Robot Training using System Identification

O Akanyeti[#], U Nehmzow[*], S A Billings

#Dept Computer Science, University of Essex
***Dept Computer Science, University of Ulster, Ireland**

Department of Automatic Control and Systems Engineering
The University of Sheffield, Sheffield, S1 3JD, UK

Research Report No. 981

July 2008

Robot Training using System Identification

O. Akanyeti[1], U. Nehmzow[1] and S.A. Billings[2]
[1] *Department of Computing and Electronic Systems, University of Essex, UK.*
[2] *Department of Automatic Control and Systems Engineering, University of Sheffield, UK.*

Abstract

This paper focuses on developing a formal, theory-based design methodology to generate transparent robot control programs using mathematical functions. The research finds its theoretical roots in robot training and system identification techniques such as Armax (Auto-Regressive Moving Average models with eXogenous inputs) and Narmax (Non-linear Armax). These techniques produce linear and non-linear polynomial functions that model the relationship between a robot's sensor perception and motor response.

The main benefits of the proposed design methodology, compared to the traditional robot programming techniques are: (i) It is a fast and efficient way of generating robot control code, (ii) The generated robot control programs are transparent mathematical functions that can be used to form hypotheses and theoretical analyses of robot behaviour, and (iii) It requires very little explicit knowledge of robot programming where end-users/programmers who do not have any specialised robot programming skills can nevertheless generate task-achieving sensor-motor couplings.

The nature of this research is concerned with obtaining sensor-motor couplings, be it through human demonstration via the robot, direct human demonstration, or other means. The viability of our methodology has been demonstrated by teaching various mobile robots different sensor-motor tasks such as wall following, corridor passing, door traversal and route learning.

1. Introduction

Fundementally, the behaviour of a robot is influenced by three components: i) the robot's hardware, ii) the program it is executing, and iii) the environment the robot is operating in. Because this is a highly complex and often non-linear system, most of the existing robot programming techniques are based on empirical trial-and-error process where the programmer refines the code iteratively until the robot's behaviour resembles the desired one to a tolerable degree of accuracy [Nehmzow, 2003]. We believe that this approach has certain disadvantages:

(i) The code generation process is costly, time consuming and error prone [Iglesias et al., 2005].
(ii) There is a lack of understanding in robot-environment interaction. The generated controllers may achieve driving the robot in a desired way but they don't tell us about the underlying rules which characterize the relationship between the robot and the environment.
(iii) The generated controllers are usually slow in execution and consume large memory space because of their complex nature. Also, error debugging in thousands lines of code is tiring and cumbersome.
(iv) Due to the iterative refinement, generated controllers are highly robot platform-dependent, which makes them almost impossible to be used in different platforms.
(v) Above all, in the future, we believe that giving an opportunity to people to program their own robots for their individual needs and preferences, rather than pre-programming robots for them will advance robotics research one step further towards personalised robotics. However the today's robot programming techniques require specialised technical skills from different disciplines and it is not reasonable to expect end-users to have these skills.

In this paper we therefore focus on developing a formal, theory based design methodology to generate transparent robot control programs using mathematical functions. The research finds its theoretical roots in robot training and system identification techniques such as Armax (Auto-Regressive Moving Average models with eXogenous inputs) and Narmax (Non-linear Armax).

The main benefits of the proposed design methodology, compared to the traditional robot programming techniques are: (i) It is a fast and efficient way of generating robot control code, (ii) The generated robot control programs are transparent mathematical functions that can be used to form hypotheses and theoretical analyses of robot behaviour, and (iii) It requires very little explicit knowledge of

robot programming where end-users/programmers, who do not have any specialised robot programming skills, can nevertheless generate task-achieving sensor-motor couplings.

In [Nehmzow et al., 2005], [Kyriacou et al., 2006] and [Akanyeti et al., 2007a] we presented a new learning paradigm where the programmer drives the robot manually using a joystick to demonstrate the desired behaviour in a target environment. Once the training data is acquired in this way, we use the NARMAX modelling approach to obtain a model which identifies a coupling between sensory perception and motor response as a linear/nonlinear polynomial. This model is then used to control the robot autonomously.

In [Nehmzow et al., 2007b], [Akanyeti et al., 2007b] and [Nehmzow et al., 2007a] we then introduced a new mechanism to program robots — programming by demonstration — based on algorithmically translating observed human behaviours into robot control code, using transparent system identification techniques. To obtain such sensor-motor controllers, we first demonstrate the desired motion to the robot by walking in the target environment. Using this demonstration, we obtain recurrent, sensor free models that allow the robot to follow the same trajectory (blindly). During this motion the robot logs its own perception-action pairs, which are subsequently used as training data for the Narmax modelling approach that determines the final, sensor-based models which identify the coupling between sensory perception and motor responses as non linear polynomials. These models are then used to control the robot.

This paper summarizes the two robot training paradigms mentioned above and presents results from real experiments. The investigated robot tasks are wall following, corridor following, door traversal and route learning.

1.1. *Related Work*

Robot training is not new in the robotics field. In the 1990s Dean Pomerleau developed the ALVINN system, which learns how to steer a vehicle by observing a human driver steer the vehicle for few minutes [Pomerleau, 1993]. In [Nguyen and Widrow, 1990], Nguyen showed that neural networks can be used to solve highly non-linear control problems: a two-layer neural network containing 26 adaptive neural elements learned to back up a computer-simulated trailer truck to a loading dock, even when initially "jack-knifed".

Also in the 1990s, Nehmzow used a neural network controller to train the mobile robot *FortyTwo* to perform a variety of different tasks such as obstacle avoidance, wall following and route learning [Nehmzow, 1995]. These sensor-motor competences were accomplished by simply retraining the robot without the need to alter the actual robot control code.

Programming mobile robots by demonstration is now a major trend in the robotics community [Pardowitz et al., 2007,Allissandrakis et al., 2005]. Many researchers demonstrate the viability of this approach in tasks such as maze navigation [Hayes and Demiris, 1994] [Demiris and Hayes, 1996], arm movement [Schaal, 1997] [Calinon and Billard, 2007] or service robotics [Demiris and Johnson, 2003].

Most research based on teaching a desired behaviour to a robot — via robot training and programming by demonstration — relies on neural network structures to link perception to action. This method is relatively fast, and generalizes well, but has the main disadvantage of using opaque mechanisms which do not reveal how the desired behaviour is achieved, using the robot's perception.

In the research presented in this paper we therefore aim to combine robot training and programming by demonstration techniques with system identification methods in order to generate *transparent*, mathematically analysable sensor-motor couplings so that we can investigate the underlying rules which govern robot-environment interaction.

2. Methodology

2.1. *Narmax Modelling Methodology*

The NARMAX modelling approach is a parameter estimation methodology for identifying both the important model terms and the parameters of unknown nonlinear dynamic systems. For multiple input, single output noiseless systems this model takes the form:

$$
\begin{aligned}
y(n) = f(&u_1(n), u_1(n-1), u_1(n-2), \cdots, u_1(n-N_u),\\
&u_1(n)^2, u_1(n-1)^2, u_1(n-2)^2, \cdots, u_1(n-N_u)^2,\\
&\cdots,\\
&u_1(n)^l, u_1(n-1)^l, u_1(n-2)^l, \cdots, u_1(n-N_u)^l,\\
&u_2(n), u_2(n-1), u_2(n-2), \cdots, u_2(n-N_u),\\
&u_2(n)^2, u_2(n-1)^2, u_2(n-2)^2, \cdots, u_2(n-N_u)^2,\\
&\cdots,\\
&u_2(n)^l, u_2(n-1)^l, u_2(n-2)^l, \cdots, u_2(n-N_u)^l,\\
&\cdots,\\
&\cdots,\\
&u_d(n), u_d(n-1), u_d(n-2), \cdots, u_d(n-N_u),\\
&u_d(n)^2, u_d(n-1)^2, u_d(n-2)^2, \cdots, u_d(n-N_u)^2,\\
&\cdots,\\
&u_d(n)^l, u_d(n-1)^l, u_d(n-2)^l, \cdots, u_d(n-N_u)^l,\\
&y(n-1), y(n-2), \cdots, y(n-N_y),\\
&y(n-1)^2, y(n-2)^2, \cdots, y(n-N_y)^2,\\
&\cdots,\\
&y(n-1)^l, y(n-2)^l, \cdots, y(n-N_y)^l)
\end{aligned}
$$

where $y(n)$ and $\boldsymbol{u}(n)$ are the sampled output and input signals at time n respectively, N_y and N_u are the regression orders of the output and input respectively, d is the

dimension of the input vector and l is the degree of the polynomial. $f()$ is a non-linear function and here taken to be a polynomial expansion of its arguments. Expansions such as multi-resolution wavelets or Bernstein coefficients can be used as an alternative to the polynomial expansions considered in this study.

The first step towards modelling a particular system using a NARMAX model structure is to select appropriate inputs $\boldsymbol{u}(n)$ and the output $y(n)$. The general rule in choosing suitable inputs and outputs is that there must be a causal relationship between the input signals and the output response, usually ascertained by computing the correlation between chosen inputs and outputs.

After the choice of suitable inputs and outputs, the NARMAX methodology breaks the modelling problem into the following steps:

(i) Polynomial model structure detection: During this step we determine the linear and non-linear combinations of inputs and outputs to detect the significant model terms.

(ii) Model parameter estimation: Then we estimate the coefficients of each term found in the polynomial using an orthogonal parameter estimation algorithm([Korenberg et al., 1988]).

(iii) Model validation: Finally we measure the prediction error of the obtained model.

The last two steps are performed iteratively (until the model estimation error is minimised) using two sets of collected data: (a) the *estimation* and (b) the *validation* data set. Usually a single set that is collected in one long session is split in half and used for this purpose.

The model estimation methodology described above forms an estimation toolkit that allows us to build a concise mathematical description of the input-output system under investigation. We are constructing these models in order to learn the underlying rules from the data. This is similar to theoretical or analytical modelling, but we let the data inform us regarding what terms and effects are dominant etc. Models are therefore constructed term by term.

One decisive advantage of the Narmax modelling procedure is that the relevance or irrelevance of model terms is determined automatically, using the Error Reduction Ratio [Korenberg et al., 1988], a process that does not require knowledge about the system being modelled.

ARMAX and NARMAX procedures are now well established and have been used in many modelling domains [Billings and Chen, 1998]. A more detailed discussion of how structure detection, parameter estimation and model validation are performed is presented in [Korenberg et al., 1988,Billings and Voon, 1986].

3. Robot Training: Method 1

The first training method is based on demonstrating the desired behaviour to the robot by driving it manually using a joystick ([Nehmzow et al., 2005], [Kyriacou et al., 2006] and [Akanyeti et al., 2007a]). The method has three stages:

3.1. *Driving Robot Manually using a Joystick*

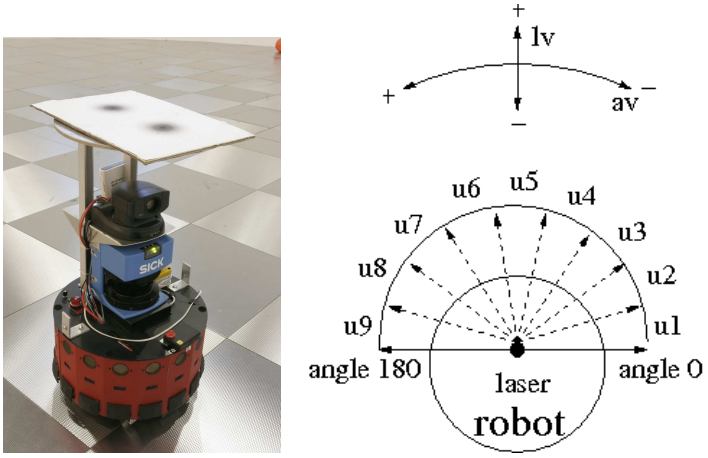
(i) **Acquisition of training data:** The programmer demonstrates the desired behaviour to the robot by driving it manually using a joystick in the target environment. During this run, the sensor perception and the desired velocity commands of the robot are logged.

