Military Technical College
Kobry El-Kobbah,
Cairo, Egypt

6th International Conference
on Electrical Engineering
ICEENG 2008

Achieving real-time Intelligent performance in underwater robotics using cheap off-the-shelf components

By

Ali Ibraheem, Bilal Hassanein, Saad Sharaf, Mohi Ahmad, Mahmoud Magdy*

Mohamed Abdelhady, Hamada Attia

Abstract:

The present paper proposes a new simple strategy to build underwater robots using cheap off-the-shelf components. The proposed strategy successfully achieves real-time operation without requiring expensive special purpose hardware/software. The simple geometry of the proposed design makes it possible to derive an accurate dynamic model of the robot. The dynamic model, hardware/software preprocessing of webcam images, multi-feature Kalman based combining as well as neural networks are used to compensate for the limited sensors and computational resources.

Keywords:
Underwater robotics, Kalman, Neural Networks, Behavior Based Robotics.

*All authors are at Alexandria University, Faculty of Engineering

1. Introduction:

Underwater robots have many important applications, including pipeline inspections, underwater environment exploration (they act as active mobile sensors) as well as rescue missions. They have the advantages of accurate operation as well as saving human operators from working in dangerous environments. However, to be able to successfully perform the required tasks in such a complex environment, they need special purpose hardware and software. This prevents their wide spread in countries with low financial resources. In contrast, to the conventional expensive and complicated designs, the present paper proposes a simple practical strategy to build an underwater robot from cheap off-the-shelf components but at the same time achieves real-time performance. The strategy is demonstrated through building an underwater robot called Misk. Misk achieves the required compromise between performance and cost as follows. First, the robot structure combines design principles common in small robots with those adopted in ship design. The robot is divided into carefully chosen stations then the skeleton of the robot is covered with a material called fiber which is cheap and easy to handle. A careful construction process (whose details are included in the paper) is adopted to combine the advantages of low cost while maintaining the required strength for the robot to support the on-board devices. The robot shape is chosen to be an ellipsoid. This has the advantages of a stream-lined low drag body as well as having a closed-form mathematical dynamical model based on Kirchoff- equations [1]. Thus, a reliable simulator is easily constructed to accurately predict the robot motion. The sensors used by the robot are a webcam and an electromagnetic sensor. The central processing unit is based on a Pentium PC operating in a Windows XP environment and 1 GB harddisk. Scilab programming language (a free Matlab-like software) is used.

To compensate for the limited computational resources as well as limited sensors used, the following strategy is adopted. Most of the processing and decision making steps are based on look-up tables or neural networks. These are used to store behavioral primitives. For example, the simple geometry of the robot and the available accurate simulator makes it possible to design simple motion primitives off-line. These primitives are then stored on board of the robot. To assist the robot in following any desired path despite its simple propulsion system and low computational resources (low speed, low memory) borrowing from nature, the robot moves in a discontinuous manner, in the form of steps. The target velocity in trajectory planning is, thus, zero (to stop the robot when the step is finished), which greatly simplifies the controller design and implementation.

Target recognition is one of the main functions of an underwater robot. Despite the low sensors resolution and low computational resources, the robot can successfully achieve this task using a multi-criterion (feature) Kalman-like decision making strategy [8] which combines decisions based on several simple fast easy-to-calculate features. Moreover, the simulator can be used as an additional sensor to predict the robot position and compare it with the measured position. In addition to that, using lens correction as well as pre–processing of the obtained images compensates for the low webcam resolution as well as distortion due to underwater effects.

An incremental learning strategy can be used to provide a stable continuously improving performance during the robot operation in its environment. The paper also proposes a simple system to teleoperate the robot over the Internet. This enables an operator to monitor the robot mission and interfere when necessary.

2. Robot Structure and Fabrication

The body of the robot is an ellipsoid. Fig. 1 shows the steps of Misk fabrication. The hull of the robot is made of several cheap materials (zinc powder, drier catalyst and fiber. First a mould which is to be painted with fiber is made. An ellipsoid is cut out of paper. Several stations are made and a mixture of cement and tartar is used to fill in the gaps. After the mould has dried, we combine a certain amount of zinc powder and drier catalyst. Fiber is distributed uniformly over such material. Prior to painting the mould with such a mixture, the mould is painted with oil or honey not to stick with the painter mixture. The mould is painted for several layers. It takes about two hours to dry after which the mould can be removed. Steel is used to make a skeleton of stiffeners. This makes it possible to fix components to the hull and makes it more stiff.

3.Information Processing and Control

a. Robot Dynamic Model and Control Law

The dynamics of an ellipsoid can be accurately represented by Kirchoff formulas [1]:

$$\begin{aligned} J\dot{\omega} &= J\omega \times \omega + Mv \times v + T \\ M\dot{v} &= Mv \times \omega + F \end{aligned} \qquad (1)$$

J is the summation of the inertia and added inertia matrix.
M is the summation of the mass and added mass matrix.
T is the input actuation torque vector.
F is the input actuation force vector
ω is the angular velocity vector.
v is the linear velocity vector.

