Sensorimotor Learning for a Humanoid Robot Head

• José Santos •

VISLAB-RT-01-2010

Sensorimotor Learning for a Humanoid Robot Head

José Santos

December 2009

VISLAB-RT-01-2010

ISR — Torre Norte
Av. Rovisco Pais, 1
1049-001 Lisboa
PORTUGAL

1 Introduction

This work adresses the problem of sensorimotor learning for a humanoid robot head. We follow a paradigm where a robot, like human babies, comes to the world with few abilities and develops then with time. Of course, an organism cannot develop without some built in ability. However, if all abilities are built in the organism will not develop either. Nevertheless, the set of abilities that are present at birth is far from clear [4].

The most obvious way in which a child has been prepared for action is the design of its body. Each body part is part of a perception-action mechanism and was tailored by evolution in order to be so.

The problem of learning to control each body part is a very complex one. However, evolution has made it easier by providing several constraints (these constraints reduce the many degrees of freedom of the motor system). For instances, a child must learn how to control the arms before gaining control of the hands (in newborn infants when the arm moves, the fingers must move accordingly).

In this report we are interested in the process by which an infant senses gravity and learns to control its head. The idea is to provide a model of the direct rotation kinematics of the head and an algorithm which updates the model incrementally in the course of action. Clearly, we are assuming that some perceptual information is given as input. This is not an artificial claim, since perception must always preceed action [4]. We assume that the propreceptive (motor) and vestibular (inertial) senses are already developed. This report presents some experimental results concerning the application of an online incremental learning algorithm to the problem of learning ICub's rotation kinematics and motor-inertial map.

2 ICub Head Kinematics

The ICub joints are organized into six sub-systems: the head, left arm, right arm, torso, left leg and right leg. In this report, we shall only consider the head sub-system.

The Head sub-system has 6 joints in the standard configuration: neck pitch (tilt), neck roll (swing), neck yaw (pan), eyes tilt, eyes version and eyes vergence, as shown in figure 2. In this report, we are solely interested in the head movements: tilt, swing and pan.

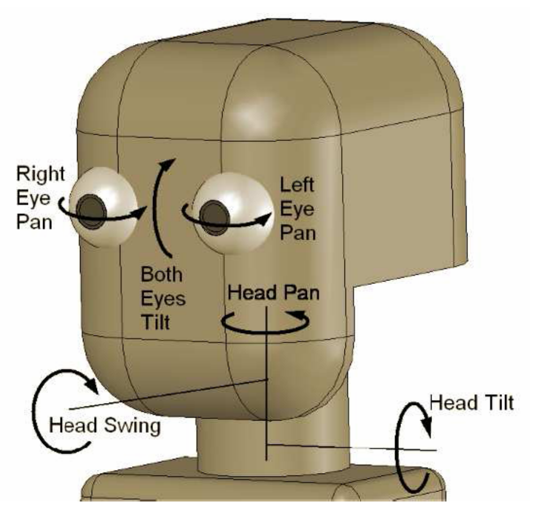


Figure 1: Illustration of the Head degrees of freedom.

It is important to establish a reference coordinate frame, which will be denoted by $\{0\}$. This reference coordinate frame corresponds to the body coordinate frame. This frame is defined by the local vertical and by the rotation of the body along this axis. Considering identical reference coordinate frames for all joints in the canonical state, the rotation matrix representing the head's orientation with respect to the body reference frame depends on the tilt, swing and pan angular displacements, and is given by:

$$\begin{aligned} {}^{0}\mathrm{R}_{3} &= {}^{0}\mathrm{R}_{1} \cdot {}^{1}\mathrm{R}_{2} \cdot {}^{2}\mathrm{R}_{3} \qquad (1) \\ &= \mathrm{ROT}_{y}(\theta_t) \cdot \mathrm{ROT}_{x}(\theta_s) \cdot \mathrm{ROT}_{z}(\theta_p) \qquad (2) \\ &= \begin{bmatrix} c_t & 0 & s_t \\ 0 & 1 & 0 \\ -s_t & 0 & c_t \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_s & -s_s \\ 0 & s_s & c_s \end{bmatrix} \cdot \begin{bmatrix} c_p & -s_p & 0 \\ s_p & c_p & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (3) \end{aligned}$$

with $c_t = cos(\theta_t)$, $s_t = sin(\theta_t)$, $c_s = cos(\theta_s)$, $s_s = sin(\theta_s)$, $c_p = cos(\theta_p)$ and $s_p = sin(\theta_p)$.