(ii) **Obtaining sensor based models:** Having obtained the training data, the sensor based control models are obtained using the Narmax system identification method described in section 2.1. These models are mathematical descriptions that link the perception of the robot to the desired motor commands to achieve the desired task.

(iii) **Model validation:** Once the sensor based controllers are obtained, they are used to drive the robot in the training environment to validate their performances.

3.2. *Experiment 1: Wall Following*

In order to demonstrate the viability of our approach, we trained a *Magellan Pro* mobile robot, *Radix* (figure 1) to achieve a right-hand wall following behaviour.



(a) (b)

Fig. 1. (a) Radix, with a 40cm diameter, has two degrees of freedom (translational and rotational). The robot is equipped with a laser range finder, sonar and infrared sensors as well as a camera. The range finder has a wide angular range (180°) with a radial resolution of 1° and a distance resolution of less than 1cm. (b) During experiments, in order to decrease the dimensionality of the input space to Narmax model, we coarse coded the laser readings into 9 sectors (u_1 to u_9) by averaging 20 readings for each 20° intervals.

All experiments described in this paper were conducted in the 100 m^2 circular robotics arena of the University of Essex. The arena is equipped with a *Vicon* motion tracking

system which can deliver position data (x, y and z), using reflective markers and high speed, high resolution cameras. The tracking system is capable of sampling the motion upto 100Hz within a 1mm accuracy.

Acquisition of training data First the robot was driven manually using a joystick for half an hour (figure 2). During this time, the coarse coded laser readings and the rotational velocities of the robot were logged every 250ms. The laser perception of the robot was coarse coded in 9 segments by averaging the laser readings over 20° intervals.

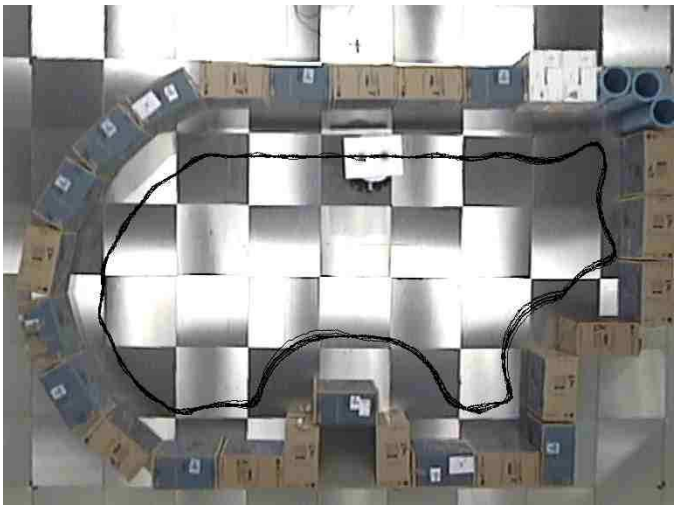


Fig. 2. The programmer demonstrates the desired behaviour, wall following, to the robot by driving it manually using a joystick for half an hour. During this time, the coarse coded laser readings and the rotational velocities of the robot were logged every 250ms.

Sensor signal encoding The coarse coded laser readings bigger than 0.8 and smaller than 0.3 were clamped to 0.8 and 0.3 respectively so that the robot would take into account the walls which are close enough to the robot. The readings were then normalized to be distributed between 0 and 1 and filtered to two element input vector $(\hat{u_1}, \hat{u}_2)$ by extracting the minimum laser reading $(\hat{u}_1)$ among all the normalized readings and the right-most normalized laser reading $(\hat{u}_2)$.

Obtaining sensor-based polynomials After the collection of training data, a polynomial model was obtained, identifying the rotational velocity ω of the robot as a function of the two filtered laser readings $(\hat{u}_1, \hat{u}_2)$. The model was chosen to be a linear ARMAX polynomial structure of first degree with no regression in the inputs and output (i.e. $l = 1$, $N_u = 0$, $N_y = 0$), and contained the 3 terms given in table 1.

In table 1 $\omega(n)$ is the rotational velocity of the robot (in rad/s) at time instant n, $\hat{u}_1(n)$ is the minimum coarse coded and normalized laser reading and $\hat{u}_2(n)$ is the right most coarse coded and normalized laser reading.

$$\omega(n) = +0.463 - 1.967 * \hat{u}_1(n) + 0.901 * \hat{u}_2(n)$$

Table 1
Experiment 1. ARMAX controller which links the coarse-coded laser readings of the robot to its angular velocity to achieve right-hand wall following behaviour (see figure 2).

Testing the model Having obtained the sensor based controller, first, we let the model drive the robot in the training environment for about 15 minutes (corresponding 5 laps around the arena). During the experiments, the travel speed of the robot was kept constant at 0.15m/s. Figure 3 illustrates that the obtained trajectory matches the target trajectory very closely (see figure 2).

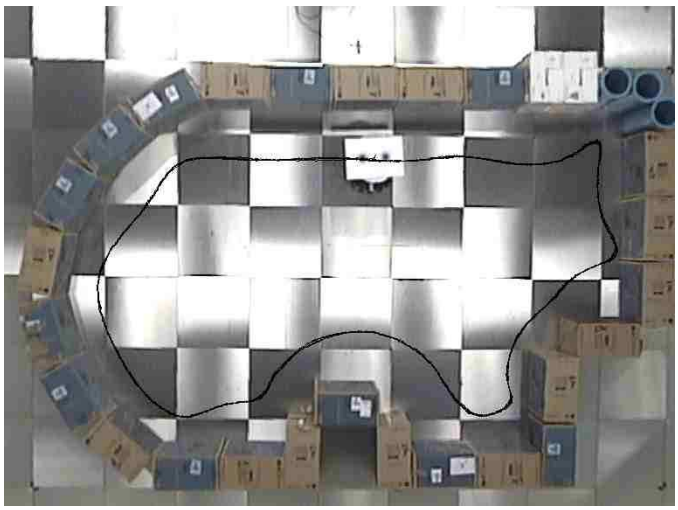


Fig. 3. Experiment 1: The trajectory of the robot driven by the ARMAX controller given in table 1 resembles the training trajectory (figure 2) very closely. Note that the robot traveled round the arena 5 times.

Then we tested our model in a different test environment 15 minutes (5 laps), to see if the model captured the fundamental relationship between the perception and action of the robot in order to achieve the desired wall following behaviour. The results (figure 4) are again satisfactory.

4. Robot Training: Method 2

The second robot training method is based on "programming by demonstration" idea, where the programmer himself demonstrates the desired behaviour to the robot by performing it in the target environment, rather than driving the robot via a joystick ([Nehmzow et al., 2007b,Akanyeti et al., 2007b]). This method has 4 stages:

4.1. *Programming by Demonstration*

(i) **Human demonstration:** First, the programmer demonstrates the desired behaviour by performing it in the target environment. For the purpose of this paper we confined our experiments to 2-dimensional

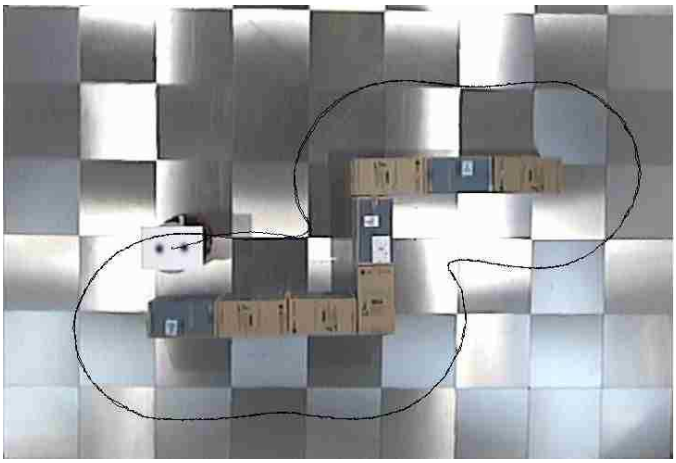


Fig. 4. Experiment 1: Performance of the ARMAX controller in a novel test environment, here the robot traveled round the arena 5 times. The results show that the ARMAX controller captured the fundamental relationship between the coarse coded laser readings and the rotational velocity commands in order to achieve right wall following behaviour, even in an environment that differed from the training scenario.

navigation problems reflecting the motion capabilities of our robot (2 degrees of motion, translational and rotational). During this initial demonstration, we log the x and y position of the human user with a sampling rate of 50Hz by using the *Vicon* motion tracking system. Once the operator's trajectory is logged, we compute the translational and rotational velocities of the demonstrator by differentiating consecutive (x, y) samples along the trajectory.

(ii) **Sensorless trajectory following** In the second stage we use the Narmax system identification method to obtain two sensor-free polynomials, one expressing rotational velocity as a function of time and past rotational velocities, the other expressing the translational velocity as a function of time and past linear velocities.

We then use these two sensor-free polynomials to drive the robot blindly along the trajectory the human had taken earlier, now logging sensor readings and velocities. We use a sampling frequency of 10Hz at this stage.

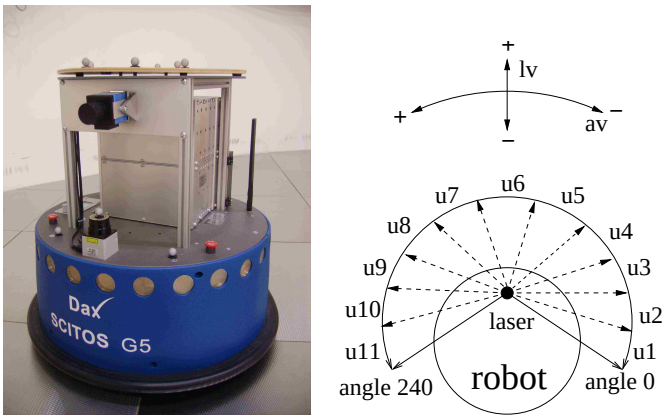
(iii) **Obtaining sensor based controllers** The sensor-free controllers obtained at stage II are ballistic controllers that drive the robot along the desired trajectory, as long as the robot is started from the same initial positions as the human. However, for real-world applications it is essential that sensor feedback is used to control the motion of the robot.

In the final stage we therefore use the Narmax system identification method to obtain sensor-based controllers, using the previously logged sensor-motor pairings. This controller can subsequently be used to control the robot in the target environment, copying the original behaviour exhibited by the human demonstrator.

(iv) **Model validation** Finally we let the obtained controllers drive the robot in the train and test environments to see if the models capture the necessary relationship between the robot's perception and action in order to achieve the desired behaviour.

4.2. *Experiment 2: Corridor Following*

To demonstrate the viability of this second method, we demonstrated to a *Scitos G5* mobile robot (figure 5) how to follow the U-shaped corridor of 150 cm width shown in figure 6. The programmer started at right side and then walked to the end of the corridor. During this time, the position of human was logged in every 20ms by using the *Vicon* motion tracking system.