For an ellipsoid the added mass and added inertia matrices are diagonal. The elements on the diagonal are calculated as follows.

$$m_i = \rho V\left(1 + \frac{\gamma_i}{2 - \gamma_i}\right) \qquad (2),$$

where ρ is the density of the fluid

$$\gamma_i = l_1 l_2 l_3 \int_0^\infty \frac{d\lambda}{(l_i{}^2 + \lambda)\Delta}$$ l_1, l_2, l_3 are robot axes

$$\Delta = \sqrt{(l_1{}^2 + \lambda)(l_2{}^2 + \lambda)(l_3{}^2 + \lambda)}$$

and the inertia matrix terms are calculated as follows:

$$j_1 = \frac{1}{5}\rho V\left(l_2^{\ 2} + l_3^{\ 2} + \frac{\left(l_2^{\ 2} - l_3^{\ 2}\right)^2 (\gamma_3 - \gamma_2)}{2\left(l_2^{\ 2} - l_3^{\ 2}\right) + \left(l_2^{\ 2} + l_3^{\ 2}\right)(\gamma_3 - \gamma_2)} \right) \qquad (3),$$

j_2, j_3 can be calculated, similarly

b. The Pre-computed Torque Controller

The accurately predicted non-linear dynamics promotes the use of precomputed torque controller. The control law used by the robot is an adapted version of that commonly used by several authors [2] in order to suit the Kirchoff-based formulation of the system dynamics.

$$\begin{aligned} T &= J\dot{\omega}_d + k_\theta \text{error}_\theta + k_\omega \text{error}_\omega - J\omega \times \omega - Mv \times v \\ F &= -Mv \times \omega + k_p \text{error}_p + k_v \text{error}_v + M\dot{v}_d \end{aligned} \qquad (4)$$

$\text{error}_\theta, \text{error}_\omega, \dot{\omega}_d, \text{error}_p, \text{error}_v, \dot{v}_d$ are the angular position error, angular velocity error, desired angular acceleration, position error, linear velocity error, desired linear acceleration, respectively. It is clear that this law successfully cancels nonlinearity and guarantees the convergence of the tracking error to zero.

The control law will be used to construct a motion library, which is prepared off-line and then downloaded to the robot onboard computer. The notion of motion library has been used successfully by several authors [3], [4]. It has the advantages of offering real-time performance despite the limited, relatively slow computational resources. This will be elaborated further in the following section. Fig. 2 shows an example of using pre-computed torque controller to steer the robot to a desired position.

c. Goal Oriented Task Decomposition (Hierarchical-Behavior-Based Architecture)

Misk was designed to participate in the International underwater vehicle contest [2]. In general, several approaches may be adopted in robot system design. The best way to identify the basic behaviors required by Misk is to decompose the mission in details. Based on such a decomposition the basic behaviors are identified and designed independently. This greatly facilitates the design procedure. Consider the following mission decomposition. The robot reads the image and electromagnetic sensor data. The image is preprocessed to filter out underwater effects. The robot defines the position of the light source and plan a path towards it. The robot executes the path one step at a time. After each step (20 cm), the robot stops and recollects the sensory information. This has several advantages. It eliminates the jerkiness in the image which would otherwise occur if the robot would capture the image while moving. Moreover, it greatly simplifies the stabilization of the robot path despite the delays of the slow available on-board computer. After the robot plans a desired step. The electromagnetic sensory information is checked to see if the move is allowed or not. Similarly, the image is processed to check if the move is allowed. Both decisions are fused and accordingly either the step is executed or another step is sought in case of the presence of an obstacle. To execute a move, the corresponding forces and torques are retrieved and the thrusters are allocated accordingly. This scenario is repeated until the robot reaches a light source, after which it plans a docking move to enter through a validation gate. The only difference however, is that in consequent steps, the prediction of the expected target location based on the robot simulator after the initial move is also fused with the target location computed through image processing, so as to get a better estimate of the target (light source) location. According to such a mission decomposition, the essential behaviors are identified as follows:

Data_collect This module consists of two parts. The first is implemented in scilab using a video processing toolbox (SIVP) that captures and stores a single frame of video upon request. The second part inputs the electromagnetic (EM) sensor reading to the computer through a PIC microcontroller [5]. The PIC microcontroller has special ports for Analog to Digital Conversion which will be used to convert the analog reading of the sensor into a digital signal. This digital signal may be input the PC serial port and read using another special Scilab command. The interested reader is referred to [5] for details of PC/PIC interface.

Image_Preprocess This module consists of two parts. The first part is implemented in hardware where a lens correction is employed to compensate for the underwater distortion. Fig. 3 illustrates the idea. The camera is placed inside a box on the axis of a lens. The distance between the lens and the camera is adjusted by trial and error until a satisfactory image is obtained. The lens used is convex so that the image is modified such that the external surface is straight for the underwater case (the liquid lens). Fig 3 shows a sample image taken after lens correction.

The second part of the module is implemented in software [6]. The procedure is as follows. Considering the illumination-reflectance model, we assume that an image is a function of the product of the illumination and the reflectance as described by equation:

$$f(x,y) = i(x,y) * r(x,y) \tag{5}$$

where f(x, y) is the image sensed by the camera, i(x, y) the illumination multiplicative factor, and r(x, y) the reflectance function. If we take into account this model, we can assume that the illumination factor changes slowly through the view field, therefore it represents low frequencies in the Fourier transform of the image. On the contrary reflectance is associated with high frequency components. By multiplying these components by a high-pass filter we can then suppress the low frequencies i.e the non uniform illumination in the image. The algorithm can be decomposed as follows :

• Separation of the illumination and reflectance components by taking the logarithm of the image . The logarithm converts the multiplicative effect into an additive one.

$$g(x,y) = \ln f(x,y) = \ln i(x,y) + \ln r(x,y) \tag{6}$$

• Computation of the Fourier transform of the log-image.