The main goal of this project is to provide an online incremental method for building a model of ${}^{0}\mathrm{R}_{3}$, that is: iCub is expected to learn its direct kinematics in an online incremental way.

The iCub has an inertial sensor which calculates the orientation between the sensor-fixed coordiante system, denoted by $\{S\}$, and an earth fixed coordinate system, denoted by $\{G\}$. By default the local earth-fixed reference co-ordinate system used is defined as a right handed Cartesian coordinate system with:

- $\hat{X}$ positive when pointing to the local magnetic North.
- $\hat{Y}$ according to the right handed coordinates (West).
- $\hat{Z}$ positive when pointing up.

Note that this coordinate system may not correspond to the body coordinate system (however, that can happen if the iCub is placed facing North). Moreover, the sensor fixed coordinate system does not correspond to the head fixed coordinate system (above denoted by $\{3\}$), because of the way the sensor is installed in iCub's head. This two coordinate systems are related acording to the following rotation matrix:

$$^{3}\mathrm{R}_{S} = \mathrm{ROT}_{z}(\pi) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore, one can say that the inertial sensor outputs the following rotation matrix:

$$^{G}\mathrm{R}_{S} = {}^{G}\mathrm{R}_{0} \cdot {}^{0}\mathrm{R}_{3} \cdot {}^{3}\mathrm{R}_{S}$$

Using the information provided by the motor encoders (the joint angle displacements) and the information provided by the inertial sensor($^{G}\mathrm{R}_{S}$), our goal is to propose an online incremental method for building a model of $^{0}\mathrm{R}_{3}$. However, in order to collect the appropriate data for builing the model, we are faced with two initial problems:

1. The inertial sensor outputs $^{G}\mathrm{R}_{S}$, from which one can easily compute $^{G}\mathrm{R}_{3}$ ($^{G}\mathrm{R}_{3} = {}^{G}\mathrm{R}_{S} \cdot {}^{S}\mathrm{R}_{3}$). However, how can we determine $^{3}\mathrm{R}_{0}$ from $^{G}\mathrm{R}_{3}$?
2. The angle displacements given by the motor encoders are measured with respect to the position in which the robot was turned on. Since the goal is to learn a model that can be stored and used again, one must estimate the initial position of the head with respect to a fixed reference frame; we will use $\{0\}$.

In order to solve the first problem it is important to develop a method to estimate ${}^{G}\mathrm{R}_0$. Note that:

$$ {}^{G}\mathrm{R}_3 = {}^{G}\mathrm{R}_0 \cdot {}^{0}\mathrm{R}_3 $$

So, considering the properties of rotation matrices:

$$ \begin{aligned} {}^{G}\mathrm{R}_0 &= {}^{G}\mathrm{R}_3 \cdot {}^{3}\mathrm{R}_0 && (4) \\ &= {}^{G}\mathrm{R}_3 \cdot {}^{0}\mathrm{R}_3^T && (5) \end{aligned} $$

We propose the following method for computing ${}^{G}\mathrm{R}_0$:

- When the iCub is turned on, its head is aligned with its body. That is, its head is put in a position such that: ${}^{0}\mathrm{R}_3 = I$.
- After aligning the head with the body, we know that:

  $$ {}^{G}\mathrm{R}_S = {}^{G}\mathrm{R}_0 \cdot {}^{3}\mathrm{R}_S $$

  So:

  $$ {}^{G}\mathrm{R}_0 = {}^{G}\mathrm{R}_S \cdot {}^{S}\mathrm{R}_3 $$

  Since ${}^{G}\mathrm{R}_S$ and ${}^{S}\mathrm{R}_3$ are both known, ${}^{G}\mathrm{R}_0$ is easily computed. We further note that ${}^{G}\mathrm{R}_0 = \mathrm{ROT}_z(\theta_0)$, so computing ${}^{G}\mathrm{R}_0$ is equivalent to computing θ_0.

Note that after aligning the iCub's head with its body, we can reset the encoders, hence solving the second problem mentioned above.

