turn around, away from the box and to start wandering again.

[2] Nomadic Technologies

[3] Boxes in this case were not tall enough to be detected by the sonar sensors mounted on the upper rim of the robot. By this the system can distinguish between wall (tall obstacles) and boxes (small obstacles).

3.2 Path–Following

In this main task of path–following, the robot has to learn to drive a certain route, which it is shown a few times. This task intrinsically consists of two subtasks:

- generalizing simple behaviour, e.g. avoiding obstacles, and
- memorizing special behaviour, e.g. when to turn where.

Depending on the difficulty of the route, either the location-to-action association, i.e. memorization, or the generalization aspect, e.g. doing wall–following, obstacle avoidance, etc., is more important.

Thus, changing the difficulty of the route, one can assess the generalization and/or memorization behaviour.

4 Preprocessing of sensor data

One aspect of our research was to evaluate what kind of sensors are useful for a more difficult path–following task. We thus evaluated a few combinations that seemed promising.

All sensor inputs were scaled to a value in the range $[0;1]$ and then inserted into a data vector which was handed over to the learning part. One of the input vector's components was formed by the bumper.

Only the 11 frontmost IR- and sonar sensors were used. This left a 'blind spot' of five sensors for each of these modalities in the back, which allowed the operator to walk behind the robot without being detected by the sensors.

The sonar sensors give good readings in a distance range from about 20 cm to 600 cm. To improve the controller's performance we duplicated sonar readings to make a short-range (high values for short distances) and a long-range (high values for long distances) sensor. For the 'short-range sensor' sonar readings were clipped above a distance of 100 cm and then scaled to the $[0;1]$ interval, while the 'long-range sensor' readings were clipped below 80 cm and then scaled. Appendix A.1 gives details on the preprocessing of sonar data.

The IR-sensors have a low resolution (values in range of 0 to 15) and are only intensity based. Their responses mainly depend on the reflection properties of the object in view. White walls are detected at a distance of approximately 70 cm, while black surfaces are only seen when they are closer than approximately 10 cm. Due to these limitations, no preprocessing comparable to the sonar sensors was deemed necessary.

The video data cannot be directly fed into the learning part of the program. Not only is the amount of data too large but also no useful generalization is possible with that data format. Instead some preprocessing is necessary.

We tried two methods[4]:

[4] All code that does video processing is based on the work by Paul Martin (Martin & Nehmzow, 1995) of Manchester University which we would hereby like to acknowledge.

- The first attempt was to detect edges and count the number of vertical edges in a region of view. This was shown by Martin and Nehmzow (1995) to be useful for an obstacle avoidance task in the same setting. However, for our needs it did not prove to add any information.
- Instead a method, which tiles the view into 5×5 subregions and calculates their respective mean gray-level (i.e. brightness), proved to be more useful in this context.

Lastly it turned out that in some situations the robot still could not detect its correct location since the sensor inputs were still ambiguous. Therefore coarse-coded information of the integrated direction from the odometry to form a pseudo-compass was added.

As an alternative means to disambiguate the sensor signals, we also considered various methods of adding a 'history' of previous sensor readings to provide some additional form of context. Lack of time did not permit us to test these ideas intensively.

5 The Network

For the robot control task a neural network for supervised training is needed. The net is to associate sensor inputs with motor outputs.

In earlier works Nehmzow (1994) used a pattern associator network (similar to a perceptron) to perform related tasks (Martin & Nehmzow, 1995). Therefore, our first try was to use such a pattern associator in the robot controller. But this network type turned out not to be the optimal choice for the task because it requires several training steps to learn a specific movement for a certain input, and it tends to forget old patterns when new ones are learnt.

The desired net is to learn an input-output pattern as fast as possible and not forget patterns that could still be useful for certain sensor inputs. The result of our considerations is a network specially designed to match the needs of our control task. The network combines different qualities of competitive learning networks, self-organizing maps (Kohonen, 1982; Kohonen, 1995), and ART-networks (Grossberg, 1976).

The network memory consists of computational units each containing a pair comprised of an input reference (weight) vector $\vec{w}^{in}$ and an output reference vector $\vec{w}^{out}$. Such a pair is regarded as a 'known' pattern to the net. The number of units, i.e. the size of the network increases during training.

5.1 Association

The main task of an already trained network is to associate an output vector $\vec{u}$ with an actual input vector $\vec{v}$. Such an association works as follows:

The net receives an input vector $\vec{v}$. In our application the input vector $\vec{v}$ consists of several preprocessed sensor signals. The net selects the n best matching units of the network memory, i.e. the units containing input reference vectors $\vec{w}^{in}$ most similar to