• High-pass filtering. The filter applied to the Fourier transform decreases the contribution of low frequencies (illumination) and also amplifies the contribution of mid and high frequencies (reflectance), sharpening the edges of the objects in the image

$$S(\omega_x,\omega_y)=H(\omega_x,\omega_y)I(\omega_x,\omega_y)+H(\omega_x,\omega_y)R(\omega_x,\omega_y),\ \omega_x=\frac{2\pi x}{m},\ \omega_y=\frac{2\pi y}{n} \quad (7)$$

$$H(\omega_x,\omega_y)=(r_{H-}r_L)(1-\exp(\frac{-(\omega_x+\omega_y)^2}{2\delta^2{}_w})+r_L \quad (8)$$

where rH = 2.5 and rL = 0.5 are the maximum and minimum coefficients values and δ_W factor which controls the cutoff frequency. These parameters are selected empirically. Finally, the inverse Fourier transform is computed to come back in the spatial domain and then taking the exponent to obtain the filtered image. Since these image preprocessing steps are computationally demanding, to cope with the limited computational resources while still maintaining real time operation, a feedforward artificial neural network (ANN) is used [7]. The ANN is trained on pairs of preprocessed images. The input to the ANN are blocks of the image before preprocessing and the target is the preprocessed image. Thus, after training the ANN is able to output a preprocessed version of a distorted input block without having to go through the tedious computations involved. Note that all feedforward networks used have the architecture shown in Fig.4. The number of hidden neurons is initially estimated to be 2n+1, where n is the number of inputs and is later fine tuned (increased) until a satisfactory error is achieved.

Target_locate This module consists of two submodules *image_locate* and *Correspond*

The submodule *image_locate* is a multi-feature/multi-scale based submodule which has two modes of operation :a training mode and a deployment mode. The training mode is described as follows. An image of the target taken at the distance at which the robot is considered to have reached the target is considered as a template. The image is manually cropped to include the target only. The cropped image size mxn is stored, so that in the future images are scaled up to this size. The features are computed for the template and stored for comparison in the deployment phase. The chosen features are the standard deviation and average, as well as the location of the highest eleven Fourier coefficients.

The deployment mode operation is described by the following steps: For each of the two feature vectors (mean –standard deviation and highest Fourier coefficients locations)

1. Divide the input image into blocks of size mxn (standard size of template). The image will be divided using 50% sliding.
2. For each block of the sliding blocks, compute the features and compare them based on Euclidean distance normalized by the stored template norm with those features computed for the target template.
3. The result of comparison is stored in the form of tuple $(x_{s1}, y_{s1}, d_{\min s1})$. This tuple records the coordinates of the best-matching block as those of the target.

4. Divide the image into blocks of size 0.5*mx0.5*n, with 50 % sliding. Resize each block up to mxn and then repeat steps 2 and 3. Store the result in the form $(x_{s2}, y_{s2}, d_{\min s2})$.
5. Repeat for block sizes 0.25*mx0.25*n, to get $(x_{s3}, y_{s3}, d_{\min s3})$

A special module will combine these estimates to decide the most probable target location.

The choice of features is crucial to the success of the module. To judge whether a feature is appropriate or not, two measures were consulted $\dfrac{\text{feature standard deviation for images of different targets}}{\text{feature average for images of different targets}}$, $\dfrac{\text{feature average for different images of the same target}}{\text{feature standard deviation for different images of the same target}}$.

These measure should be large to minimize the false acceptance and false rejection rates, respectively.

Correspond sub-module is implemented using a feedforward ANN. It takes as input the x, y and scale at which the target is proposed to be present and outputs the corresponding x,y,z coordinates in the physical robot-centered coordinate systems.

*Multi-fuse_locate: **T**he* input to this module is the target location estimate based on Fourier coefficient feature at different scales (block sizes) (x_{fs1}, y_{fs1}), (x_{fs2}, y_{fs2}), (x_{fs3}, y_{fs3}) the estimate based on mean and standard deviation (x_{ms1}, y_{ms1}), (x_{ms2}, y_{ms2}), (x_{ms3}, y_{ms3}) and the expected target location based on the simulator. A Kalman filter (KF) [8] is used to fuse these estimates in to the most probable target location. The KF consists of two main equations: the motion model (which describes the system dynamics) and the observations model (which relates the measured features to those required to be estimated). Since the robot is restricted to move one step at a time, the system motion model can be expressed as:

$$\begin{aligned} x_k &= x_{k-1} + \Delta x + wx_k \\ y_k &= y_{k-1} + \Delta y + wyk \end{aligned} \tag{9}$$

Where $\Delta x = step * \cos\theta, \Delta y = step * \sin\theta$. Note that when the path planning module chooses a suitable step from the pre-stored motion library, the associated motor inputs are retrieved and executed and the robot is presumed to finish the step in a pre-estimated timing. However, there are practical deviations that occur in actual experiments. To account for these deviations, the terms wx_k, wy_k are included in the motion model.As for the observation model, it is described by:

$$z_k = \begin{bmatrix} x_{fs1} \\ y_{fs1} \\ x_{fs2} \\ y_{fs2} \\ x_{fs3} \\ y_{fs3} \\ x_{ms1} \\ y_{ms1} \\ x_{ms2} \\ y_{ms2} \\ x_{ms3} \\ y_{ms3} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + v_k = H * X_k + v_k \tag{10}$$

Where v_k represents the noise in the estimate of target location based on used features. The noise vectors are all assumed to be additive, zero mean uncorrelated Gaussian observation errors with covariance Rx_k, Ry_k, R_k respectively. The covariance matrices are estimated based on linear regression principles, where several experiments are conducted for known target locations and a record of the sum squared error between actual and estimated location is computed for each feature as well as for the simulated and actually executed step. Based on these errors, the covariance matrices are estimated [9].