3 Aligning ICub's head with its body through the sense of Gravity

Considering equation 2 and having performed the required calculations, one can state that:

$$ \left({}^{G}\mathrm{R}_S\right)_{31} = \mathrm{r}_{31}^{GS} = s_t \cdot c_p - c_t \cdot s_s \cdot s_p \qquad (6) $$

$$ \left({}^{G}\mathrm{R}_S\right)_{32} = \mathrm{r}_{32}^{GS} = -s_t \cdot s_p - c_t \cdot s_s \cdot c_p \qquad (7) $$

The goal is to place the iCub's head in a position such that ${}^{0}\mathrm{R}_3 = I$. In such a position the following equalities must be verified:

$$ \mathrm{r}_{31}^{GS} = 0 \qquad (8) $$

$$\mathrm{r}_{32}^{GS} = 0 \tag{9}$$

In order to put the iCub's head in a position in which this two equations are verified, we only need to change the tilt and swing angular displacements. This problem is one of solving a system of nonlinear equations (the number of equality conditions is equal to the number of variables).

$$r(\theta_t, \theta_s) = \begin{bmatrix} r_{31}^{GS}(\theta_t, \theta_s) \\ r_{32}^{GS}(\theta_t, \theta_s) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

A popular way to solve this kind of problems is to use Newton's method for nonlinear equations [2]. Newton's method defines a linear model $M_k(\Delta\theta)$ of $r(\theta + \Delta\theta)$ in the following way:

$$r(\theta + \Delta\theta) = r(\theta) + J(\theta) \cdot \Delta\theta$$

where $J(\theta)$ denotes the jacobian of r evaluated in θ.

Newton's method in its pure form chooses the step $\Delta\theta$ to be the vector for which $M_k(\Delta\theta) = 0$, that is:

$$\Delta\theta = -J(\theta)^{-1} \cdot r(\theta)$$

However, in this case, since r is not known, the information required for computing the jacobian is not given. Hence, we will use Broyden's method [2]. Broyden's method is a secant method, it constructs its own approximation of the Jacobian, updating it at each iteration so that it mimics the behavior of the true Jacobian J over the step just taken.

The requirement that the approximate Jacobian should mimic the behavior of the true Jacobian can be specified as follows. Let s_k denote the step form θ_k to θ_{k+1} and let y_k denote the corresponding change in r, that is:

$$s_k = \theta_{k+1} - \theta_k$$

$$y_k = r(\theta_{k+1}) - r(\theta_k)$$

Broyden's method requires that the updated Jacobian approximation B_{k+1} to satisfy the following equation, which is known as the secant equation:

$$y_k = B_{k+1} \cdot s_k$$

which ensures that B_{k+1} and $J(x_{k+1})$ have similar behavior along direction s_k. The Broyden's method corresponds to the following update:

$$B_{k+1} = B_k + \frac{(y_k - B_k \cdot s_k) \cdot s_k^T}{s_k^T \cdot s_k} \tag{10}$$

The Broyden update makes the smallest possible change to the Jacobian (as measured by the Euclidean norm: $\|B_k - B_{k+1}\|_2$) that is consistent with the secant equation, which can be formally stated as:

$$B_{k+1} \in \underset{B \,:\, y_k = B \cdot s_k}{\arg\min} \|B - B_k\|$$

The specification of the algorithm is presented below.

Algorithm 1 Broyden's Method

```
Choose $\theta_0$ and a nonsingular initial Jacobian approximation $B_0$;
for $k = 0, 1, 2, \cdots$ do
    Calculate a solution $\Delta\theta_k$ to the linear equations $B_k \cdot \Delta\theta_k = -r(\theta_k)$
    $\theta_{k+1} \Leftarrow \theta_k + \Delta\theta_k$
    $s_k \Leftarrow \theta_{k+1} - \theta_k$
    $y_k \Leftarrow r(\theta_{k+1}) - r(\theta_k)$
    Obtain $B_{k+1}$ from formula 10
end for
```

After applying Broyden's algorithm it is reasonable to expect that the tilt and swing displacements are almost zero. So:

$$\theta_t^0 = -\sum_{k=1}^{n} \Delta\theta_t^k$$

$$\theta_s^0 = -\sum_{k=1}^{n} \Delta\theta_s^k$$

Using these values one can easily compute the initial pan displacement θ_p^0 using equations 6 and 7:

$$s_p^0 = \frac{b \cdot r_{31}^0 - a \cdot r_{32}^0}{a^2 + b^2} \tag{11}$$

$$c_p^0 = \frac{r_{31}^0 - b \cdot s_p^0}{a} \tag{12}$$

with $a = s_t$ and $b = -c_t \cdot s_s$. Therefore:

$$\theta_p^0 = atan2(s_p^0, c_p^0)$$

4 Learning a Motor-Inertial Map

Receptive Field Weighted Regression [6] is an algorithm designed in order to be used in an incremental online way.