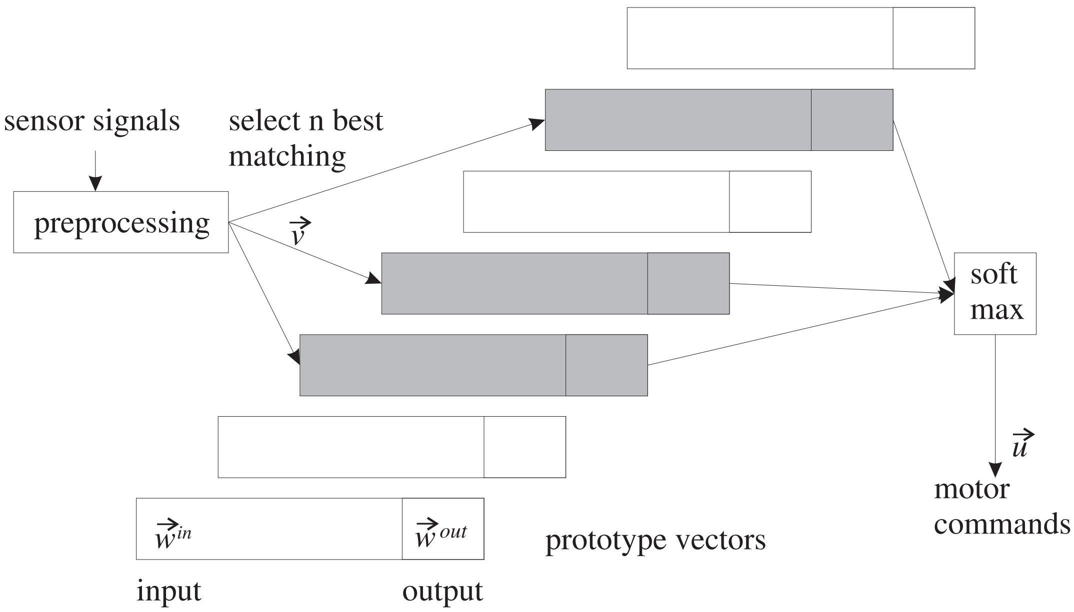

Figure 2: Association in the network: the n best matching units are selected, the output is computed by a 'soft-max' interpolation of their output reference vectors.

the input vector $\vec{v}$ regarding euclidean distance. In this, the network resembles competitive learning paradigms such as Kohonen's self-organizing feature map (Kohonen, 1982; Kohonen, 1995).

The output of the net is calculated by an interpolation based on the output reference vectors of the n (being 3 in our experiments) best matching units such that the contribution of each unit is proportional to the similarity of its input reference vector $\vec{w}^{in}$ to the input pattern $\vec{v}$.

In summary, for each of the n best matching units the output procedure calculates the distance d_i to the input vector $\vec{v}$:

$$d_i := ||\vec{v} - \vec{w}_i^{in}|| \tag{1}$$

With units $1 \ldots n$ being the n best matching ones the output vector $\vec{u}$ is computed by

$$\vec{u} := \frac{1}{t} \sum_{i=1}^{n} \vec{w}_i^{out} \prod_{j=1, j \neq i}^{n} d_j \quad \text{where} \quad t = \sum_{i=1}^{n} \prod_{j=1, j \neq i}^{n} d_j \tag{2}$$

Since the network memory is empty at the beginning and the number of units grows during the learning process, this 'soft-max'-output cannot be applied until the number of units is n or higher. While the number of units is too small, the network output is the output weight vector $\vec{w}_s^{in}$ of the best matching unit s.

5.2 Training

A training step for a new pattern $(\vec{v}, \vec{u})$ of sensor input $\vec{v}$ and motor output $\vec{u}$ is performed as follows:

The algorithm determines the best matching unit s, i.e. the unit with the input weight vector $\vec{w}^{in}$ most similar to the input part $\vec{v}$ of the new pattern regarding euclidean distance. If the distance between these vectors is larger than a certain value r, the new pattern is added to the pattern memory as a new unit. The network grows.

If the distance is smaller than r, the best matching unit s is adapted, i.e. the input and output reference vectors $\vec{w}_s^{in}$ and $\vec{w}_s^{out}$ of this unit are moved in the direction of the new pattern $(\vec{v}, \vec{u})$:

$$\vec{w}_s^{in} := \vec{w}_s^{in} + \epsilon(\vec{v} - \vec{w}_s^{in}) \tag{3}$$

$$\vec{w}_s^{out} := \vec{w}_s^{out} + \epsilon(\vec{u} - \vec{w}_s^{out}) \tag{4}$$

The adaptability of the net is controlled by the learning parameter ϵ. In our application ϵ was 0.5. This parameter did not change during the training process.

The growing of the network can be limited to a maximum number of units. In our application, the network did not generate more than 250 artificial neurons (the limit was 500). The speed of growing depends on the parameter r mentioned above. The smaller r the larger the final network size.

After some informal testing, r was set to $1/5\sqrt{\textit{number of network inputs}}$ for all robot experiments. To compensate for different dimensionalities of the input space the value is dependent on the number of inputs. Otherwise, a different input dimensionality might cause a different generalization behaviour. Alternatively one could scale the input vector to a maximum length.

5.3 The Network's Properties

The network is simple and easily implemented. It learns, associates, and adapts quickly.

By adjusting the parameter r one can easily explore the continuum between full memorization and broad generalization. By adjusting n one can either sharply distinguish or smoothly interpolate between learnt patterns.

Forgetting old patterns when training new ones does not pose a problem in this approach. 'Known' patterns can only be affected if a new situation is very similar to one that has already been stored in memory. Because of this quality our robot controller achieves a stable behaviour.

Obviously, this stability which is beneficial in the current application can be inappropriate if the net is used for tasks that require a memory permanently adapting to an ever changing environment.

Figure 3: A view of the experimental setup.