The equations for location estimation based on an KF scheme are as follows:

$$\text{Prediction} \quad \begin{aligned} X_k^- &= X_{k-1} + \Delta X \\ P_k^- &= A_k P_{k-1} A_k^T \end{aligned} \tag{11}$$

$$\text{Where } X_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}, A_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \begin{aligned} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\ X_k &= X_k^- + K_k (z_k - H_k * X_k^-) \\ P_k &= (1 - K_k) H_k P_k^- \end{aligned}$$

Move-Plan The main function of this module is to issue an appropriate motor command to adjust the robot current position to one that is closer to the visually tracked target. This may be considered as a control problem for a non-linear system. A major advantage of linear systems, from a control point of view, is that they eliminate the need to solve the correspondence problem. Thus, the error in performance (or its integral and derivative like in conventional PID controller) can be used directly to adjust the input signal. However, this advantage is lost in non-linear systems. It becomes then necessary to adopt one of two approaches. Either to solve the correspondence problem. For example, for the present module, this would require finding a mapping between the error in target position in the image space and the actual physical space. This requires a process commonly known as camera calibration. The non-linear map maybe represented using a neural network. However, the complexity of the mapping relation required usually demands a large training data-set which may not be available. The alternative is to seek a suitable mapping that transforms the non-linear system into a linear one. Dealing with linear systems transforms the module design problem into a common PD or PID controller design problem. For the current implementation status of Misk, a calibration based on a feedforward ANN is used to translate the (x, y, scale) into (x,y,z) in a robot centered frame. A move is thus planned as follows. Since the robot moves a single step at a time, the plan is only for a single step in direction of the desired location. The end coordinates of such a step are input to the Kirchoff-based simulator, so as to compute the necessary torques and forces sequence required to steer the robot to the desired location. These torques and forces are the input to the *Move_execute*

Move_execute This module translates forces and torques into required speeds for each of the four motors of the robot. This is done using the thrust-allocation matrix which is derived based on motors positions and available manufacturer information regarding their torque/speed relations. Once the speeds of the motors corresponding to a certain command are determined, the PIC-based pulse width modulation procedure is activated to regulate the input voltage to the motors, so that they acquire the desired speeds. In order to insure real-time operation, the input command sequence to the motors corresponding to steps at different angles are stored in a look up table so that they can be retrieved directly during the robot operation, without having to do the computations all over again.

Progress_assess This module has several functions. First, the robot should know when it has finished the step, so as to reactivate the *Data_collect* module at the correct time and avoid otherwise noisy data that would be collected if the module is activated while the robot is still moving. The Kirchoff-based simulator predicts the time of a step. Using Scilab, it is easy to activate a clock for a period equal to the step duration, after which an appropriate command is used to *Data_collect* module. The second function for this module is to keep track of the robot mission progress over time i.e. it serves as memory to the robot. This is implemented as follows. The robot position to target is stored over 3 consecutive steps. The module compares the distance to target over the different time steps. In case, the distance is increasing, the robot becomes aware that it is diverging from target and seeks an alternative plan.

Obstacle_avoid_em The function of such a module is to detect the presence of obstacle based on the time signal reading of an electromagnetic (EM) sensor. The sensor is inspired from a fish species called electric fish [1]. The sensor consists of two circuits. The transmitter consists of a low frequency crystal attached to a radial type inductor,

which is used to set up an EM field surrounding the robot. The basic receiver circuit of the EM sensor is shown in Fig.5. The main function of such a circuit is to detect EM field surrounding Misk. This circuit is sensitive to low frequency electromagnetic radiation. The EM field is detected using a radial type inductor, used as a probe which responses well to low frequency changing magnetic and electric fields. The sensitivity of this circuit is good. A small load on the electric supply is all that is needed; a 20 watt desk lamp or similar will suffice. The volt – that will arrive to head phones – is input to a PIC microcontroller, which will convert it into a digital signal.
The module has two modes of operation. The calibration mode and the deployment mode. In the calibration mode, the sensor readings corresponding to free and obstacle occupied cells are recorded so that they can be used as reference signals for comparison with future input signals to decide whether they correspond to free or occupied cells. During the deployment phase, an input signal is compared with the stored template to decide whether it corresponds to a free or occupied cell. The outcome of the calibration process is stored in a look up table whose entries take the form of the tuple $(\mathrm{sensor\_reading\_sequence\_template}, \mathrm{cell}_{ijr}, \mathrm{status})$ where each of the six neighboring cells to the robot location cell_{ijr} at different radii r which depend on the range of detection of the EM sensor (this is found empirically during the calibration experiments) and the logical division of the robot environment into different cells. By "status" we mean a code which shows whether the cell is free ("1") or occupied ("0"). To eliminate the need of global coordinate transformations, the camera and EM sensors are aligned and their position is considered to be the origin and robot location. Thus, all calculations are in a robot-centered frame.
Obstacle_avoid_image The function of this model is to detect whether a cell is free or occupied based on an input image. The module is implemented using a feedforward ANN. The input to the ANN are images corresponding to free or occupied cells, while the target is a label (1 for a free a cell and 0 for an occupied cell).
Obstacle_avoid_fuse This module has a similar function to that of the *Multi-fuse_locate* module, since it combines the decisions of the *Obstacle_avoid_EM and Obstacle_avoid_image* modules. The same strategy is thus, employed for the fusion. The *Obstacle_avoid_image* module operates in the image space, while the *Obstacle_avoid_EM* operates in the physical space. To fuse the decisions of the two modules directly, the correspondence between the image and physical space coordinates must be resolved.
Dock Image processing modules are used to identify the validation gate position relative to the robot. When the gate is one step away from the robot, the center position of the validation gate (the point of intersection of its diagonals, since it is a square gate) is thus, input to the precomputed torque controller, as the desired position to track.