The RFWR algorithm is used to build a model of ${}^3\mathrm{R}_0$, that is a function $F : R^3 \to R^9$, such that:

$$F(\theta_{tilt}, \theta_{swing}, \theta_{pan}) = \mathrm{vec}\left({}^3\mathrm{R}_0(\theta_{tilt}, \theta_{swing}, \theta_{pan})\right) \tag{13}$$

Where vec corresponds to the vectorization operator.

4.1 RFWR: Incremental Online Learning

The goal of RFWR is to construct a system of receptive fields for incremental function approximation. A prediction $\hat{y}$ for a query point x is built from the normalized weighted sum of the individual predictions $\hat{y}_k$ of all receptive fields:

$$\hat{y} = \frac{\sum_{k=1}^{K} w_k \hat{y}_k}{\sum_{k=1}^{K} w_k} \tag{14}$$

The weights w_k correspond to the activation strenghts of the corresponding receptive fields.

The weights w_k are computed using a gaussian kernel:

$$w_k = exp\left(-\frac{1}{2}(x - c_k)^T D_k (x - c_k)\right) \tag{15}$$

where $D_k = M_k^T M_k$ and M_k is an upper diagonal matrix, which ensures that the positive definiteness of D_k (D_k is a distance metric).

Each receptive field corresponds to a local linear model.

$$\hat{y}_k = (x - c_k)^T b_k + b_{0,k} \tag{16}$$

Defining $\tilde{x} = \left((x - c_k)^T, 1\right)^T$ and $\beta_k = \left(b_{0,k}, b_k^T\right)^T$, it is possible to rewrite equation 16 as:

$$\hat{y}_k = \tilde{x}^T \beta_k \tag{17}$$

Since RFWR is an incremental online learning method, the update of the parameters of each linear model (β_k) has to be done incrementally. Nevertheless, it is important to understand that for each linear model, β_k could be computed explicitly using a locally weighted linear regression. Instead, this update is computed by recursive least squares.

Adjusting the shape and size of the receptive field is accomplished by adjusting the distance metric $D = M^T M$. This update is computed by gradient descent in order to minimize the leave-one-out cross validation criteria:

$$J = \frac{1}{W} \sum_{i=1}^{p} w_i \left\| y_i - \hat{y}_{i,-i} \right\|^2 \tag{18}$$

Where $\hat{y}_{i,-i}$ denotes the prediction of the i-th data point calculated when training the learning system with the i-th data point excluded from the training set. Without storing data in incremental learning, we cannot use cross validation and, thus, we cannot obtain the true gradient. Therefore, a stochastic approximation of the gradient is derived. As such, given a new training point (x, y), the algorithm will compute a stochastic approximation of the gradient and then perform the steepest descent step according to:

$$M^{n+1} = M^n - \alpha \cdot \frac{\partial J}{\partial M} \tag{19}$$

where α denotes the learning rate.

Lastly, when using RFWR one must establish two important thresholds: w_{gen} and w_{prune}. Given a new training point x_q, if no receptive field is activated by more than w_{gen}, a new receptive field is created with center x_q. The idea is to create a new receptive field each time a training sample does not activate any of the existing receptive fields by more than a threshold w_{gen}. Conversely, if two receptive fields overlap too much, one of them must be pruned. The idea is to prune the one which has a larger covariance matrix.

4.2 Learning a Rotation Matrix

When using RFWR, building a model for $F = (f_1, \cdots, f_9)$ 13 is equivalent to building 9 different models independently. However, after learning the whole model, given a query point $\vec{\theta}_q$, the model's prediction $\hat{R}_q$ may not correspond to a rotation matrix. In this situation we choose to project $\hat{R}_q$ onto the subspace of rotation matrices.

So, giving a query point $\vec{\theta}_q$, the learned model is used to obtain a prediction $\hat{R}_q$. Then, one has to solve the following problem:

$$\min_{R \text{ is a rotation matrix}} \left\| R - \hat{R}_q \right\| \tag{20}$$

Consider the singular value decomposition (SVD) of matrix $\hat{R}_q$, given by:

$$\hat{R}_q = UDV^T$$

The solution of the optimization problem 20 corresponds to $R = UV^T$.