6 Experimental Setup

6.1 The environment

The experiments were mostly performed in the robotics lab, a room of approximately 10m by 4m which was fitted with boxes to form obstacles detectable by the IR and sonar sensors. Only two thirds of the room were used, but the whole room was visible to the sensors.

Some experiments of the path–following task were also performed in the public hallways of the Computer Science Department. There, orientation is a more challenging task since less information can be derived from the sensors while at the same time a larger area has to be navigated. Most hallways generate similar perceptual stimuli, there are few unique features—to such an extent that even new students and personnel have a hard time to find their way around.

The lack of windows in both the lab and the hallways eased the problem of varying lighting situations, which would have caused more problems with the IR–sensors and the image processing. However, the varying reflection properties of the walls diminished the utility of the IR–sensors due to their properties, described above. The smooth concrete walls reflected the sonar signals well and at times caused mirror reflections making the sonar data also unreliable.

6.2 Testing procedure

The tests can be subdivided into a first group of preliminary tests and a second set of rigidly performed and evaluated tests at the end.

The preliminary tests consist of:

- experiments with the box-pushing task, which were stopped when it became apparent that the test was easily solved but possibly harmful for the robot,
- path–following in the hallway to find the limits of the sensors, preprocessing, and algorithm chosen, and
- some path–following in the lab with various combinations of sensors, to establish what were the most promising combinations for the second group of tests.

6.2.1 The rigid tests

Exactly comparing the performance of two robot systems is impossible when done in the real world. One will never succeed in establishing identical experimental conditions (lighting, battery charges, starting position, etc.). Furthermore, the operator (teacher) will never react exactly the same way in an identical situation. This was made worse in our system when using the video inputs. Due to the longer delay times necessitated by the processing of the video data, steering the robot felt completely different to the operator from a system without the video data.

Nevertheless, for this second group of tests all attempts were made to make experimental conditions as similar as possible by doing them all on the same day with the same teacher and the same strict sequence of maneuvers.

For all learning tasks the IR- and the sonar sensors and the same network were used. Only the use of video data and pseudo-compass were varied resulting in four tests.

Our method of approach was to perform a strict sequence of training, testing, retraining, testing and so on. Each of the four path–following tests had three training phases and three test phases. The first training–phase (A) consisted of three rounds along the path (see fig. 4). The second phase (B) was an additional round of training, emphasizing the situations where problems occurred during the first test. In the last phase (C), 'convergent behaviour' was trained, i.e. actions that lead from slightly off–route to on–route situations. Also, those situations that still lead to errors in phase (B) were re-trained. In this phase two kinds of tests were performed: (C') re-tests the problematic situations while (C") tests the convergent behaviour by starting from a few selected off-route situations.

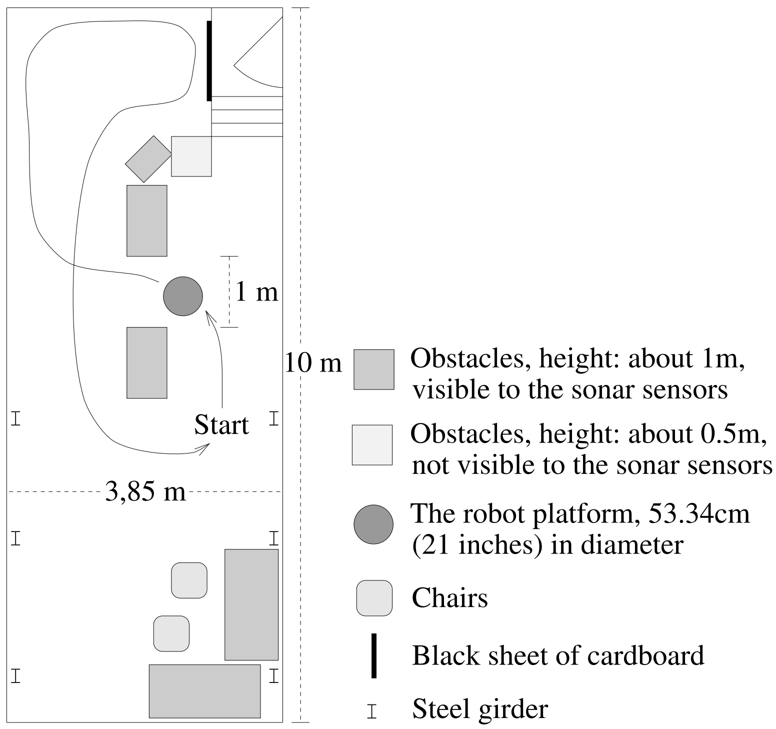

Figure 4: The robot's path in the lab environment.

Some variations to this procedure were made. When it was clear, that a test would fail because previous, easier tasks had failed it was not continued. When the subjective impression was that one test was unusually unsuccessful compared to earlier trials, a second attempt was made to verify the result.

7 Experimental Results

7.1 Box–Pushing

Since it takes quite some pressure to activate the touch sensors it was difficult to train the robot to turn fast enough before the box was virtually destroyed. After some informal tests showed that this task could surprisingly easily be trained we decided to concentrate on the path–following task.

7.2 Path following in the hallways

Some less rigidly designed experiments were done in the hallways around the robotics laboratory. As described above, the conditions in the hallways were more difficult than in the lab itself.