Accordingly, the controller outputs the required forces and torques to steer the robot into the gate. The commands are later executed using the *Move_execute* module.

4. Future Extensions

a. Fourier Learning Controller

The reliability of the *Correspond* module which translates an image coordinate into a physical location is questioned. An attractive alternative is to solve the problem totally in the image space. However, a suitable transformation is needed to transform the system dynamics into a linear one in order to be able to implement simple guranteed-to-converge control strategies. A common transformation in case of periodic signals is to use Fourier series [4]. To make use of such an advantageous transformation (from time to frequency domain) in case of non-periodic signals, a learning controller may be used [10]. The learning controller uses the following trick. In order to determine the appropriate control signal, the reference is presented to the control system several times. This makes the "learning" process a periodic process which can be adequately represented using Fourier series. Note that transforming all calculations to be in the image space require a modification to the *Obstacle_avoid_fuse* module. The *Obstacle_avoid_image* module operates in the image space, while the *Obstacle_avoid_EM* operates in the physical space. To fuse the decisions of the two modules directly, the correspondence between the image and physical space coordinates must be resolved. Instead, the problem can be solved indirectly by presenting the Fourier calculated forces/torques sequences as input to the system dynamics Kirchoff simulator to predict the expected physical location. To build the motion library based on Fourier controller a hybrid practical/ simulation based approach is employed. A setup similar to that of the contest site is constructed and divided logically into distinct cells. Next, the camera is moved manually underwater, at each cell images are taken corresponding to robot occupying such cell. This image set is used in a simulation as follows. Given a certain robot location, an input image is presented to the simulator which calculates the target location and plans a step towards it in the image space. The planned step is input as desired location to the Fourier learning controller. At each time step, the Fourier controller suggests a set of forces and torques which are input to the system dynamics simulator based on Kirchoff equations to predict the robot physical position. Based on such position, the appropriate image is retrieved from the image database and presented to the simulator at the next time step.

b. Incremental Learning

Incremental learning is important from two points of view. First, during the course of robot operation there are always more examples showing up. It is thus, desirable to update the robot experience without losing or greatly altering previously stored knowledge. An additional advantage to this incremental learning is to be an automated process that continues without the need of a supervisor. The present paper proposes two approaches to incremental learning.

The first incremental learning scheme is based on a modification of the traditional Self-organizing map (SOM) neural network [7], [11]. It has been pointed out before that SOM can be slightly modified to include supervisory training. We carry out additional modifications so as to introduce a learning paradigm that allows the robot performance to improve (i.e. continues learning) continuously as it proceeds with its job in its working environment. The learning algorithm is comprised of two phases:

a. A supervised initialization phase: The neurons are arranged in a grid (see Fig. 6) as in any traditional SOM. Initially each neuron is associated with a random codeword (weight vector). The available training examples are used to tune the random codewords as follows. Consider, for example, an ANN constructed to classify an input sensory data as corresponding to an occupied or free from obstacles region. The former will be given the code 1 while the latter will be given the code 0. Each one of the available training examples is compared with the random codewords. The codeword of the winning neuron is updated according to the following formula: $w_i = w_i + lr * (w_i - ex\_w)$ (12)
where lr is an arbitrary learning rate (0<lr<1).

In addition to that the codeword of the winner neuron is augmented with the label corresponding to the example as well as the Euclidean distance between the codeword and the example.

The neighboring neurons weights are adjusted according to the following formula:

$$w_j = w_j + lr * mod * (w_j - ex\_w) \tag{13}$$

Where mod is a modulation factor that adjusts the learning rate $mod = \frac{\text{Euclidean distance of the winner neuron}}{\text{Euclidean distance of the neighbor neuron}}$. Moreover, the codewords of the neighbor neurons are augmented with a label =mod in case of class 1 example or 1-mod in case of class 0 example. In addition to that, the Euclidean distance between the current example and the codeword is stored.

b. Incremental non-supervised phase

During this unsupervised phase the robot continues to learn from the examples it encounters in its environment. However, since there are no labels associated with the sensory data, the codewords of a winning neuron and its neighbors are modified according to the following formulas:

For a winner neuron: $w_j = w_j + lr * modwin * (w_j - ex\_w)$ (14)

Where $modwin = \frac{\text{Old Euclidean distance}}{\text{Current Euclidean distance}}$ and the label of the winner neuron is modified proportional to modwin if it was originally >0.5 or to 1-modwin, otherwise. The current Euclidean distance is averaged with the previously stored one and the

average replaces the previously stored value. For a neighbor neuron, the weight is adjusted according to $w_j = w_j + lr * mod neighb * (w_j - ex\_w)$ (15)
Where mod is a modulation factor that adjusts the learning rate $mod neighb = \frac{\text{Euclidean distance of the winner neuron}}{\text{Euclidean distance of the neighbor neuron}}$. While the label is modulated by 1-modneigb, for labels originally <0.5 and proportional to modneigb otherwise.
The second approach to incremental learning is based on a formulation of the learning problem as an interpolation/extrapolation problem, where the available training examples may be considered as control points for a nurb-based [12] curve fitting problem. Thus, any additional point that comes up later is considered as a modification of the closest control point to it. The whole nurb curve is updated based on this control point update. It is well known that nurbs (see Fig.7) have the excellent property of confined local update. This is clear in Fig. 7, which shows that an update of control point only modifies the shape of the curve in the neighborhood of such a point and not globally. During the course of operation, the sensory data will be input with out supervisory labels, to be able to complete the point coordinates with an appropriate label, an input sensory vector is assigned a label similar to the label of the closest (best matching) control point. For example, if the sensory data matches that of the closest control point by 90%, then if the label of the control point is 1, the new readings will be assigned a label 0.9 (0r 0.1 if the label was zero). Once the coordinates of the new point are completed with the appropriate label, the average of the new point and old control point is chosen to replace the old control point and the whole curve is updated.