(a) (b)

Fig. 5. (a) The Scitos G5 robot Dax. Dax, with a base diameter of 60cm, has two degrees of freedom (translational and rotational). The Hokuyo range finder mounted on the robot has a wide angular range (240°) with a radial resolution of 0.36° and distance resolution of less than 1cm. (b) In order to decrease the dimensionality of the input space to the Narmax model, we coarse coded the laser readings into 11 sectors (u_1 to u_{11}) by averaging 62 readings for each 22° intervals.

After we estimated the rotational and translational velocities of the programmer along the desired trajectory through differentiation of (x,y) positions, we used the Narmax system identification method to obtain two sensor-free controllers for the translational velocity and the rotational velocity of the robot. Both models are given in table 2.

After obtaining these sensor-free polynomials, we use them to drive the robot in the U-shaped corridor (figure 7). During this run, the laser readings and the robot's translational and rotational velocities were logged every 100ms.

Sensor Signal Encoding In order to decrease the dimensionality of the input space to the Narmax model, we coarse coded the laser readings into 11 sectors by averaging 62 readings for each 22° interval. We then used the Narmax identification procedure to estimate the robot's translational and rotational velocities as a function of the

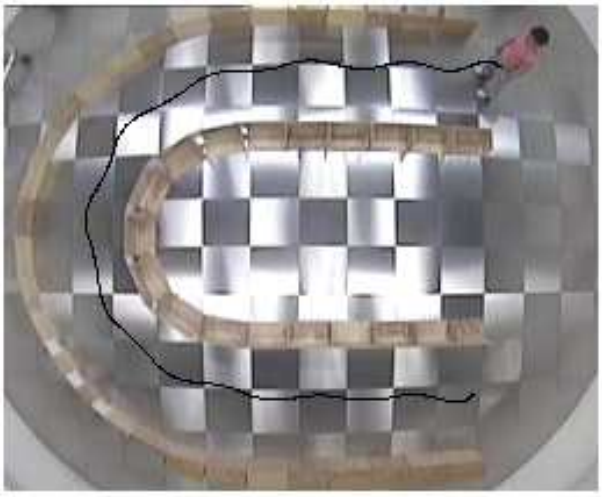


Fig. 6. Experiment 2: The trajectory of the human demonstrator in the U-shaped corridor environment. The width of the corridor is 150 cm.

$lv(n) =$	$av(n) =$
$+0.347$	-0.005
$+0.004 * t(n)$	$+0.001 * t(n)$
$-0.001 * t(n)^2$	$+0.001 * t(n)^2$
	$-0.001 * t(n)^3$
	$+0.818 * y(n-1)$
	$+0.158 * y(n-1)^2$
	$-0.276 * y(n-1)^3$
	$+0.001 * t(n) * y(n-1)$
	$-0.001 * t(n)^2 * y(n-1)$

Table 2
Two sensor-free polynomials to drive the robot "blindly" along the human trajectory given in figure 6. $lv(n)$ is the translational velocity in m/s, $av(n)$ the rotational velocity in rad/s and $t(n)$ is time variable at time instant n.

coarse coded laser readings (u_1, u_2, $\cdots$, u_{11}). Both models are given in table 3.

Models validation We then validated the sensor-based models given in table 3 by letting them control the robot in the U shaped corridor. We started the robot from 10 different locations, and observed correct corridor following behaviour in all cases. The resulting trajectories are shown in figure 8.

5. Extending Method 2

The experiments presented so far used an external, camera based motion tracking system to log trajectory information. However, such tracking systems are complicated to set up, have to be calibrated, and, most importantly for service robotics applications, are not found in real world environments.

Fig. 7. Experiment 2: The trajectory of the robot under control of the sensor-free polynomials given in table 2. The models drive the robot along the human trajectory given in figure 6 without using any sensory perception. During this run, the robot logs its own perception and velocity commands. The logged data is then used to obtain the final, sensor-based controllers which link the perception of the robot to motor commands.

$lv(n) =$	$av(n) =$
$+1.011$	$+0.570$
$-0.037 * u_1(n)$	$+0.002 * u_1(n)$
$+0.164 * u_2(n)$	$+0.069 * u_2(n)$
$+0.147 * u_3(n)$	$+0.052 * u_3(n)$
$-0.128 * u_4(n)$	$-0.181 * u_4(n)$
$-0.116 * u_5(n)$	$-0.046 * u_5(n)$
$-0.051 * u_6(n)$	$-0.049 * u_6(n)$
$-0.075 * u_7(n)$	$-0.038 * u_7(n)$
$-0.051 * u_8(n)$	$-0.020 * u_9(n)$
$-0.074 * u_9(n)$	$-0.050 * u_{10}(n)$
$-0.131 * u_{10}(n)$	

Table 3
Two sensor-based polynomials which link the robot's perception to motor commands in order to achieve the behaviour shown in figure 6. $lv(n)$ and $av(n)$ are the robot's translational velocity in m/s and rotational velocity in rad/s at sampling point n. u_1 to u_{11} are the laser bins defined in figure 5.

In [Nehmzow et al., 2007a], we therefore proposed a new method for replacing the *Vicon* motion tracking system, estimating the position of the programmer by interpreting camera images through a Narmax model, which was trained to predict the position of the demonstrator from the camera image.

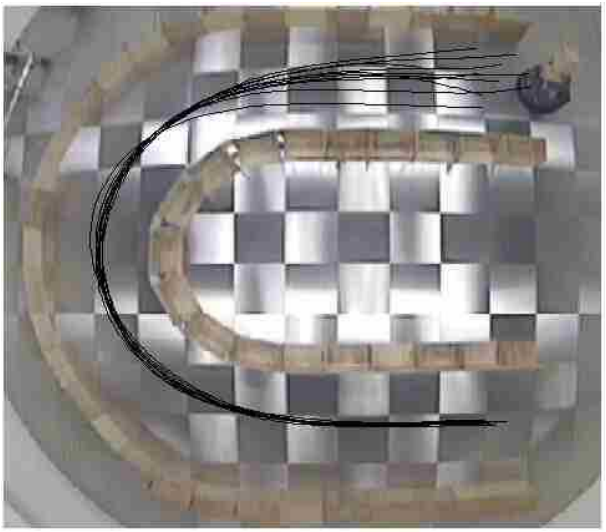
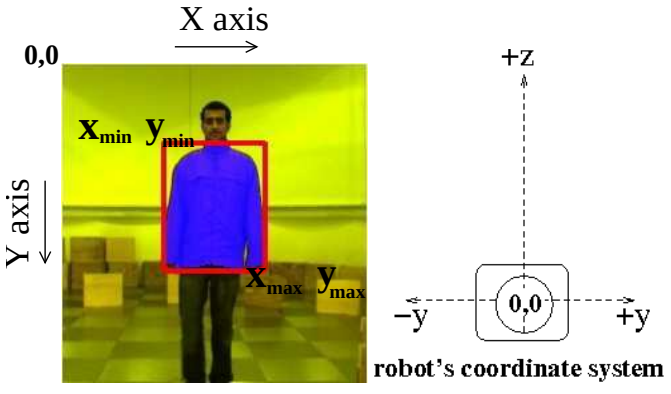


Fig. 8. Experiment 2: Ten trajectories of the robot under control of the sensor-based controllers given in table 3. Note the convergence to one stable trajectory, irrespective of starting point.

5.1. *Demonstrator position estimation, using perceptual features*

Our method is based on tracking the red jacket worn by the programmer during demonstration. Theoretically, assuming that the programmer is always facing to the robot's camera — so that the jacket appears as a rectangular object in the images — the position of the programmer can be estimated using the position and the size of the jacket blob appeared in the images (see figure 10 and 9). The information about the apparent position and the size of the jacket is obtained by extracting the "top left" and "bottom right" coordinates of the jacket blob in the images (figure 9a) and these coordinates are then fed to the Narmax models as inputs (figure 10).



(a) (b)

Fig. 9. (a) The output image after a pre-processing stage. The demonstrator's jacket was isolated from the rest of the image using a blob colouring algorithm. (b) The position information of the jacket is then fed to Narmax models which identify the position of the demonstrator relative to the robot's reference coordinate.

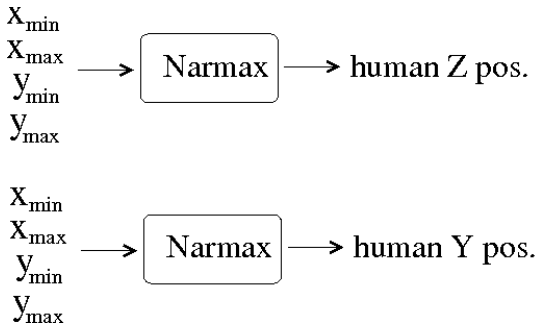


Fig. 10. Position estimation models, identifying the relationship between the position of the jacket in the image and the position of the demonstrator referenced to the robot's coordinate system. (x_{min}, y_{min}) and (x_{max}, y_{max}) are the coordinates of the "top left" and "bottom right" corners of the rectangle surrounding the red jacket (see figure 9). These coordinates are computed by separating the red jacket from the background of the image using a blob colouring algorithm.

Note that, the position of the programmer is computed relative to the robot's coordinate frame which is given in figure 9b.

Finding the position of the jacket Figure 11 shows the block diagram that illustrates how we extract the two coordinates (x_{min}, y_{min}) and (x_{max}, y_{max}) from the captured images.

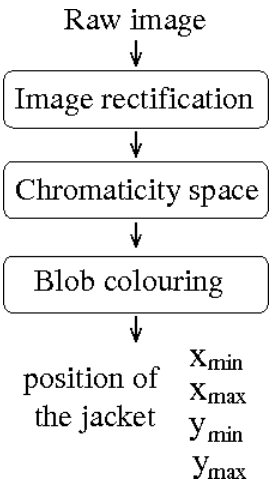


Fig. 11. The block diagram that illustrates how the red jacket is extracted from the captured images. First the lens distortions are removed from the raw image. Then the corrected image is converted into chromaticity space which is less illumination dependent. In the final stage, the jacket is separated from the background of the image by blob colouring algorithm (see figure 9).

First we remove lens distortions from the captured images. Once the image is corrected, we convert it into "chromaticity colour space" which is less illumination dependent that the RGB colour space (equations 1-3).

$$C_r = \frac{R}{R+G+B}, \tag{1}$$

$$C_g = \frac{G}{R+G+B}, \tag{2}$$

$$C_b = \frac{B}{R+G+B}, \tag{3}$$

where C_r, C_g and C_b are the red chromaticity, green chromaticity and blue chromaticity components respectively, and R, G and B are the red, green and blue values respectively of the colour to be described.

In the final stage, we separate the jacket from the background of the image by using a blob colouring algorithm. Blob colouring is a technique used to find regions of similar colour in the image. The connected pixels are grouped together if they have similar intensity values and assigned to different regions if the difference in their intensity values is bigger than a certain threshold.

After completion of the blob colouring algorithm we assume that the biggest red coloured region corresponds to the red jacket, making sure that assumption holds through our experimental procedure. We then compute the minimum rectangular box which can frame the jacket entirely (figure 9). The coordinates of the "top left" and "bottom right" corners of the rectangle are then fed to Narmax polynomials obtained to predict the position of the demonstrator.

Acquisition of the training data In order to collect training data for the estimation of the programmers's position during the demonstration, the human trainer wearing a red jacket walked randomly in the robot's field of view for half an hour. During the training session the robot was stationary, and the robot's camera was aligned parallel to the floor so that the robot's field of view has the maximum coverage area for the demonstrator.