For testing we elected a complicated route which included navigating in open territory (i.e. a larger empty space much unlike the narrow hallways), one intersection, and four other turning points. It was designed to be a round course, so the robot had to turn around after the first half and drive the route back. Figure 5 shows the path.

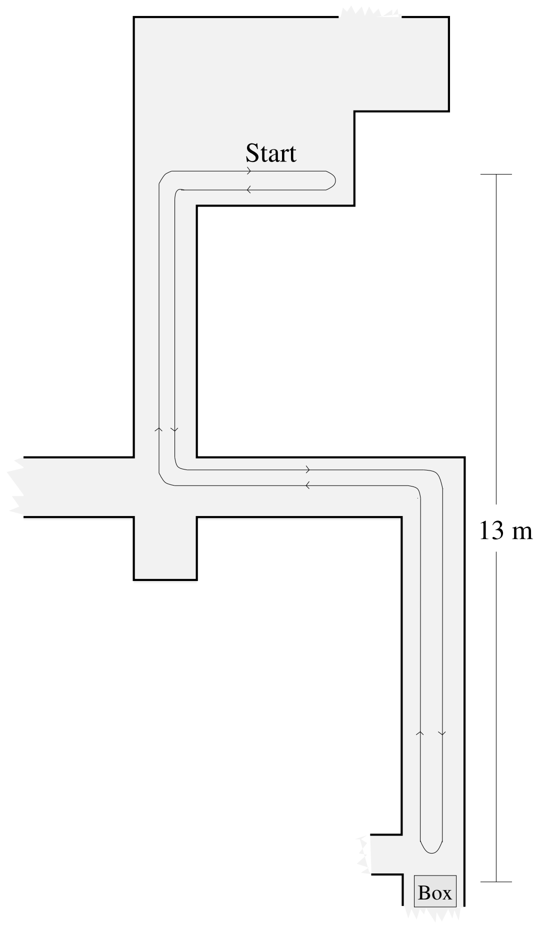

Figure 5: The robot's path in the hallway.

The most difficult part turned out to be this turn-around point. It was indicated by an open door in the hallway, in front of which the robot was supposed to turn around. Due to limited sensor resolution (and lack of a sensor history), the open door was only detected in one out of three cases. When additional cues in form of a box in the corridor were present, detection of the turning point became considerably more successful.

Due to the chosen slow movements of the robot, one whole round lasted about 10–15 minutes. (A training round is a bit faster than a recall round, due to some implementation decisions in the controller which favour security over speed.)

Two to three training rounds were necessary to acquire a basic maneuvering capability. This behaviour could be improved when extra attention was paid to difficult locations, i.e.

Table 1: NUMBER OF ERRORS ON THE RIGID TESTS

EXPERIMENT	TEST A	TEST B	TEST C'	TEST C"
IR, sonar	1	1 ; 0	-	1
IR, sonar, compass	0	1 ; 1	0 ; 0	0
IR, sonar, video	2	1	0 ; 1	-
IR, sonar, compass, video	2	1	1 ; 0	0

the robot was shown these situations more often with varying slight changes of orientation and positioning relative to the walls.

However, the overall performance was still disappointing. At no time did the robot manage to complete one full round without interference from the operator. It appears that the sensor preprocessing was still not sufficiently elaborate to disambiguate the more difficult situations. Also, generalization of common behaviour (in this case: avoiding walls) was suboptimal—here the controller seemed to distinguish too many situations.

Our conclusion is that more experiments with other preprocessing (especially some history of previous sensor readings) are necessary to complete this task. Due to the long training times (one battery charge lasts for approximately 3 rounds) we decided to restrict ourself to the lab where we replicated some of the difficult situations encountered in the hallway.

7.3 Path following results

For the path-following experiments, the robot was trained on the path shown in figure 6. Results on the four tests are shown in table 1 which states the number of errors during the test. Hitting an obstacle or taking a wrong turn were counted as errors, as were interventions to avert such a situation. When two rounds of tests were made, both figures are displayed separated by a semicolon. A dash states, that no test was made.

The winner of this contest is the version with the pseudo–compass but without the video–data. Not only did it have the least number of errors, but the overall subjective performance was also considered to be the best. The winner configuration had 2 errors during six rounds. All other configurations had more errors in four or five rounds. The worst configuration, with IR, sonar, and video, had four errors during four rounds. This, and the "IR, sonar" configuration, i.e. those without the pseudo–compass, were the only configurations that made errors of turning in the wrong direction. This is due to ambiguous situations which can be disambiguated with the compass (see figure 6).

As one can see in figure 6 there are two clusters of errors. One is right next to the stairs in the top right-hand corner of the diagram, the other is at the junction, where all "false turning" errors occurred. There are two reasons for the cluster at the stairs. A black sheet of cardboard had been affixed to the wall next to the stairs and a low box being invisible to the sonar sensors was placed there. The IR-sensors are only intensity based. Therefore the black cardboard was only seen when the robot was closer than

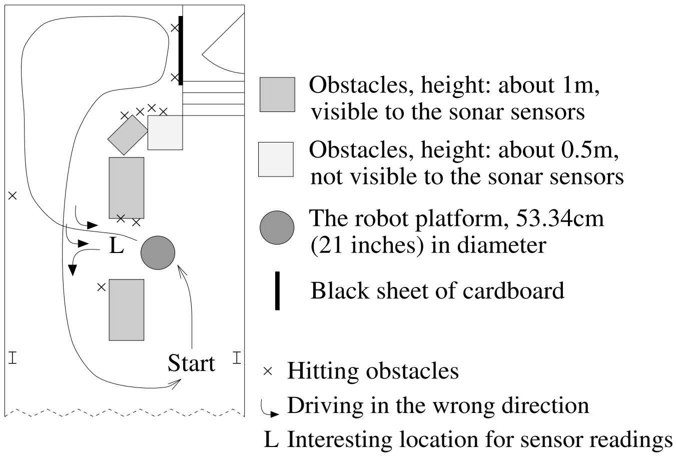

Figure 6: The positions where errors occured

approximately 10 cm.