c. Web-based Teleoperation

Being able to control and teleoperate the robot over the worldwide web is an attractive possibility for several applications, including underwater welding, pipeline inspection and scientific exploration mission. This is possible through connecting the onboard PC to an on-shore PC that is connected to the worldwide web. This robot server runs a Web server program running under a Windows operating system. A user inputs a command to the robot using a facility called forms. The user simply fills the command in a data field in a common hypertext webpage [13]. When the command in such a data field is received by the server, it is interpreted by a special Scilab program and the corresponding control command is send to the robot via a serial link .

5. Discussion and Conclusions

The present paper proposed a simple strategy for building an autonomous underwater robot using limited sensors and computational resources. The choice of a simple robot geometry as well as simple-multifeature based signal processing combined with feedforward networks enables the robot to achieve real-time performance. The relation between different modules is shown in Fig. 8. Possible extensions include applying incremental learning strategies as well as web-based teleoperation offer further opportunities for using the robot in real-life applications.

Acknowledgement

Thanks to Allah almighty for helping us with this project.

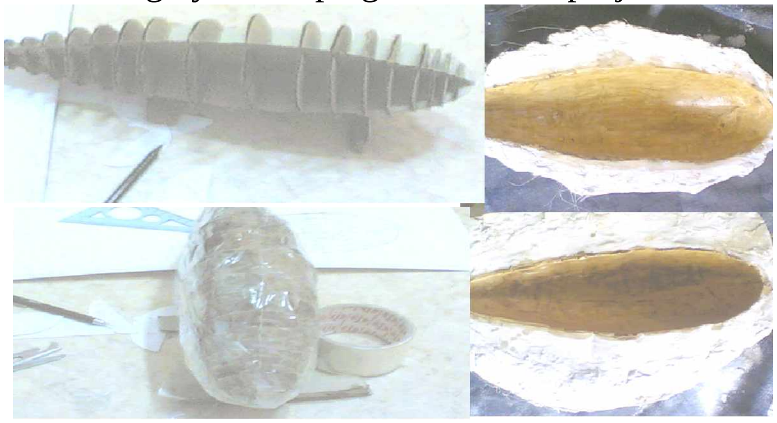


***Figure (1):** Misk construction process combines ship design and small robots design principles.*

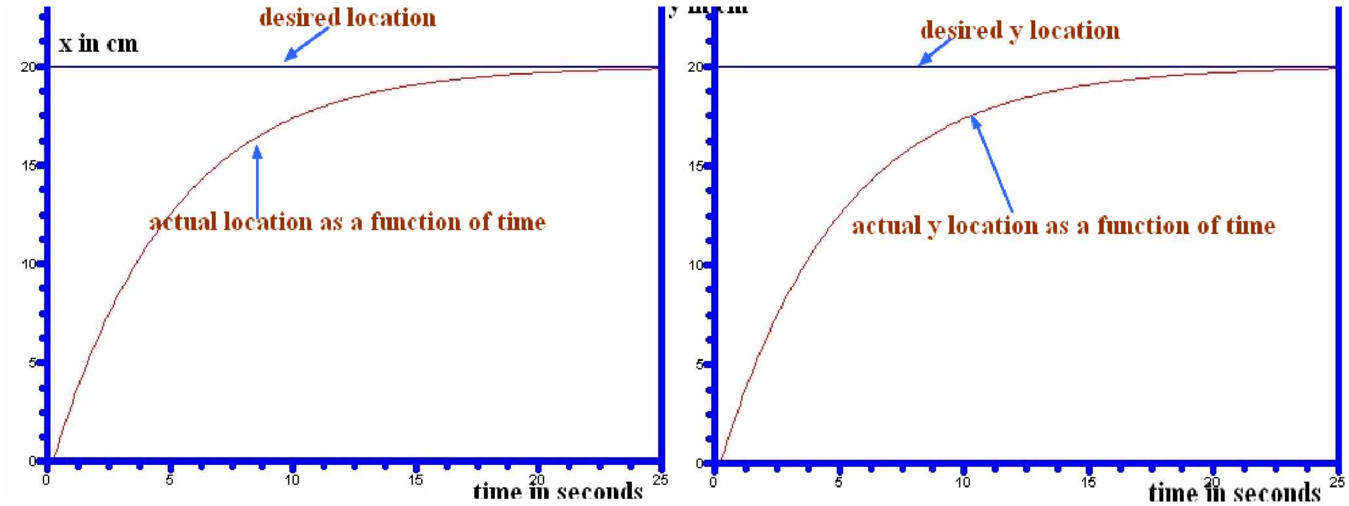


***Figure (2):** The path followed by the robot under control of the precomputed torque controller*

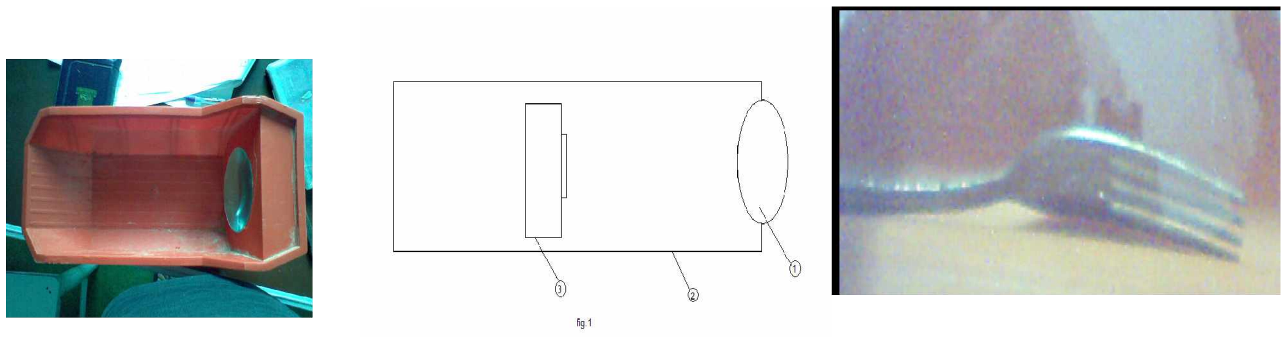
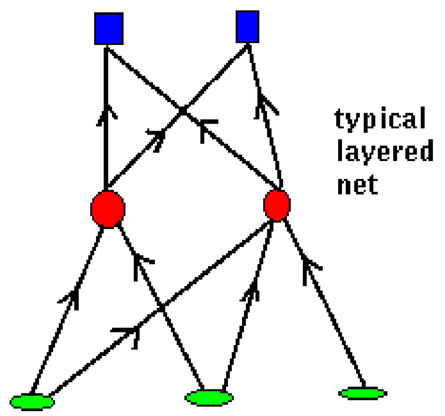
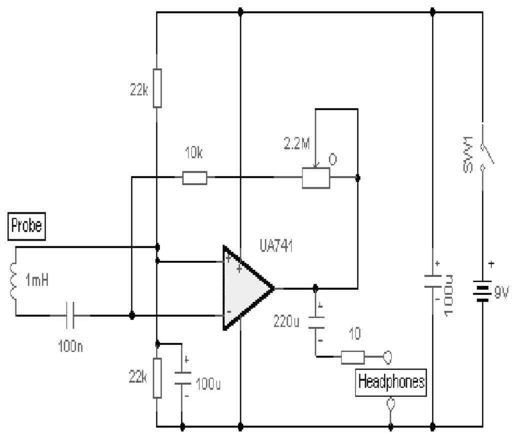


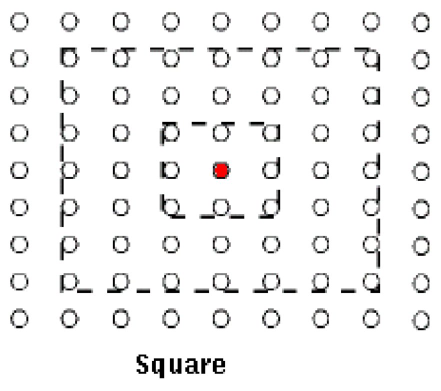
Figure (3): *illustrates the main components of the lens correction system, along with an image taken underwater after lens correction.*