During this time the "top left" and "bottom right" coordinates of the jacket blob and the relative position of the demonstrator referenced to the robot — obtained through the *Vicon* motion tracking system — were logged synchronously every 250ms. Figure 12 shows the stream of the original and the processed images captured during the training session.

Obtaining position prediction models We then used the Narmax system identification procedure to estimate the position of the demonstrator as a function of (x_{min},x_{max}) and (y_{min},y_{max}). The Z direction position prediction model Z_j was chosen to be second degree with no regression in the input and output (i.e. $l = 2$, $N_u = 0$, $N_y = 0$). The resulting model contained 6 terms. The Y direction position prediction model Y_j was chosen to be fourth degree with no regression in the input and the output (i.e. $l = 4$, $N_u = 0$, $N_y = 0$) and contained 7 terms. Both models are given in table 4.

Figure 13 shows the predicted and actual position of the demonstrator using validation data set obtained during training.

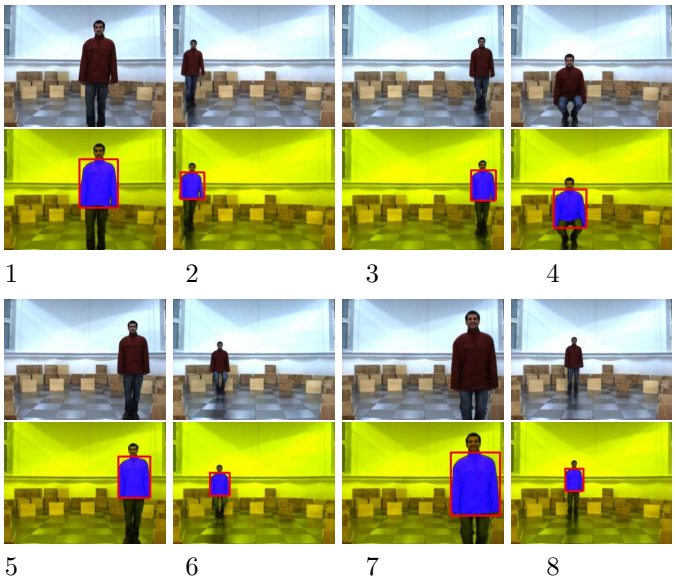


Fig. 12. Stream of images captured by the robot's camera during the acquisition of the training data set used in obtaining Narmax model in order to calculate the position of the demonstrator. Each image is pre processed using the procedure given in figure 11 in order to extract the position of the red jacket in the image.

$Z_j(n) =$	$Y_j(n) =$
$+14.684$	-4.128
$+0.148 * u_3(n)$	$-0.015 * u_1(n)$
$-0.184 * u_4(n)$	$+0.051 * u_2(n)$
$+0.001 * u_3(n)^2$	$-0.001 * u_2(n)^2$
$+0.001 * u_4(n)^2$	$+0.001 * u_1(n) * u_2(n)^2$
$-0.001 * u_3(n) * u_4(n)$	$+0.001 * u_1(n) * u_1(n)^3$
	$-0.001 * u_1(n) * u_2(n)^3$

Table 4
Two position polynomials which link the perception of the robot to the position of the demonstrator relative to the robot's coordinate frame. $Z_j(n)$ and $Y_j(n)$ are the z position (in m) and y position (in m) of the demonstrator at time instant n and u_1 to u_4 are the x_{min}, x_{max}, y_{min} and y_{max} coordinates of the "red jacket blob" extracted from the image as described in figure 11.

Model validation We compared the predicted position of the demonstrator with the actual position by analysing the error distributions. The results show that the average error between the predicted and actual position of the demonstrator is less than 10mm±3mm for both models (see figure 14).

5.2. *Experiment 3 Door Traversal*

After obtaining the position prediction models (Z_j,Y_j), we replaced the *Vicon* motion tracking system with the new models tested the whole system by teaching a *Scitos G5* mobile robot, Rex (figure 15) to achieve door-traversal

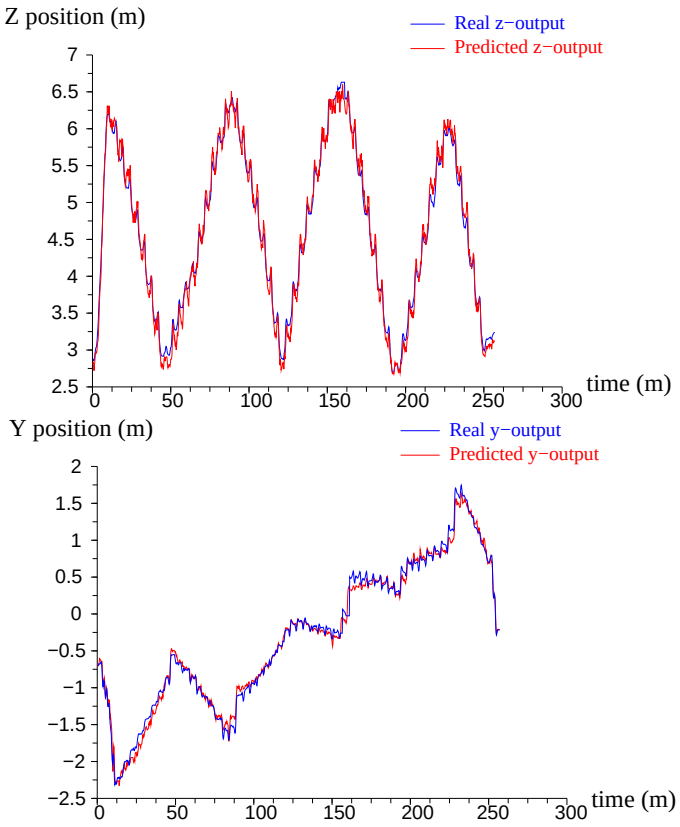


Fig. 13. The predicted and actual position of the demonstrator in the robot's coordinate frame (in meters).

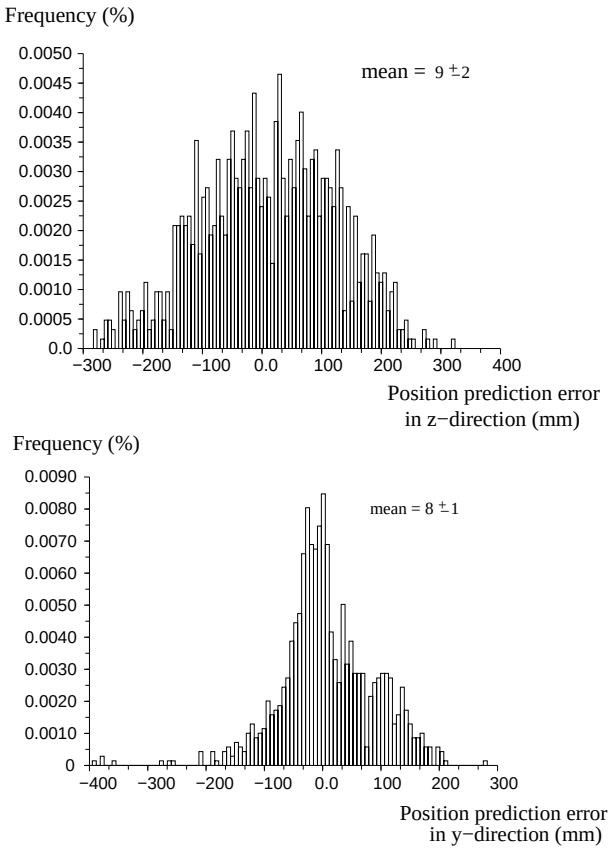


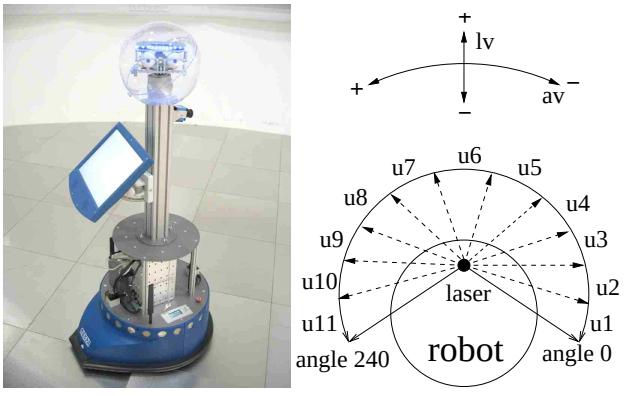
Fig. 14. The error distributions of the Z position prediction model and the Y position prediction model. The average error between the predicted and actual position of the demonstrator is less than 10mm±3mm for both models.

behaviour. The programmer walked through two consecutive door like openings of 120 cm width. During this time, the robot calculates the trajectory of the programmer using the trajectory capturing mechanism described in section 5.1 every 250ms. Figure 16 shows the general experimental scenario in which the programmer performed the desired behaviour while the robot was observing him. Figure 17 shows the stream of images which were captured and processed by the robot during the demonstration.

Analysis of the observed trajectory reveals that there is noise in the data because of two reasons:

(i) There is a constant oscillation in the motion of the demonstrator, which originates from the swinging motion perpendicular to the heading direction. This is a general characteristic of the two legged locomotion in humans.

(ii) The polynomials that compute the position of the demonstrator with respect to the robot are very sensitive to how accurate the position and the size of the jacket is extracted from the image. So far we assumed that the red jacket is a rectangular, rigid object; in other words, that the coordinates of the top left and bottom right corners of the jacket would give us the position in the image and the size of the jacket accurately. However the information about the position and the size can be noisy when the shape of the



(a) (b)

Fig. 15. (a) Rex has two degrees of freedom (translational and rotational) and is equipped with a Hokuyo laser range finder. The range finder has a wide angular range (240°) with a radial resolution of 0.36° and distance resolution of less than 1cm. During experiments, in order to decrease the dimensionality of the input space to Narmax model, we coarse coded the laser readings into 11 sectors (u_1 to u_{11}) by averaging 62 readings for each 22° intervals (b).

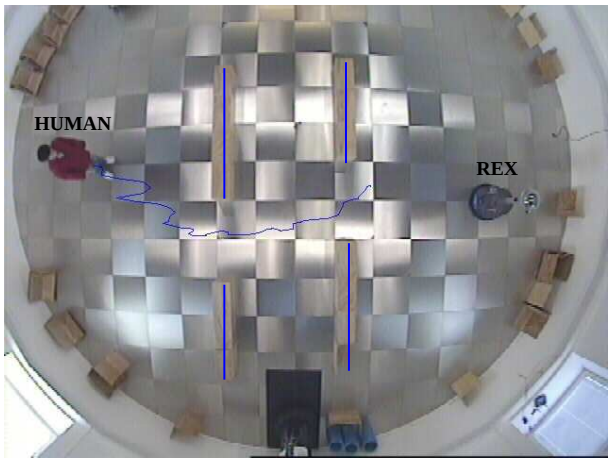


Fig. 16. Experiment 3: The trajectory taken by the demonstrator while passing through the two openings of 120cm width. While the demonstrator was performing this behaviour, the robot was observing him and calculating the demonstrator's relative trajectory according to the robot itself using position prediction models given in table 4.

jacket is deformed due to the shoulder motions of the demonstrator.

We eliminated some noise by using a low pass filter, assuming that the demonstrator did not do sharp changes in the heading direction while performing the desired task. We then computed the translational and rotational velocities of the demonstrator along the trajectory.