Therefore, to make a prediction one must apply the following steps:

1. Use the learned model to compute a prediction $\hat{R}_q$.
2. Compute its singular value decompostion (SVD) $\hat{R}_q = UDV^T$.
3. Ouput UV^T.

5 Experimental Results

5.1 Inertial Sensor Noise Characterization

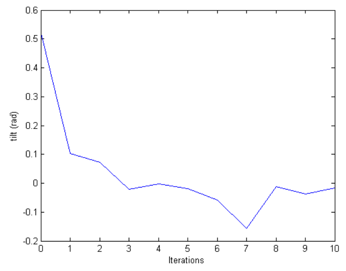
As was stated in section 2, the inertial sensor outputs a rotation matrix, ${}^{G}\mathrm{R}_{S}$, which describes the orientation of the sensor fixed co-ordinate system, $\{S\}$, with respect to the earth fixed co-ordinate system, $\{G\}$. Naturally, the information provided by this sensor includes noise. Therefore, in order to simulate it properly one needs to characterize the noise variance.

We have evaluated the sample variance of the rotation matrix provided by the inertial sensor in a wide range of positions. In each position we recorded 100 samples and then computed the sample variance. The maximum variance registered was 0.00334233. Hence, in each of the simulations presented in this section we shall assume a zero-mean gaussian white noise with variance 0.0034.

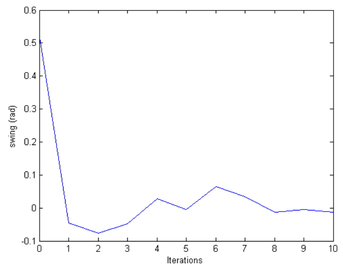
5.2 Aligning ICub's head with its body

The Broyden's Method as described in algorithm 1 was implemented in Matlab in order to assess the way it converges when applied to the problem of aligning the ICub's head with its body. We assume that initially each joint angle (tilt, swing and pan) is reasonably close to 0 with respect to the body reference frame (every joint angle is assumed to be lower than 30°).

Broyden's method is applied in order to find a position such that: $\theta_{tilt} = 0$ and $\theta_{swing} = 0$. After finding the corresponding initial values, one can easily compute the initial pan displacement. As such, the simulations presented in figures 2 to 5 illustrate the way θ_{tilt} and θ_{swing} are changed during the application of the algorithm. In each simulation we only consider ten iterations since we expect this method to converge quickly.

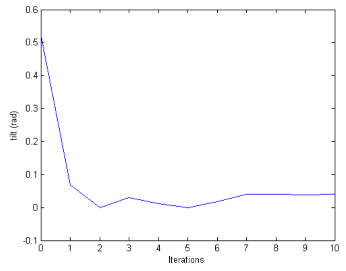


(a)

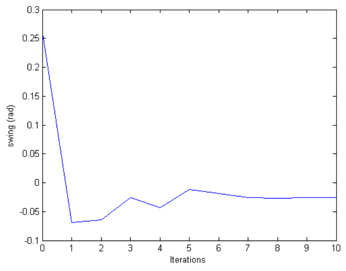


(b)

Figure 2: $\theta_{tilt} = \frac{\pi}{6} \quad \theta_{swing} = \frac{\pi}{6}$

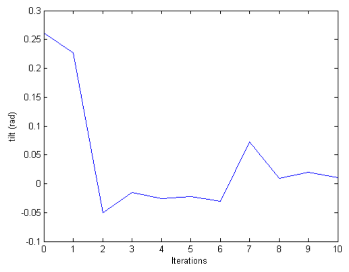


(a) $\theta_{tilt} = \frac{\pi}{6} \quad \theta_{swing} = \frac{\pi}{12}$

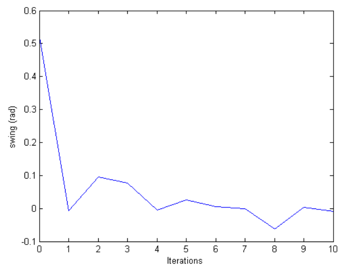


(b)