The two collisions at the second cluster can only be explained by bad generalization. These were not front hits but touching the box with the side, due to turning too late or too slowly. The controller simply had not learned obstacle avoidance, but *memorized* when to do so. Re-training these situations and teaching the 'convergent behaviour' helped to avoid these errors in later phases.

Table 2 lists the number of units each configuration created during the three training phases. The growing of the network was limited to 500 units, but the highest number of units used was 248, generated by the version with the pseudo–compass and the video data. There is a direct correlation between size of input vector and number of units generated.

Table 2: Number of units allocated

Experiment	Number of units
IR, sonar	200
IR, sonar, compass	226
IR, sonar, video	204
IR, sonar, compass, video	248

7.3.1 Analysis of representative vectors

During training the network collects the $\vec{w}_i^{in}$ and $\vec{w}_i^{out}$ vectors. These are thought to be representative sensor–action tuples. It is interesting to analyse, what representative vectors have been chosen by the network. This will help to further understand how to improve the performance by changing parameters, preprocessing or even by deciding what sensors deliver suitable data. Here, we will concentrate on input vectors.

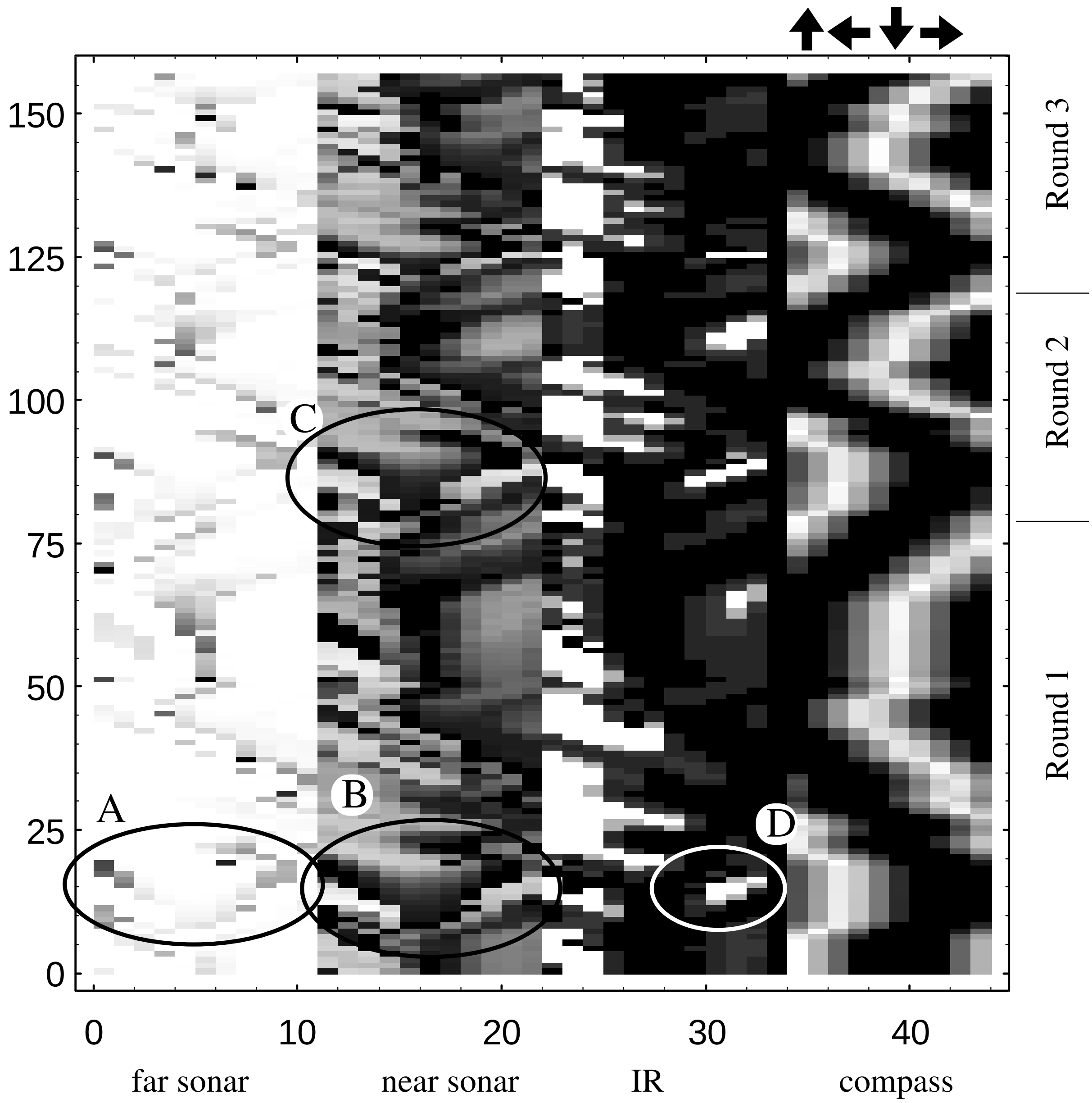

Figure 7: Sensor inputs: sonar (far, near), infra red, bumper and compass

In figure 7 the input vectors of the network memory after the first round of training (training the path three times) with just sonar, IR, bumper and compass information are shown.