Having obtained the velocity information of the programmer along the desired path, we used them to drive the robot blindly in the test environment. During this first robot interaction with the environment, laser readings and the robot's translational and rotational velocities were logged in every 250ms (see figure 18).

Sensor signal encoding In order to decrease the dimensionality of the input space to the Narmax model, we coarse coded the laser readings into 11 sectors by averaging 62 readings for each 22° interval. We then used the Narmax identification procedure to estimate the robot's translational and rotational velocities as a function of the coarse coded laser readings $(u_1, u_2, \cdots, u_{11})$, see figure 15.

Both the translational and the steering speed model were chosen to be second degree. No regression was used in the inputs and output (i.e. $l = 2$, $N_u = 0$, $N_y = 0$) resulting in non linear Narmax structures. Both the models contained 18 terms (table 5).

Models validation We then let the sensor based models drive the robot in the test environment starting from 15 different locations. Figure 19 shows that the obtained models are successfull in driving the robot through both the door like openings without collisions.

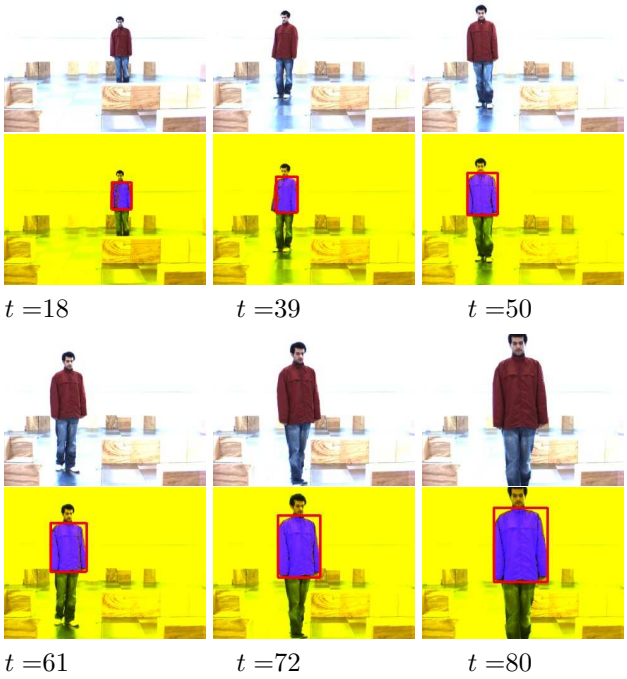


Fig. 17. Stream of six images captured by the robot's camera during the demonstration of the desired behaviour. The numbers below each image indicate the frame number of the image (frame rate 4Hz). The captured images are then pre processed as described in section 5.1 in order to extract the position of the red jacket.

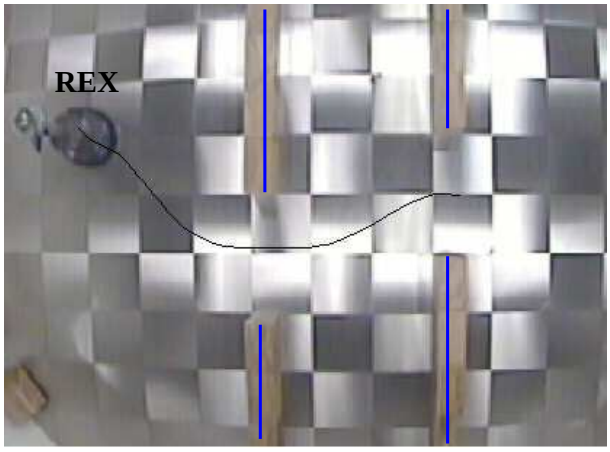


Fig. 18. Experiment 3: The trajectory of the robot under the control of the given velocity commands obtained from the human trajectory given in figure 16, not using any sensory perception. During this pass, *Rex* logs its own perception perceived along the trajectory and the velocity commands. The logged data is then used to obtain sensor-based controllers which link the perception of the robot to the desired behaviour.

6. Better Position Prediction

The method of estimating the programmers's position described in the previous section assumes that his red jacket is a rigid and rectangular object where the size of the jacket

$lv(n) =$	$av(n) =$
-0.751	$+2.260$
$+0.059 * u_3(n)$	$-0.054 * u_3(n)$
$+0.243 * u_4(n)$	$-0.480 * u_4(n)$
$+0.040 * u_5(n)$	$-0.144 * u_5(n)$
$+0.139 * u_6(n)$	$-0.637 * u_6(n)$
$+0.219 * u_7(n)$	$-0.195 * u_7(n)$
$-0.040 * u_8(n)$	$+0.212 * u_8(n)$
$-0.001 * u_9(n)$	$-0.056 * u_9(n)$
$-0.003 * u_3(n)^2$	$-0.003 * u_3(n)^2$
$-0.008 * u_4(n)^2$	$+0.014 * u_4(n)^2$
$-0.007 * u_6(n)^2$	$+0.004 * u_5(n)^2$
$-0.024 * u_7(n)^2$	$+0.085 * u_6(n)^2$
$+0.008 * u_8(n)^2$	$-0.029 * u_8(n)^2$
$-0.003 * u_9(n)^2$	$+0.013 * u_9(n)^2$
$-0.010 * u_3(n) * u_4(n)$	$+0.012 * u_3(n) * u_4(n)$
$-0.001 * u_3(n) * u_5(n)$	$-0.004 * u_3(n) * u_8(n)$
$-0.028 * u_4(n) * u_6(n)$	$+0.033 * u_4(n) * u_6(n)$
$-0.024 * u_4(n) * u_7(n)$	$+0.050 * u_4(n) * u_7(n)$

Table 5
EXPERIMENT 3. TWO SENSOR-BASED POLYNOMIALS WHICH LINK THE PERCEPTION OF THE ROBOT TO THE DESIRED BEHAVIOUR SHOWN IN FIGURE 18. $lv(n)$ AND $av(n)$ ARE THE TRANSLATIONAL VELOCITY (IN m/s) AND ROTATIONAL VELOCITY (IN rad/s) OF THE ROBOT AT TIME INSTANT n AND u_1 TO u_{11} ARE THE COARSE CODED LASER READINGS STARTING FROM THE RIGHT EXTREME OF THE ROBOT.

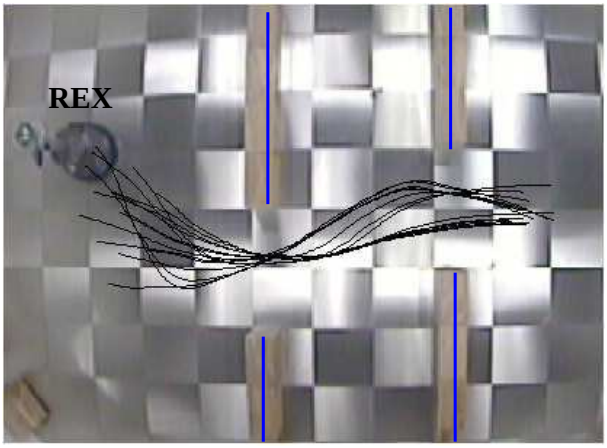


Fig. 19. EXPERIMENT 3: THE TRAJECTORIES OF THE ROBOT UNDER THE CONTROL OF SENSOR BASED CONTROLLERS GIVEN IN TABLE 5. THE ROBOT WAS STARTED FROM 15 DIFFERENT LOCATIONS AND IT PASSED THROUGH THE BOTH DOOR LIKE OPENINGS SUCCESSFULLY.

only changes as a function of the position of the programmer. This assumption is limiting, for two reasons:

(i) The programmer always has to face to the robot's camera during demonstration so that the jacket is visible to the camera as a rectangular object. Besides being uncomfortable trying to walk always facing to the camera, it is almost impossible to do so consistently, and some noise is thereby introduced.

(ii) The jacket takes the form of the programmer's body, therefore the size and the shape of the jacket does not only depend on the position of the programmer, but also on body form and demonstrator movements.

We therefore decided to track an orange sphere, rather than the red jacket, because the perception of a sphere is orientation-independent (figure 20).

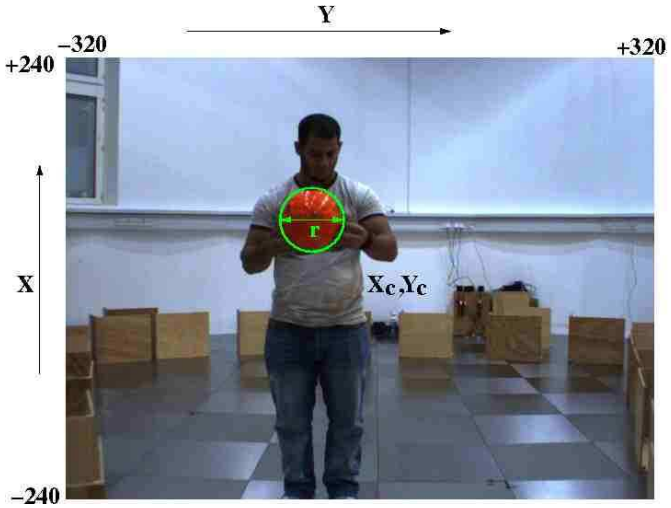


Fig. 20. TRACKING AN ORANGE BALL TO ESTIMATE THE POSITION OF THE PROGRAMMER DURING DEMONSTRATION. ONCE THE IMAGE IS CAPTURED, THE BALL IS EXTRACTED FROM THE IMAGE USING A BLOB COLOURING ALGORITHM WHERE X_c AND Y_c ARE THE CENTRE POSITION AND r IS THE DIAMETER OF THE BALL IN THE BLOB IMAGE.

6.1. *Position prediction using an orange ball*

For narrow angle view and good quality cameras like the one mounted on the robot, we can assume that the perspective projections of the ball correspond to circular regions on the image plane regardless of the position and the orientation of the ball in the real world. Therefore:

(i) The radius of the circular region on the image plane is directly proportional with the radius of the ball and inversely proportional to the distance between the ball and the robot's camera. As the ball is further away from the camera, the size and therefore the radius of the ball get smaller and vice versa.

(ii) The perspective projection of the ball is independent from the orientation of the ball, therefore during the demonstration the programmer does not have to face the camera all the time.

Based on the above assumptions, the relative position of the ball according to the robot can be computed as:

$$Z_b = \frac{R}{r} * f \tag{4}$$

$$Y_b = \frac{R}{r} * y_c \tag{5}$$

where Z_b and Y_b are the relative z and y positions of the ball centre according to the robot, R is the diameter of the ball in the real world, r is the diameter of the circular region

on the image plane corresponding to ball, y_c is the y pixel coordinate of the ball silhouette on the image plane and f is the focal length of the camera (figure 21).

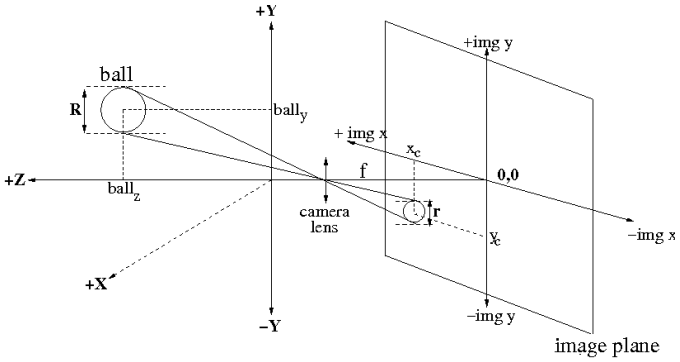


Fig. 21. The perspective projection of the ball on the image plane where R and r are the diameter of the ball and ball silhouette on the image plane respectively.