Figure 3: $\theta_{tilt} = \frac{\pi}{6} \quad \theta_{swing} = \frac{\pi}{12}$

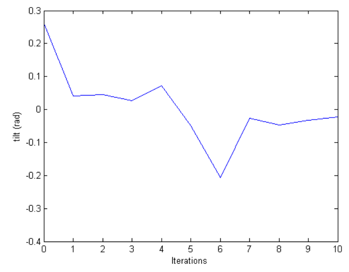


(a)

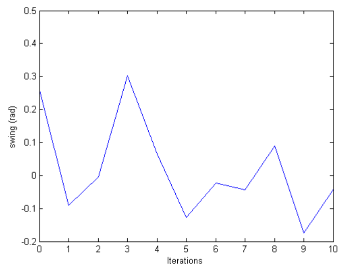


(b)

Figure 4: $\theta_{tilt} = \frac{\pi}{12} \quad \theta_{swing} = \frac{\pi}{6}$



(a)



(b)

Figure 5: $\theta_{tilt} = \frac{\pi}{12} \quad \theta_{swing} = \frac{\pi}{12}$

Broyden's algorithm was successfully implemented on Chica. This algorithm was applied to the problem of calibrating the motor encoders. Therefore, it is quite difficult to evaluate its results in practice, since the real zero is not known. Nevertheless, we can illustrate the application of the algorithm by showing how the entries of the rotation matrix provided by inertial sensor change during the corresponding application. When the head of the robot is aligned with its body, the orientation matrix provided by the inertial sensor must be a rotation about the Z axis. So entries r_{31}, r_{32}, r_{13} and r_{23} must be zero and entry r_{33} must be 1. Table 5.2 presents the evolution of these entries when Broyden's algorithm is applied to the real robot. Naturally, it is only possible to estimate the initial joint angles after applying the algorithm and assuming that the head is then aligned with the body.

Table 1: $\theta^0_{tilt} = 0.7561rad \quad \theta^0_{swing} = 0.3047rad \quad \theta^0_{pan} = 0.5927rad$

Iterations	r_{31}	r_{32}	r_{13}	r_{23}	r_{33}
0	0.448124	-0.565588	-0.365259	-0.622327	0.692311
1	0.404702	0.03108	0.220109	-0.34103	0.91392
2	0.030612	0.262968	0.250975	0.084264	0.964319
3	-0.138035	0.059402	-0.008302	0.150045	0.988644
4	-0.064814	-0.097917	-0.116638	0.013573	0.993082
5	0.054952	-0.052626	-0.019847	-0.073453	0.997101
6	0.042871	0.023935	0.041518	-0.02621	0.998794
7	0.003065	0.041944	0.038273	0.017432	0.999115
8	-0.029368	-0.002641	-0.016464	0.024462	0.999565
9	0.003777	-0.005103	-0.002637	-0.005775	0.99998

5.3 RFWR - Tuning Parameters

In order to use the RFWR algorithm properly one has to tune several parameters. In this report, we shall only consider the initial distance metric $D_0 = M_0^T M_0$. For all the other parameters we have used the values sugested in [5].

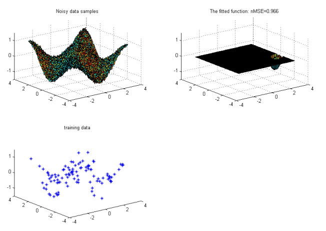
It was experimentally verified that the initial choice for the distance metric D_0 plays a major role in controlling the creation of new receptive fields (at least in the beginning of the learning process). This is quite relevant since the creation of many receptive fields will damage the smoothness of the learned model. Nevertheless, the creation of new receptive fields is a fundamental part of the RFWR algorithm because it allows it to model highly non-linear functions. Given a query point $\vec{\theta}_q$, the activation strength of each receptive field is given by a gaussian kernel. In order to get a larger receptive field, one must increase the diagonal values of the corresponding covariance matrix.