One line represents one memory entry, each column one value in the input vector to the net. The columns are logically and visually divided into four main groups for the four sensor kinds: 11 entries for the far sonar sensor, shown inverted for display purposes, 11 entries for the near sonar sensor, 11 entries for the IR sensor (in all these three sensor modalities, dark areas represent far objects, light areas near objects), and the 10 entries for the compass (coarse-coded). The one black column between the IR and compass data shows the bumper readings which were always 0 (no contact).

The order of recording of the vectors is maintained, i.e. the lowest row represents the first vector collected, the upper row the last one. Since only little adaptation was done until this snapshot, each shown vector still approximates the sensor readings of the time it was added to the network's memory.

The 'visual' sensors (IR and sonar) are grouped as if the robot looked ahead, i.e. the leftmost column represents the sensor to the very left, the center column the sensor straight ahead, the rightmost column the sensor to the very right.

For the coarse-coded compass reading, the position of the light region indicates the robot's direction. A bright marker on the very left edge indicates an 'up'-direction with regard to figure 4, next is the 'left'-direction, a marker in the middle indicates a 'down'-direction and lastly further right is the 'right'-direction. A marker on the very right is again an almost 'up'-direction. (See upper edge for indicators.)

A few things are easily visible:

- The far sonar sensors are mostly white (after inversion), indicating that only rarely, far objects were detected. This suggests that the representation range of this sensor has not been fully used. This implies that either the splitting distance between near and far sensors should be lowered, or the far sensor should not be used at all. However, in some situations this sensor might help to distinguish situations. One example are the sensor readings obtained around the location marked 'L' in figure 6, where the robot has just passed the narrow gap and entered the 'corridor'. In the far sonar sensor (marked 'A' in figure 7), but not in the near sensor (marked 'B' in figure 7), one can see that the distance to the left is longer than to the right.

- The periodical pattern of the compass data can be noted, by which one can clearly follow the motion of the robot along its path. Here, the first three rounds, each of which lasted between 1'30 and 3 minutes, can be separated since each round starts with high activity in the left column (up-direction in figure 4). The frequency of the pattern increases, i.e. the later rounds seem compressed, since more and more the robot successfully recalls its memory and storing new patterns is needed less often.

Closer inspection of the data shows that at a few spots the sensor data seem to be strongly correlated (e.g. the V-shaped pattern at marks A and B). However, generally this seems not to be the case as calculated correlations between corresponding far and near sonar sensors or near sonar and IR sensors reveal. The correlation between sensors looking in the same direction is about 0.5. This is likely due to (a) the clipping of data in the sonar preprocessing and (b) the different ranges and detection heights of sonar and IR and the non-linear processing of data (see appendix A).

However some other patterns are clearly visible:

- Angled light lines especially in the IR group (e.g. mark D) indicate an object being passed. At first it is visible to the frontmost sensors (i.e. the center columns) then, with time, it passed the robot and is detected by the sensors more to the left or right (i.e. the left or right columns).

- Dark angled lines indicate passing a "hallway" to that side and are caused by the same effects as the light lines.

- 'V'-shaped lines (e.g. marks A, B, and C) indicate passing the narrow region between the cartons. First the light V-shaped lines indicate the boxes, then follow dark V-shaped lines indication the wide "hallway" following where nothing can be detected to either side.

When also collecting video data as shown in figure 8, one can see other weak correlations. The video data are inserted between the IR sensor and the compass. The data of the 5×5 matrix are rowwise concatenated beginning with the top row and ending with the bottom row.

In order to cancel the effect of global brightness changes, we normalized the video data thus that the 'brightest' element had value 1 and the 'darkest' 0. Intermediate values were linearly interpolated. By this, a totally bright view results in the same data a completely black one does. Structure but not brightness or contrast in an image is maintained regardless of changes in global lighting conditions. Unfortunately this scheme does not allow comparing brightness or contrast of different lines in figure 8, which makes the interpretation of data more challenging.

Regarding the visual data, the following observations can be made:

- The lower rows (i.e. those further right in the figure) tend to be darker, because typically they showed the floor unless the robot stood close to a carton. The uppermost row is usually very light, indicating the white walls most of the time.

- Dark regions correlate with center compass positions, i.e. the robot driving down wrt. figure 4. Thus, the camera only saw the floor since the walls were too far off (the camera was tilted downwards by approximately $40°$). Only the lighter boxes to the left could be seen (marks E) while the lower parts are uniformly dark (mark F).

- The dark cardboard can be identified due to sharp differences in brightness (cf. mark G).

- During the passage through the narrow gap a dark object to the left can be seen (mark H). This is a wooden door that sharply contrasts with the white walls.