Training position prediction models We did not use intrinsic camera parameters such as focal length and the image centre of the camera, but decided to train an ARMAX polynomial in the same form with theoretical formulae given in equations 4 and 5 to identify the coefficients of the formulae without needing to know the camera parameters.

In order to collect training data for the estimation of the programmers's position, the human trainer holding an orange ball walked randomly in the robot's field of view for half an hour. During the training session the robot was stationary, and the robot's camera was aligned parallel to the floor so that the robot's field of view had maximum coverage area for the demonstrator.

During this time the position and the radius of the ball in the captured images and the relative position of the demonstrator referenced to the robot — obtained through the *Vicon* motion tracking system — were logged synchronously every 100ms. Figure 22 shows stream of original and the processed images captured during the training session.

In order to find the centre and the radius of the ball in the images, we modified the blob colouring algorithm — previously described in section 5.1 — in such a way that it extracted the biggest orange coloured region in the image.

We then used an ARMAX system identification process to find the coefficients of the theoretical formulae which express the relative position of the ball with respect to the robot coordinate frame in the real world as a function of the position and the diameter of the ball silhouette in the image. The resulting models are given in table 6, where $Z_b(n)$ is the position of the ball in z direction and $Y_b(n)$ is the position of the ball in y direction.

Figure 23 shows the actual and predicted positions of the ball in the robot's coordinate frame.

Models analysis and validation The analysis of the Z position prediction graph in figure 23 confirms the expectation that prediction gets worse as the distance between ball and

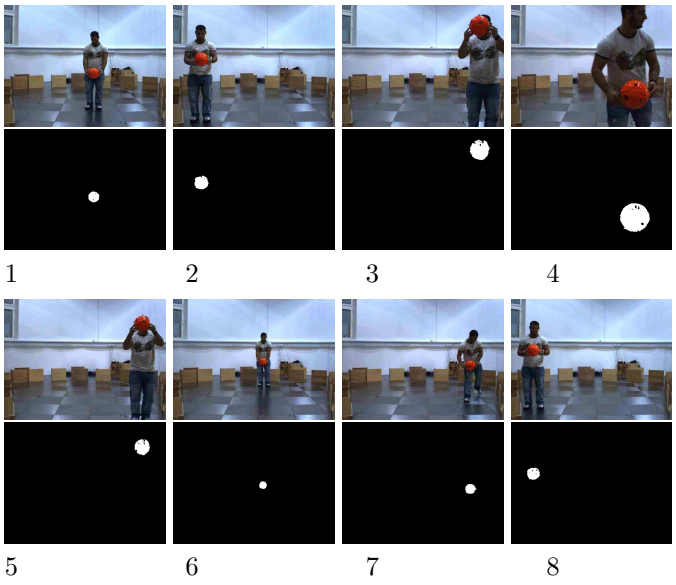


Fig. 22. Stream of images captured by the robot's camera during the training session. The captured images are then preprocessed to compute the position and the diameter of the ball in the image (see also figure 20).

$Z_b(n) =$	$Y_b(n) =$
$+128.321$	-63.429
$+145792.540 * u_z(n)$	$-209.167 * u_y(n)$

Table 6
Two position predictors which model the balls z and y positions (in mm) within the robot's coordinate system as a function of the balls image positions. $u_z(n)$ is $\frac{1}{r}$ and $u_y(n)$ is $\frac{y_c}{r}$ at time instant n.

camera increases. As the apparent diameter of the ball decreases prediction models become more sensitive to errors during the calculation of the input terms at the end of the blob colouring algorithm.

In order to find out how sensitive the performance of the Z_b position predictor is to the smallest possible error of "one pixel" we took the first derivative of the Z_b position prediction model with respect to the diameter of the ball:

$$\frac{dZ_b}{du_z} = \frac{-145792.540}{u_z^2} \tag{6}$$

and plotted the resultant equation (6) for different diameter values varying between 20 pixels (when the ball is approximately 6 metres away from the camera) to 140 pixels (when the ball is approximately 1 meter away from the camera) .

The resulting graph (figure 25) shows that the performance of the prediction model is very robust when the ball is close to the camera. When the distance between the camera and the ball is around 1m, the diameter of the ball in the camera image is around 140 pixels and the error between the actual and the predicted position for one pixel miscalculation error is less than a centimeter. As the ball gets further away from the camera, the performance of the position prediction model drops drastically. When the di-

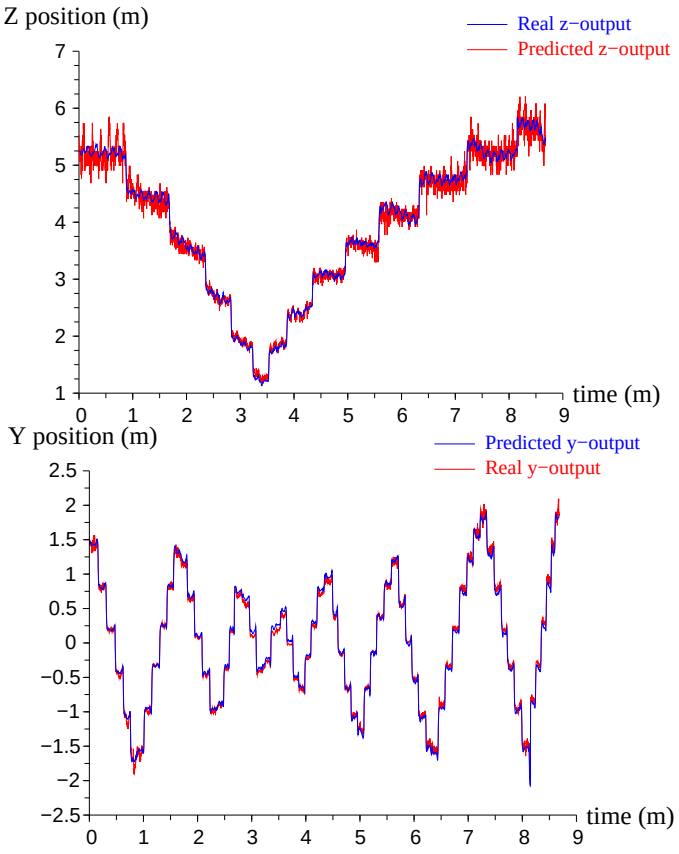


Fig. 23. The predicted (red) and actual position (blue) of the ball relative to the robot's coordinate frame (in meters).

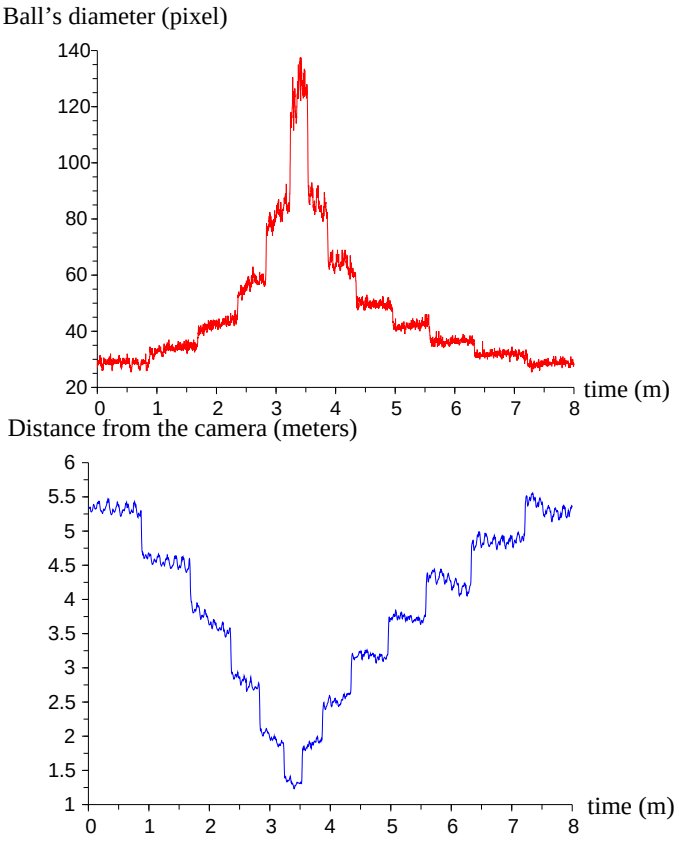


Fig. 24. These figures show how the diameter of the ball silhouette on the image plane is related to the relative position of the ball according to the camera. As the ball goes further away from the camera, the diameter of the ball on the image plane gets smaller and vice versa.

ameter of the ball is around 20 pixels (the ball is approximately 6m away from the camera) the one pixel miscalculation of the balls diameter causes approximately 35cm error in predicting the position.

We also computed the error distributions between the actual and the predicted positions of the demonstrator in the y and z directions (figure 26).

Comparing performance using the two different markers When we compare the error distributions of the two position prediction models obtained for the orange ball (Z_b, Y_b) with the two models obtained for the red jacket (Z_j, Y_j), we see that the red jackets models with an average error of 10mm±3mm are significantly more accurate in predicting the position of the programmer than the models obtained for the orange ball where the mean error is 100mm±3mm (U-test, p< 0.05).

This result was exactly the opposite of what we expected to see at the beginning of the experiments because we assumed that, opposite to the jacket, the image silhouette of the ball is not affected from the motions of the programmer during the demonstration and therefore we expected to obtain more accurate position estimations when we track the ball rather than the jacket.

However the error analysis revealed that the performance of the position prediction models also depends on the size

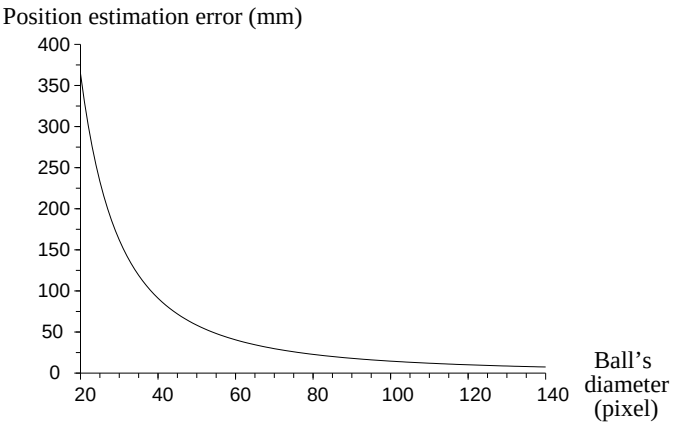


Fig. 25. The plot of $\frac{dZ_b}{du_z} = \frac{-145792.540}{u_z^2}$ where u_z, the diameter of the ball in the image, varies between 20 and 140. This graph indicates how much error would be expected from the performance of the Z position prediction model for a "one pixel error" with respect to the true diameter of the ball.

of the object being tracked, here as the size of the object increases, the models become less sensitive to the noise introduced while computing the position and the size of the tracked object during image processing stage (figure 25.