As discussed in section 4.2, our goal is to use the RFWR algorithm to build a model of ${}^3R_0$ as a function of the joint angle displacements (according to formula 13). Clearly, not all entries of ${}^3R_0$ depend on the three angular displacements. For instance, $({}^3R_0)_{11}$ does not depend on θ_{tilt}:

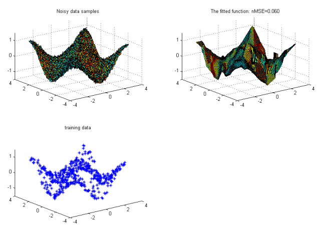
$$\left({}^3R_0\right)_{11} = \cos\theta_{swing} \cdot \cos\theta_{pan}$$

Therefore, we will start by analysing the behaviour of the algorithm in the two-dimensional case, considering the following function:

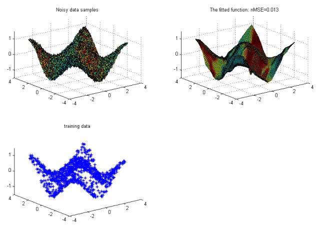
$$r_{11}(\theta_s, \theta_p) = \cos\theta_s \cdot \cos\theta_p$$



(a) 100 training samples



(b) 500 training samples



(c) 1000 training samples

Figure 6: $[\pi/18 \quad 0.01; \; 0.01 \quad \pi/18]$

The results corresponding to 3 different initial distance metrics:

$$\begin{bmatrix} \pi & 0.01 \\ 0.01 & \pi \end{bmatrix} \qquad \begin{bmatrix} \frac{\pi}{9} & 0.01 \\ 0.01 & \frac{\pi}{9} \end{bmatrix} \qquad \begin{bmatrix} \frac{\pi}{18} & 0.01 \\ 0.01 & \frac{\pi}{18} \end{bmatrix}$$

are presented in figures 6 to 8. In each case, a set consisting of 100000 query points was generated, from this set 3 different sets consisting of 100, 500 and 1000 query points, respectively, were randomly selected. The algorithm was then tested on each of these sets.

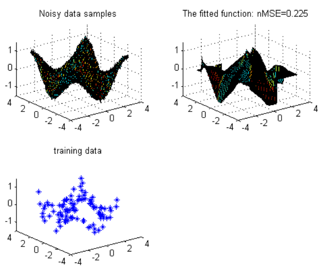
5.4 RFWR - Learning the whole model

This section presents two simulations in which the RFWR algorithm is used to model ${}^3R_0$ as a function of the joint angles.

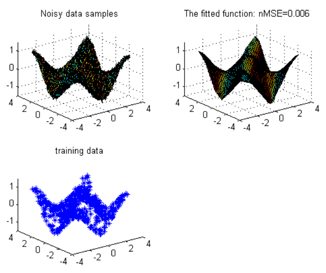
In the first simulation a test set consisting of 1000 was generated. Ten test sets with different dimensions were randomly selected from the original test set. The results are presented in table 2.

In the second simulation the algorithm was applied to real data. There were two major problems concerning the acquisition of real data:

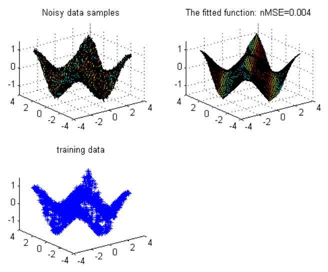
1. The inertial sensor and the motor encoders are not synchronized.



(a) 100 training samples



(b) 500 training samples



(c) 1000 training samples

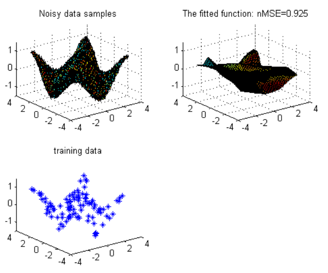
Figure 7: $[\pi/9 \quad 0.01 \; ; \; 0.01 \quad \pi/9]$

2. The information provided by the inertial sensor includes noise.

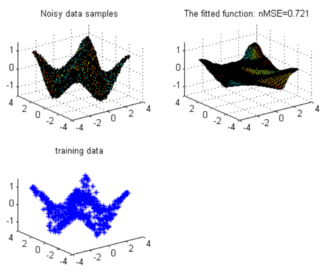
The ICub's head was put in each position for a fixed period of time. During this period all data provided by the inertial sensor was recorded. The rotation matrix associated with the position in which the ICub's head was put corresponds to the arithmetic mean of the recorded set. Results are presented in table 3.

6 Conclusions and Future Work

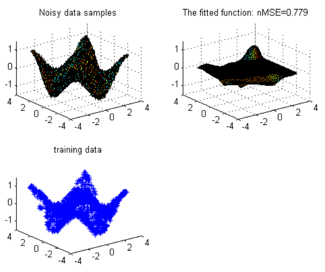
In order to build a model which establishes a relation between the orientation of the head of the robot (with respect to the body coordinate system) and the joint angles, one must start by calibrating the motor encoders. Results concerning the application of Broyden's algorithm to this particular problem show that it generally converges to a very close value in less than ten iterations. Therefore, it performs as well as it is required.