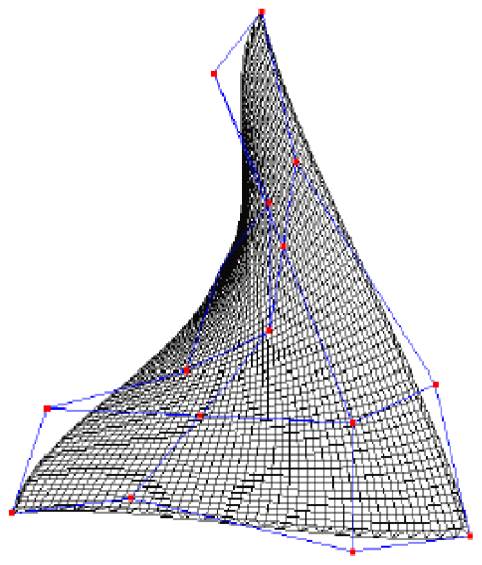
Figure(4): *Feedforward ANN*



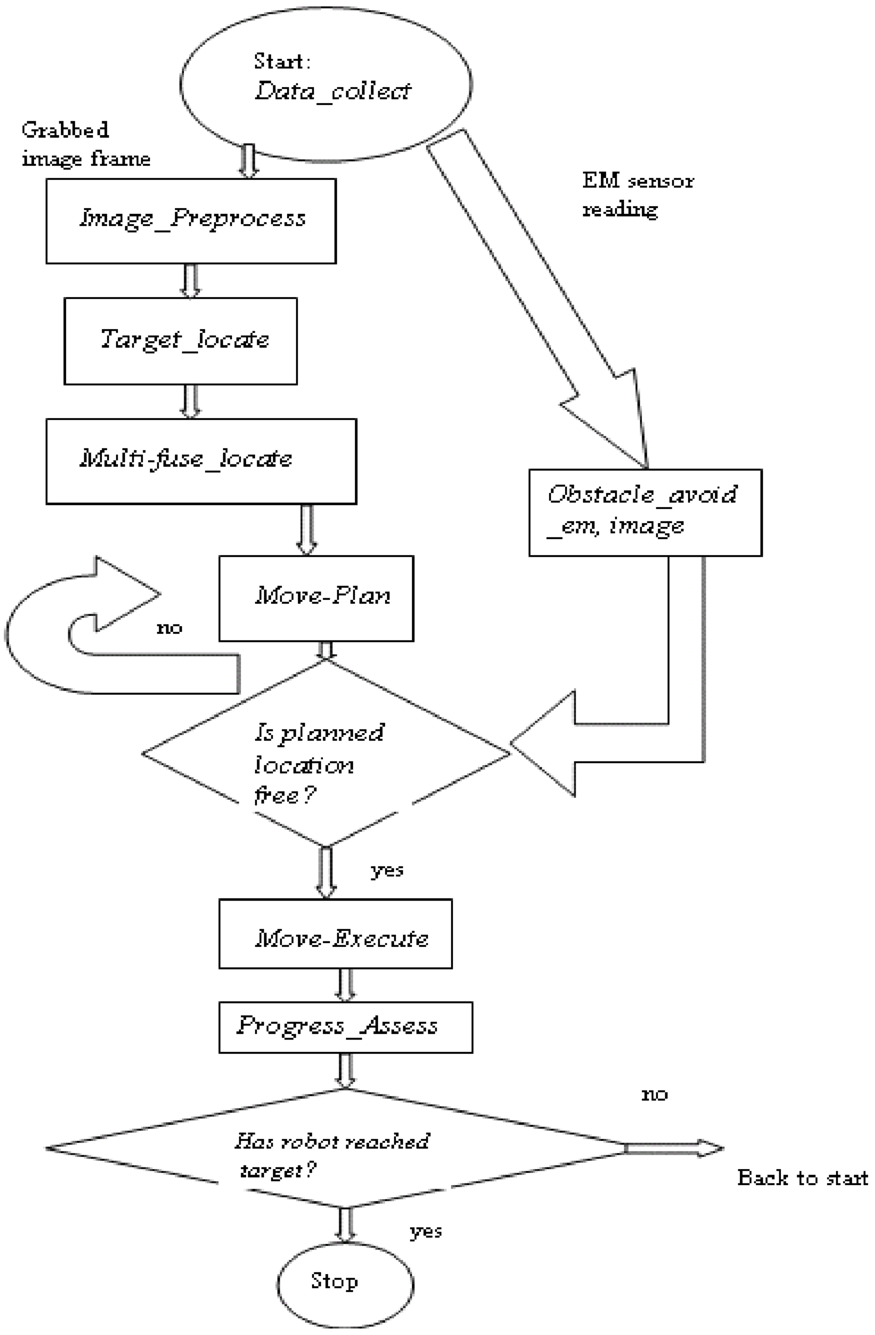
Figure(5): *The receiver circuit for the EM*



Figure(6): *SOM ANN architecture. The neurons are assumed to be arranged on a grid, such that neighboring neurons store similar codewords and are updated together according to the training algorithm.*



Figure(7): *Illustrating the concept of control point modification in NURBs. The training data give the nodes of the shown control polyhedron. It is clear that a change in any of these nodes introduces only local changes in the surface as a whole*


Start:
Data_collect
Grabbed image frame
EM sensor reading
Image_Preprocess
Target_locate
Multi-fuse_locate
Obstacle_avoid _em, image
Move-Plan
no
Is planned location free?
yes
Move-Execute
Progress_Assess
no
Has robot reached target?
Back to start
yes
Stop

References

[1] Malcolm A. MacIver, Ebraheem Fontaine, and Joel W. Burdick, *Designing Future Underwater Vehicles: Principles and Mechanisms of the Weakly Electric Fish*, IEEE Journal of Oceanic Engineering, Vol. 29, no. 3, July 2004

[2] L. Gonzalez, *Design Modelling and Control of an Autonomous Underwater Vehicle,* BSC. Honors Thesis, The University of Western Australia, October 2004.

[3] J. Edward Colgate, K. M. Lynch, *Mechanics and Control of Swimming: A Review*, IEEE Journal of Oceanic Engineering, vol. 29, no. 3, July 2004

[4] H. Attia, A. Saber, M. Haleem, *A New Simple Approach To Biomimetic Automated Underwater Vehicles Design,* International conference of Mechanical Design and Production, MDP9, Cairo, Egypt, 2008.

[5] J.Becker,*PIC16F87X MiniTutorial,EverydayPractical Electronics*,October 1999

[6] S. Bazeille, I. Quidu, L. Jaulin, J. Phillipe, Malkasse, *Automatic Underwater Image Pre-Processing,* CMM'06 - Caracterisation Du Milieu Marin, 2006.

[7] S. Haykin, *Neural Networks,* Prentice Hall Inc., N.J, U.S.A, 1999.

[8] H. Durrant-Whyte, T. Bailey, Simultaneous *Localisation and Mapping(SLAM)Part I The Essential Algorithms*, IEEE Robotics and Automation Magazine, June 2006.

[9] M. Nguyen, *Design of an Active Acoustic Sensor System for an Autonomous Underwater Vehicle*, BSC. Honors Thesis, Mobile Robotics Laboratory, The University of Western, November 2004.

[10] Xiaoqi Tang, Lilong Cai, and Weiqing Huang, *A Learning Controller for Robot Manipulators Using Fourier Series*, IEEE Transactions on Robotics and Automation, Vol. 16, no. 1, February 2000

[11] O. Abdel Aleem, *Mine Detection by Robots Using Code Division Multiple Access (CDMA) Signals*, Master thesis, Alexandria University, July 2002

[12] F. Andersson, *Bezier and B-spline Technology*, Honors Thesis, 2003 http://citeseer.ist.psu.edu/612466.html

[13] K. Taylor, J. Trevelyan, *Australia's Telerobot on the Web*, International Symposium on industrial robots, Singapore, 1995