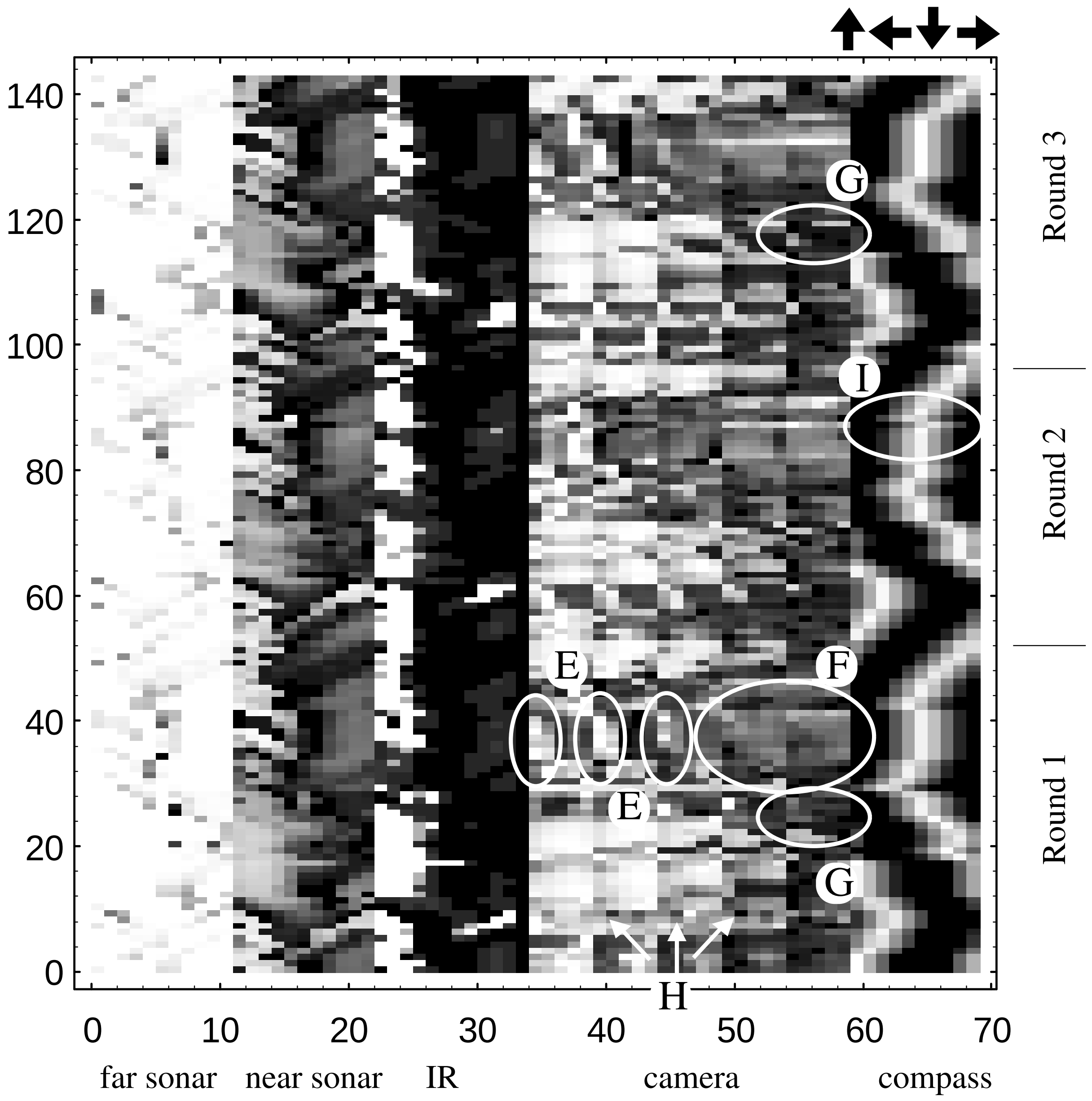

Figure 8: Sensor inputs: sonar (far, near), infra red, camera, and compass

- At one point the robot took a wrong turn and the operator was too slow to intercept this motion early. Two 'off-course' points (mark I) were added while trying to get back onto the normal route.

Obviously, there is a correlation between the robot's orientation and the camera's data which might indicate that the latter could be used to replace the former. However, the odometer information contained precisely what was needed for our testing procedure, learning to turn left into the cardbox passage in the beginning and to keep going ahead when returning. For a different setup the video data might be more helpful compared to the compass which has its problems with loss of precision during long-time operations.

8 Discussion and Future Work

The main advantages of the learning approach discussed here are the adaptation speed, low computational complexity and robustness of the algorithm. We believe that this is an important consideration for real world applications. By adapting the parameter r (see section 5.2) one can explore the continuum between full memorization and broad generalization, thus determining experimentally the best combination of the two for the task at hand. However, some experimentation is needed to set this parameter correctly. Certain errors during the tests indicate that more generalization would have helped. No such experimentation with the parameter was done due to lack of time.

Another drawback of our approach is the non-adaptive preprocessing and the fact that we did not provide past signals to the controller. We believe that this prevented us from solving the path–following task in the hallway. No thorough investigations as to what sensor signals are relevant were performed, yet this would have helped to automate the selection of which sensors and preprocessing methods to use.