In this particular experiment, eventhough the noise intro-

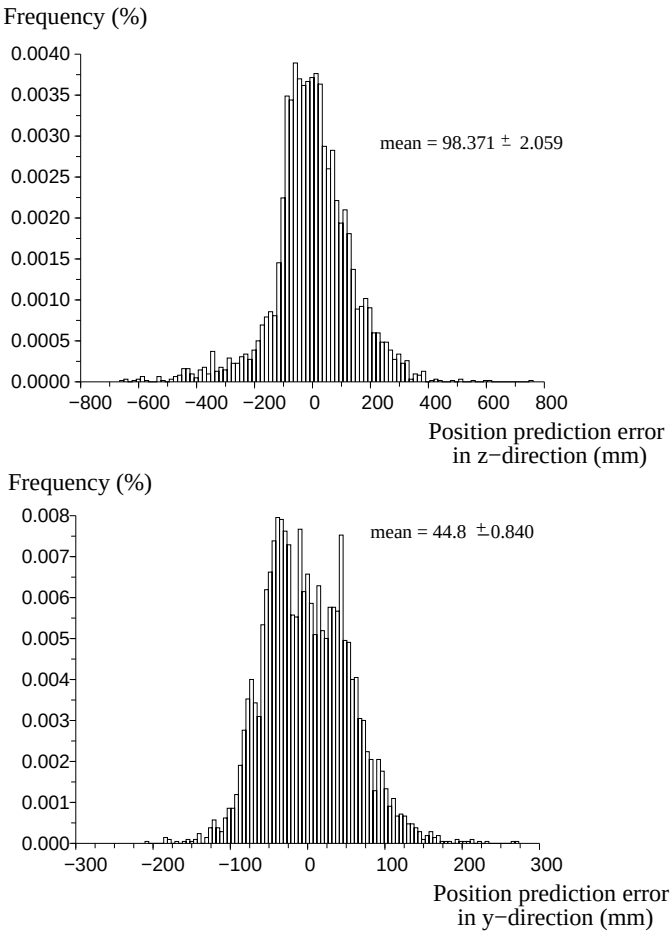


Fig. 26. The error distributions of the position prediction models obtained for predicting the position of the orange ball. Compared with the performance of the models obtained for the red jacket (figure 14), we see that the red jackets models with an average 10mm±3mm are significantly more accurate in predicting the position of the programmer than the models obtained for the orange ball with an average 100mm±3mm (U-test, $p < 0.05$).

duced by the motions of the programmer to image silhouette of the ball is less than the image silhouette of the jacket, since the ball has smaller size compared to the jacket, the position prediction models obtained for the ball give worse performance.

6.2. *Experiment 4: Route Learning*

After obtaining position prediction models (Z_b,Y_b), we tested the viability of the models by teaching Rex (figure 15) to follow a user-specified route in the environment given in figure 27. The programmer walked once to show the robot the desired route. During this time, the robot calculated the position of the demonstrator using Z_b and Y_B every 100ms. Figure 28 shows a stream of images which were captured and used by the robot to obtain the position information of the programmer during the demonstration.

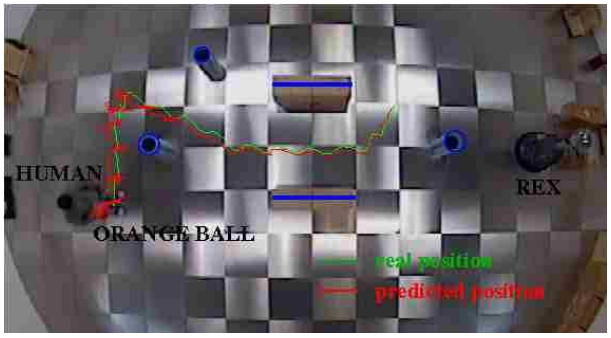


Fig. 27. Experiment 4: The trajectory followed by the programmer while following the desired route. The green line shows the real trajectory of the programmer obtained from Vicon motion tracking system and the red line shows the predicted trajectory of the programmer computed by the position prediction models given in table 6.

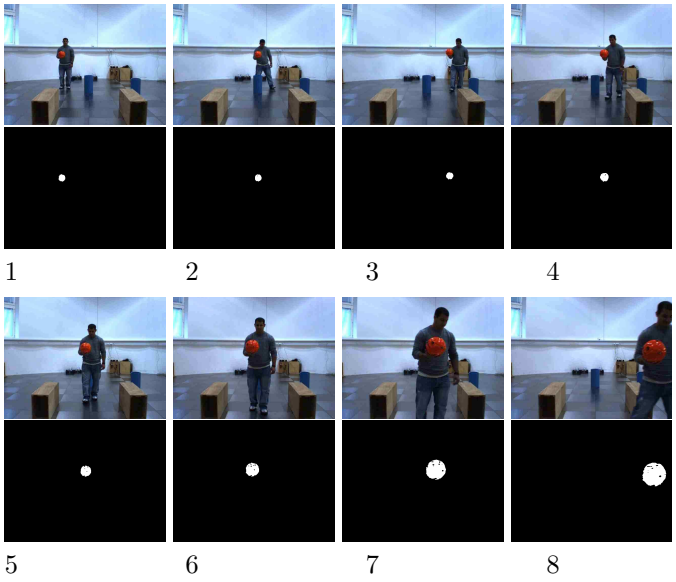


Fig. 28. Stream of images captured by the robot's camera during the demonstration of the desired route following behaviour.

Acquisition of the training data We then computed the translational and rotational velocities of the programmer along the desired trajectory. Having obtained the velocity information, we drove the robot blindly in the target environment. During this time the laser readings and the robot's translational and rotational velocities were logged in every 100ms (figure 29).

Obtaining sensor based polynomials As in the previous door traversal experiment described in section 5.2, we coarse coded the laser readings into 11 sectors by averaging 62 readings for each 22 degree intervals in order to decrease the dimensionality of the input space to the Narmax model. We then used the Narmax identification procedure to estimate the robot's translational and rotational velocities as a function of the coarse coded laser readings (u_1, u_2, $\cdots$, u_{11} (see figure 15)).

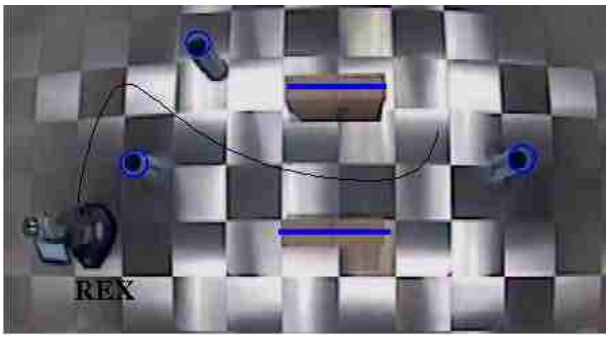


Fig. 29. Experiment 4: The trajectory of the robot while following the desired route blindly (figure 27). During this run, the robot logs its own perception and the desired velocity commands every 100ms. The logged data were then used in the training of NARMAX polynomials to obtain sensor-based controllers given in table 7.

Both the translational and the steering speed model were chosen to be second degree. No regression was used in the inputs and output (i.e. $l = 2$, $N_u = 0$, $N_y = 0$) resulting in non linear Narmax structures. The linear velocity model contained 43 terms and the angular velocity model contained 40 terms (table 7).

Model analysis and validation Having obtained the models given in table 7, we used them to drive the robot in the test environment starting from 10 different locations. Figure 30 shows that the obtained models successfully negotiate the route, but that in three runs the robot did not finish at the exact route end F.

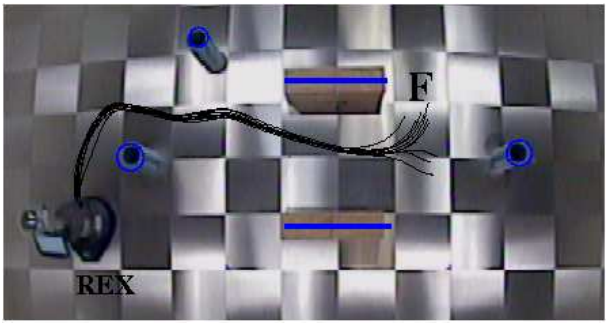


Fig. 30. Experiment 4: The trajectories of the robot under the control of the sensor-based polynomials given in table 7. During the experiments the robot was started from 10 different locations. The robot completed the route in all ten cases, but in three cases did not finish at location F.

We observed that the obtained perception models are not robust, but sensitive to small perturbations, such as the initial starting positions of the robot. We attribute this to the complexity of the route which we were trying to learn where along the desired trajectory the relationship between the robot's perception and the desired motor responses vary, depending on the robot's position along the trajectory. For example when the robot is in region A (figure 31) the actions of the robot are related to the input readings coming from the sensors found on the right side of the robot. Later

$lv(n) =$	$+0.218$	$av(n) =$	-0.608
	$-0.008 * u_1(n)$		$+0.059 * u_1(n)$
	$-0.109 * u_2(n)$		$-0.178 * u_2(n)$
	$+0.071 * u_3(n)$		$-0.070 * u_3(n)$
	$-0.075 * u_5(n)$		$+0.040 * u_4(n)$
	$-0.039 * u_6(n)$		$+0.032 * u_5(n)$
	$+0.132 * u_7(n)$		$+0.089 * u_6(n)$
	$+0.029 * u_8(n)$		$+0.264 * u_7(n)$
	$-0.037 * u_9(n)$		$+0.235 * u_8(n)$
	$-0.045 * u_{10}(n)$		$-0.110 * u_9(n)$
	$-0.092 * u_{11}(n)$		$-0.080 * u_{10}(n)$
	$+0.010 * u_1(n)^2$		$-0.100 * u_{11}(n)$
	$+0.023 * u_2(n)^2$		$+0.008 * u_1(n)^2$
	$-0.015 * u_3(n)^2$		$+0.037 * u_2(n)^2$
	$+0.001 * u_4(n)^2$		$+0.030 * u_3(n)^2$
	$+0.007 * u_5(n)^2$		$-0.002 * u_4(n)^2$
	$+0.003 * u_6(n)^2$		$-0.009 * u_5(n)^2$
	$-0.029 * u_7(n)^2$		$-0.017 * u_6(n)^2$
	$-0.005 * u_8(n)^2$		$-0.047 * u_7(n)^2$
	$+0.010 * u_9(n)^2$		$-0.025 * u_8(n)^2$
	$+0.014 * u_{10}(n)^2$		$+0.030 * u_9(n)^2$
	$+0.016 * u_{11}(n)^2$		$+0.025 * u_{10}(n)^2$
	$-0.006 * u_1(n) * u_2(n)$		$+0.020 * u_{11}(n)^2$
	$+0.001 * u_1(n) * u_4(n)$		$+0.001 * u_1(n) * u_2(n)$
	$-0.004 * u_1(n) * u_6(n)$		$-0.020 * u_1(n) * u_3(n)$
	$-0.001 * u_1(n) * u_7(n)$		$-0.022 * u_1(n) * u_5(n)$
	$-0.003 * u_1(n) * u_8(n)$		$-0.004 * u_1(n) * u_6(n)$
	$-0.001 * u_2(n) * u_4(n)$		$+0.004 * u_1(n) * u_7(n)$
	$+0.001 * u_2(n) * u_5(n)$		$+0.004 * u_1(n) * u_9(n)$
	$+0.005 * u_2(n) * u_6(n)$		$+0.001 * u_1(n) * u_{10}(n)$
	$+0.008 * u_2(n) * u_7(n)$		$+0.008 * u_2(n) * u_3(n)$
	$-0.001 * u_2(n) * u_8(n)$		$+0.001 * u_2(n) * u_5(n)$
	$-0.003 * u_2(n) * u_9(n)$		$-0.008 * u_3(n) * u_8(n)$
	$-0.005 * u_2(n) * u_{10}(n)$		$-0.001 * u_4(n) * u_5(n)$
	$+0.002 * u_3(n) * u_4(n)$		$-0.003 * u_4(n) * u_7(n)$
	$-0.003 * u_3(n) * u_8(n)$		$-0.004 * u_4(n) * u_9(n)$
	$-0.002 * u_4(n) * u_{10}(n)$		$+0.027 * u_5(n) * u_6(n)$
	$+0.005 * u_5(n) * u_6(n)$		$-0.008 * u_5(n) * u_9(n)$
	$+0.009 * u_5(n) * u_8(n)$		$-0.011 * u_7(n) * u_{10}(n)$
	$-0.003 * u_5(n) * u_9(n)$		$-0.031 * u_8(n) * u_{10}(n)$
	$+0.002 * u_5(n) * u_{11}(n)$		
	$+0.004 * u_6(n) * u_8(n)$		
	$+0.005 * u_6(n) * u_{11}(n)$		

Table 7
Experiment 4. Two sensor-based polynomials which link the perception of the robot to the desired route following behaviour.

on in region B, the front sensors of the robot starts dominating the behaviour and finally in region C the sensors on the left side of the robot become more important.