(a) 100 training samples



(b) 500 training samples



(c) 1000 training samples

Figure 8: $[\pi \quad 0.01\ ;\ 0.01 \quad \pi]$

The results obtained after applying the RFWR algorithm were mildly good. However, we feel that different methods should be tried, for instances: sparse online gaussian processes [3]. This second method is known to perform better in the presence of smaller amounts of training data. Moreover, the RFWR algorithm has such an amount of tuning parameters that it becomes very hard to find the setting for which it presents better results. Nevertheless, it is important to stress the qualities of this algorithm: particularly, the fact that it is fairly simple to compute the gradient of the obtained model.

References

[1] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *AI Review*, 11:11–73, April 1997.

[2] G. Broyden. A class of methods for solving non-linear simultaneous equations. *Mathematics of Computation*, 19:577–593, October 1965.

Table 2: Evaluation of the RFWR algorithm on randomly generated data

Training Instances	Mean Square Error								
	r_{11}	r_{12}	r_{13}	r_{21}	r_{22}	r_{23}	r_{31}	r_{32}	r_{33}
100	0.40228	0.32540	0.18200	0.21063	0.19953	0.21100	0.34161	0.40272	0.25554
200	0.17004	0.16433	0.11591	0.11455	0.11574	0.07190	0.27037	0.21320	0.11476
300	0.13430	0.14538	0.07695	0.07963	0.07986	0.04563	0.14544	0.17820	0.07369
400	0.10036	0.11263	0.07387	0.06402	0.04002	0.03292	0.14834	0.13129	0.05835
500	0.08064	0.10402	0.06157	0.05715	0.02459	0.02070	0.10376	0.10523	0.05694
600	0.06435	0.10953	0.05627	0.06113	0.02186	0.01551	0.08586	0.09448	0.04296
700	0.06683	0.11134	0.05030	0.04129	0.02196	0.01330	0.07796	0.09177	0.03136
800	0.05711	0.09400	0.04099	0.03697	0.01646	0.01385	0.07638	0.09260	0.02404
900	0.05078	0.07658	0.03646	0.03671	0.01392	0.01032	0.08355	0.08850	0.02133
1000	0.04771	0.07845	0.03483	0.03326	0.01223	0.00906	0.08012	0.08995	0.02434

Table 3: Evaluation of the RFWR algorithm on real data

Training Instances	Mean Square Error								
	r_{11}	r_{12}	r_{13}	r_{21}	r_{22}	r_{23}	r_{31}	r_{32}	r_{33}
100	0.00527	0.10341	0.09361	0.09361	0.00413	0.10453	0.10339	0.09475	0.00413
200	0.00527	0.10341	0.09361	0.09361	0.00413	0.10453	0.10339	0.09475	0.00413
300	0.00527	0.10055	0.09361	0.09075	0.00413	0.10063	0.09988	0.09223	0.00413
400	0.00526	0.10047	0.09011	0.09068	0.00413	0.10191	0.09995	0.09212	0.00413
500	0.00526	0.10045	0.09010	0.09137	0.00413	0.10207	0.09983	0.09214	0.00413
600	0.00526	0.10064	0.09003	0.09075	0.00412	0.10212	0.09989	0.09203	0.00413
700	0.00526	0.10073	0.09008	0.09064	0.00412	0.10202	0.09989	0.09220	0.00412
800	0.00526	0.10074	0.09009	0.09080	0.00412	0.10182	0.09979	0.09222	0.00412
900	0.00526	0.10070	0.09009	0.09071	0.00412	0.10182	0.09987	0.09218	0.00412
1000	0.00526	0.10061	0.09009	0.09065	0.00412	0.10174	0.09987	0.09222	0.00412

[3] L. Csato and M. Opper. Sparse online gaussian process. *Neural Computation*, 14:641–668, 2002.

[4] C. Hofsten. Roadmap for the development of cognitive capabilities in humanoid robots. Technical report, January 2002.

[5] S. Kankle and S. Vijayakumar. A library for locally weighted projection regression. Technical report, March 2008.

[6] S. Schaal and C. Atkeson. Receptive field weighted regression. Technical Report Tech. Rep. TR-H-209, 1997.

[7] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, December 2005.