Further work needs to be done regarding the problem of adequate sensor signal preprocessing, as well as suitable ways of incorporating past sensor signals (i.e. taking temporal aspects into consideration). Moreover, gathering information in an unsupervised way, e.g. for self-localization (estimation of the robot's position, see, e.g. Nehmzow and Smithers (1991), Bühlmeier and Herrling (1995) might be helpful.

Another conclusion we draw from these experiments is the importance of the man-machine-interface. The operator's training signals must be recognized at any moment and must be associated to the appropriate sensor data. Thus, especially when including high dimensional input like video, very fast soft- and hardware or at least some form of pseudoparallel processing is required.

Acknowledgements

We wish to thank the University of Manchester for supporting our experiments. Travelling and accommodation funds were provided by the DFG and the Freie Hansestadt Bremen through the PhD study program "Raumorientierung und Handlungsorganisation Autonomer Systeme".

References

Bühlmeier, A., & Herrling, U. 1995. An Approach to Adaptive Self-Localization. *Pages 469–474 of:* Fogelman-Soulié, F., & Gallinari, P. (eds), *International Conference on Artificial Neural Networks (ICANN)*.

Fäustle, P., Daxwanger, W., & Schmidt, G. 1994. Steuerung lokaler Fahrmanöver durch direkte Kopplung abbildender Sensorik an ein künstliches neuronales Netz. *In:* Levi, Paul, & Bräunl, Thomas (eds), *Autonome Mobile Systeme*. Informatik aktuell. Springer Verlag.

Grossberg, S. 1976. Adaptive Pattern Classification and Universal Pattern Recoding: I. Parallel Development and Coding of Neural Feature Detectors. *Biological Cybernetics*, **23**, 121–134.

Kohonen, T. 1982. Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, **43**, 59–69.

Kohonen, T. 1995. *Self-Organizing Maps*. Springer Verlag.

Martin, P., & Nehmzow, U. 1995. "Programming" by Teaching: Neural Network Control in the Manchester Mobile Robot. *In: Intelligent Autonomous Vehicles 95*. Pergamon Press.

Nehmzow, U. 1994. *Autonomous Acquisition of Sensor-Motor Couplings in Robots*. Tech. rept. UMCS-94-11-1. University of Manchester, UK.

Nehmzow, U., & Smithers, T. 1991. Mapbuilding Using Self-Organising Networks in "Really Useful Robots". *Pages 152–159 of:* Meyer, J. A., & Wilson, S. (eds), *From Animals to Animats*. MIT Press.

Pommerleau, D. A. 1989. Neural Network Based Navigation. *Chap. 5 of:* Thorpe, Charles E. (ed), *Vision and Navigation-The Carnegie Mellon Navlab*. Kluwer Academic Publishers.

Rao, R.P.N., & Fuentes, O. 1995. Perceptual Homing by an Autonomous Mobile Robot Using Sparse Self-Organizing Sensory-Motor Maps. *In: World Congress on Neural Networks*.

Rumelhardt, D.E., Hinton, G.E., & Williams, R. J. 1986. Learning Internal Represantations by Back-Propagating Errors. *Chap. 8 of: Parallel Distributed Processing*, vol. 1. MIT Press.

Tani, J. 1995. Essential Dynamical Structure in Learnable Autonomous Robots. *Pages 721–732 of:* Morán, F., Moreno, A., Merelo, J. J., & Chacón, P. (eds), *Advances in Artificial Life*. Lecture Notes in Artificial Intelligence. Springer Verlag.

A Preprocessing Details and Parameters

A.1 Sonar

Sonar distance measurements s were computed to form *far* signals i by

$$i(s) = \left(\frac{s - d_{min}}{d_{max} - d_{min}}\right)^2 \tag{5}$$

where s was clipped to be in the range of d_{min} ($\approx$ 60 inches) and d_{max} ($\approx$ 200 inches).

Sonar distance measurements s were computed to form *near* signals i by

$$i(s) = \left(1 - \frac{s - d_{min}}{d_{max} - d_{min}}\right)^2 \tag{6}$$

where s was clipped to be in the range of d_{min} ($\approx$ 5 inches) and d_{max} ($\approx$ 100 inches).

A.2 Infra-Red

IR distance measurements s were computed to form *near* signals i by

$$i(s) = \left(1 - \frac{s}{d_{max}}\right)^2 \tag{7}$$

with d_{max} set to 15.

A.3 Odometer

The odometer based pseudo compass was used by computing for each unit x of N inputs

$$i_x(o) = 2\max\left(1 - \frac{\min(|x-\alpha|, |\alpha - x|)}{rad}, 0\right) \tag{8}$$

with $rad = 3$, we used $N = 10$ units to code the orientation $\alpha = No/360$, where o is the system's orientation in degrees.

A.4 Vision

A subview of 260×240 pixels was averaged into 5×5 image areas, obtaining 25 intensities $(i_1, \ldots, i_{25})$ for the input vector. In order to cancel the effect of global brightness changes, resulting values $v(i_x)$ were normalized, using the minimum i_{min} and the maximum i_{max} of all intensities, thus that the 'brightest' element had value 1 and the 'darkest' 0:

$$v(i_x) = \frac{i_x - i_{min}}{i_{max} - i_{min}} \tag{9}$$