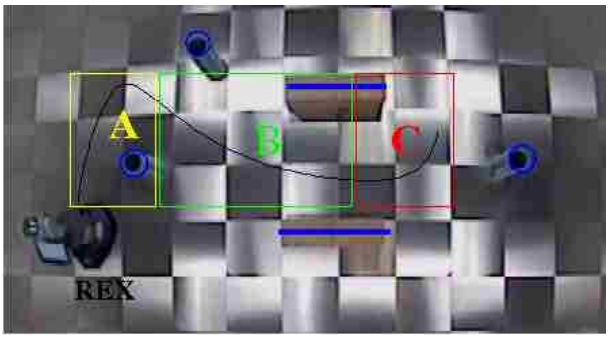


Fig. 31. Experiment 4: The relationship between the robot's perception and the desired actions is dependent on the position of the robot along the route.

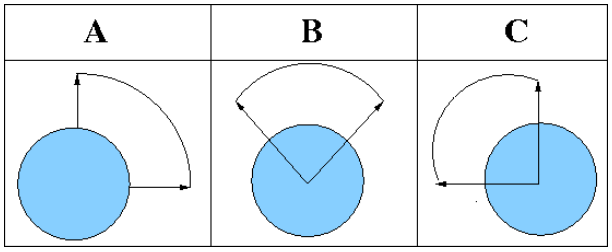


Fig. 32. Experiment 4: This figure shows how the contributions of the sensors around the robot to obtain the desired motor response changes along the desired route. In region A the sensors on the right side of the robot are more important than the rest. As the robot enters region B the sensors in front of the robot are more correlated with the desired motor response. Finally, in region C the sensors on the left side of the robot become more dominant for the desired motor response.

These results indicate that when the causal relationship between the perception and the desired motor responses of the robot varies along the desired route, trying to represent the whole relationship in a single polynomial may not be possible.

We are currently investigating how to address this problem. On straightforward method would be to represent complex, time-dependent sensor-motor tasks by a several polynomial models, using a classifier which divides the perception-action space of the robot into subspaces and generates a separate model for each subspace. This method has already been seen to work in initial experiments at Essex. We are, however, also investigating ways of representing complex tasks in *one* model; this is work in progress.

7. Conclusion and Future Work

This paper discusses robot experiments aiming at developing a formal, theory based design methodology to generate transparent robot control programs using mathematical functions. The research finds its theoretical roots in robot training and system identification techniques such as Armax (Auto-Regressive Moving Average models with eXogenous inputs) and Narmax (Non-linear Armax).

In the first set of experiments ([Nehmzow et al., 2005], [Kyriacou et al., 2006] and [Akanyeti et al., 2007a]), we operate the robot manually, using a joystick, to follow a desired trajectory. Once training data is acquired in this way, we use the NARMAX modelling approach to obtain a model which identifies a coupling between sensory perception and motor response as a polynomial model. This model is then used to control the robot autonomously.

In ([Nehmzow et al., 2007b] and [Akanyeti et al., 2007b]) we introduced a new mechanism to program robots — programming by demonstration — based on algorithmically translating observed human behaviours into robot control code, using transparent system identification techniques. To obtain such sensor-motor controllers, we first demonstrate the desired motion to the robot by walking in the target environment. Using this demonstration, we obtain recurrent, sensor free models that allow the robot to follow the same trajectory blindly. During this motion the robot logs its own perception-action pairs, which are subsequently used as training data for the Narmax modelling approach that determines the final, sensor-based models which identify the coupling between sensory perception and motor responses as non linear polynomials. These models are then used to control the robot.

Previously we used an external, camera-based motion tracking system to log the trajectory of the human demonstrator during his initial demonstration of the desired motion. However, besides being expensive, such tracking systems are complicated to set up and not always available.

We therefore improved our training method by replacing the *Vicon* motion tracking system with Narmax polynomial models which are trained to predict the position of the red jacket worn by the demonstrator using the robot's own vision system [Nehmzow et al., 2007a]. The statistical analysis showed that the obtained models are able to predict the position of the demonstrator in our laboratory with an accuracy of 10mm$\pm$3mm.

We compared the position prediction models obtained for two different markers, a red jacket and an orange ball. Even though the jacket is a non-rigid object, more accurate predictions can be made using it since it is spatially bigger than the ball — the larger objects make the position estimation models less sensitive to the noise introduced by the blob colouring algorithm.

Our experiments in modelling time-variant sensor motor tasks using a NARMAX system identification technique show that using a single model to represent complex sensor-motor tasks (such as entire long routes) may not be viable, and dividing perception-action spaces into subspaces may be needed.

7.1. *Future work*

Ongoing work in our laboratories addresses the following three issues:

Automatic parameter selection for NARMAX methodology Although the proposed method does not require any theoretical knowledge of robot programming, the need for model structure identification remains.We are interested to automate this process.

Object Tracking under occlusion So far our position estimation models only take the current image into account. This can be brittle in cases where the tracked object is either under occlusion or not visible in the image. We therefore are currently investigate methods of integrating extra information about the tracked object into the model (such as the previous position of the ball and the velocity of the ball), and expect improved performance in cases of occlusion.

Generating multi-polynomial sensor-motor couplings As discussed in the previous section, if the sensor-motor task under investigation is time-dependent or very complex, it may not be possible to represent the whole sensor-motor relationship in a single model. We therefore currently investigate ways of automatically clustering the perception-action space into subspaces, and generating a separate model for each of the subspaces.

Acknowledgments

We thank to Theocharis Kyriacou, Roberto Iglesias and Christoph Weinrich for their contributions to this work. We gratefully acknowledge that the RobotMODIC project was supported by the Engineering and Physical Sciences Research Council under grant GR/S30955/01.

References

[Akanyeti et al., 2007a] Akanyeti, O., Kyriacou, T., Nehmzow, U., Iglesias, R., and Billings, S. (2007a). Visual task identification and characterisation using polynomial models. *Robotics and Autonomous Systems*.

[Akanyeti et al., 2007b] Akanyeti, O., Nehmzow, U., Weinrich, C., Kyriacou, T., and Billings, S. (2007b). Programming mobile robots by demonstration through system identification. In *ECMR*, Freiburg, Germany.

[Allissandrakis et al., 2005] Allissandrakis, A., Nehaniv, C., Dautenhahn, K., and Saunders, J. (2005). An approach for programming robots by demonstration: Generalization across different initial configurations of manipulated objects. In *In Proceedings of the IEEE international symposium on computational intelligence in robotics and automation*, pages 61–66.

[Billings and Chen, 1998] Billings, S. and Chen, S. (1998). The determination of multivariable nonlinear models for dynamical systems. In Leonides, C., (Ed.), *Neural Network Systems, Techniques and Applications*, pages 231–278. Academic press.

[Billings and Voon, 1986] Billings, S. and Voon, W. S. F. (1986). Correlation based model validity tests for non-linear models. *International Journal of Control*, 44:235–244.

[Calinon and Billard, 2007] Calinon, S. and Billard, A. (2007). What is the teacher's role in robot programming by demonstration? toward benchmarks for improved learning. *Interaction studies, special issue on psychological benchmarksin human-robot interaction.*, 8:3 In press.

[Demiris and Hayes, 1996] Demiris, J. and Hayes, G. (1996). Imitative learning mechanisms in robots and humans. In *Proc. 5th European Workshop on Learning Robots*, pages 9–16, Bari, Italy.

[Demiris and Johnson, 2003] Demiris, Y. and Johnson, M. (2003). Distributed, predictive perception of actions: biologically inspired robotics architecture for imitation. *CONNECTION SCIENCE*, 15(4):231–243.

[Hayes and Demiris, 1994] Hayes, G. and Demiris, J. (1994). A robot controller using learning by imitation. In *Proc. 2nd Int. Symposium on Intelligent Robotics Systems*, pages 198–204, Grenoble, France.

[Iglesias et al., 2005] Iglesias, R., Kyriacou, T., Nehmzow, U., and Billings, S. (2005). Robot programming through a combination of manual training and system identification. In *Proc. of ECMR 05 - European Conference on Mobile Robots 2005*. Springer Verlag.

[Korenberg et al., 1988] Korenberg, M., Billings, S., Liu, Y. P., and McIlroy, P. J. (1988). Orthogonal parameter estimation algorithm for non-linear stochastic systems. *International Journal of Control*, 48:193–210.

[Kyriacou et al., 2006] Kyriacou, T., Akanyeti, O., Nehmzow, U., Iglesias, R., and Billings, S. (2006). Visual task identification using polynomial models. In *TAROS*, Guildford, UK.

[Nehmzow, 1995] Nehmzow, U. (1995). Applications of robot training: Clearing, cleaning, surveillance. In *Proc. of International Workshop on Advanced Robotics and Intelligent Machines*, Salford, UK.

[Nehmzow, 2003] Nehmzow, U. (2003). *Mobile Robotics: A practical introduction*. Springer Verlag, 2nd edition.

[Nehmzow et al., 2007a] Nehmzow, U., Akanyeti, O., Weinrich, C., Kyriacou, T., and Billings, S. (2007a). Learning by observation through system identification. In *TAROS*, Aberystwyth, Wales.

[Nehmzow et al., 2007b] Nehmzow, U., Akanyeti, O., Weinrich, C., Kyriacou, T., and Billings, S. (2007b). Robot programming by demonstration through system identification. In *IROS*, San Diego, USA.

[Nehmzow et al., 2005] Nehmzow, U., Kyriacou, T., Iglesias, R., and Billings, S. (2005). Self-localisation through system identification. In *Proc. of ECMR 05 - European Conference on Mobile Robots 2005*. Springer Verlag.

[Nguyen and Widrow, 1990] Nguyen, D. and Widrow, B. (1990). The truck backer-upper: An example of self-learning in neural networks. pages 287–299.

[Pardowitz et al., 2007] Pardowitz, M., Knoop, S., Dillmann, R., and Zoellner, R. (2007). Incremental learning of tasks from user demonstrations, past experiences and vocal comments. *IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART B*, 2:322–332.

[Pomerleau, 1993] Pomerleau, D. (1993). *Neural Network Vision for Robot Driving*.

[Schaal, 1997] Schaal, S. (1997). Learning from demonstration. In *Advances in Neural Information Processing Systems*, volume 9, pages 1040–1046.