Robot Visual Homing using Conjugate Gradient Temporal Difference Learning, Radial Basis Features and A Whole Image Measure

Abdulrahman Altahhan, *Member, IEEE*

***Abstract*—This work describes a robot visual homing model that employs, for the first time, the conjugate gradient Temporal Difference (TD-conj) method. TD-conj was proved to be equivalent to a gradient TD method with a variable λ, denoted as (TD($\lambda_t^{(conj)}$)), when both are used with function approximation techniques. This fact is employed in the model to improve its performance. Based on visual input that is passed through radial basis layer, the model takes advantage of the model-free interactive-learning capability of reinforcement learning (RL) by using a whole image measure to recognize the goal, without the aid of special landmarks. Therefore, unlike other models, this model refrains from artificially manipulating the environment or assuming a priori knowledge about it, two typical constraints that widely restrict the applicability of existing models in realistic scenarios. An on-policy on-line control method was used to train a set of neural networks. With the aid of variable eligibility traces, these networks approximates the agent's action-value function allowing it to take optimal actions to reach its home. The effectiveness of the model was experimentally verified where an agent equipped with it achieved efficacy in finding a goal location with no a priori knowledge of the environment.**

I. Introduction

Homing is the ability to find its home location after an animal has forged for food. This ability is essential for animals survival and sociability. Visual homing is the act of finding a home location using vision. In general, it is performed by comparing the image currently viewed with stored 'snapshot' images of the home (normally taken while an animal or robot is heading off its home location). A movement decision is then taken accordingly [1].

A skill which plays a critical role in achieving robot autonomy is the ability to learn to operate in a priori unknown environments [1]-[3]. In particular learning to navigate and return to home in a priori unknown environments is essential. If this process is automated and made easy to apply, it can be used as a first-step to learn to reach any location in any environment, and hence navigation in its broadest convention is accessible. Numerous models have been proposed in the literature to allow mobile robots to navigate and return home for a wide range of environments. Some focus on learning [1], [4]-[7], whilst others focus on the successful application of a model or algorithm for a specific environment and ignore the learning problem [8]-[10].

Some robotics approaches borrow conceptual mechanisms from animal homing and navigation strategies described in neuroscience or cognition literature [11], [12]. Algorithms based on the snapshot model [7], [12], [13] propose various strategies for finding features within images and establishing correspondence between them in order to determine home direction. Block matching, for example, takes a block of pixels from one image and searches for the best matching block in another image within a fixed search radius [14].

The preponderance of the homing models proposed in the literature have the limitations of either depending on landmarks [15]-[17] which makes them environment-specific, or requiring a pre-processing stage (such as obtaining derivatives of some kind for the studied environment) in order to learn (or perform) the task [6], [18], which made them not immediately practical. These assumptions restrict the employability of such models in a useful and practical way. In fact, new findings in cognition suggest that humans are able to do homing in the absence of feature-based landmark information [19]. This means that the former limitation is unnecessarily imposed on existing models and biology suggests that it is possible to eliminate it.

To escape the first limitation a whole image measure will be used in our model. Object recognition is not necessary to recognize a location, or to perform navigation, homing or localization tasks. It is the location itself that should be used in the recognition. Therefore, we aim at avoiding specific object recognition by using a whole image measure technique in our model. Whole image measures of current and stored snapshots have been used to do robot localization [20]. Nevertheless, visual homing based on a whole image measure and RL to complete the task and augment the reward is a novel aspect of this work. Whole image measures are also discussed extensively in the information retrieval field [21]. In particular, measures that use bins of histograms are of interest due to their effectiveness and simplicity. Jeffery Divergence Measure, for example, uses the number of elements that belong to bins in the histograms of the two compared images in order to obtain a scale that expresses how different those images are to each other. JDM has been successfully used to perform robot localization [20].

To escape the second limitation a general learning process

A. Altahhan is with the Faculty of Informatics and Communications Engineering, Yarmouk Private University, Dar'a, Syria. www.his.sunderland.ac.uk (phone: +963158691000 (1071); e-mail: abed.tahhan@gmail.com, abdulrahman-altahhan@ypu.edu.sy).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

should be employed that is able to capture the specificity of any environment without gearing it up into the model architecture. Reinforcement learning (RL) provides such a capability. RL has been used previously in robot's navigation and control. Several of the models using it are inspired by biological findings, e.g. [16], [22]. Although successful, some of those models lack the generality and/or practicality, and some are restricted to their environment. The model proposed by [16], [17], [23] for example depends on object recognition of a landmark in the environment to achieve the task. Therefore, we aim at exploiting the capability of RL as much as possible by general model design as well as using whole image measure. Reinforcement learning advocates a general learning approach that avoids human intervention of supervised learning and, unlike unsupervised learning, has a specific problem-related target that should be met. What is more, since RL deals with reward and punishment it has strong ties to the biology, making it suitable to our homing problem. While environment-dynamics or map-building may be necessary for more complex or interactive forms of navigation or localization, visual homing based on model-free learning can offer an adaptive form of local homing. Furthermore, although the immediate execution of model-based navigation can be successful [24], [25], RL techniques have the advantage of being model-free i.e. no knowledge needed about the environment prior to operating the robot in order to achieve the task. The agent learns the task by learning the best policy that allows it to collect the largest sum of rewards from its environment according to the environment dynamics.

II. The visual homing Sarsa-conj(0) model

For visual homing we assume that the image at each time step represents the current state, and the state space S is the set of all images, or views, that can be possibly taken for any location (with specific orientation) in the environment.

This complex state space has two problems. First; each state is of high dimensionality, i.e. each state is represented by a large number of pixels. Second; the state space is huge, and a policy cannot be learned directly for each state. Instead, a feature representation of the states is used to reduce the high-dimensionality of the state space and to gain the advantages of coding that allows a parameterized representation to be used for the value function [26]. In turn, parameterization permits learning a general value function representation that can easily accommodate for new unvisited states by generalization. Eventually, this helps to solve the second problem of having to deal with huge state space.

The feature representation can reduce the high-dimensionality problem simply by reducing the number of components needed to represent the views. Hence, reducing dimensionality is normally carried out on the cost of less distinctiveness for states that belong to a huge space. Therefore, the features representation of the state space, when successful, strikes a good balance between distinctiveness of states and reducing dimensionality. This assumption is of importance towards the realization of any RL model with a high-dimensional states problem.

A. High-dimensionality problem and histogram of an image

One representation that maintains an acceptable level of distinctiveness and reduces the high-dimensionality of images is the histogram of the image. A histogram of image *im* consisting of p pixels is a vector of components, each of which contains the number of pixels that belong to a certain range of values. The significance of histograms is that they map a large two-dimensional matrix to a lower and smaller one-dimensional vector. This effectively encodes the input state space to a coarser feature space.

Therefore, if the RGB (Red, Green, and Blue) representation of colours is used for an image *im* the histogram of each colour channel is the vector of components, each of which is the number of pixels that lie in the component's interval. The interval each component represents is called the bin, and according to a pre-specified bin size b of the range of the pixel values, a pre-specified number of bins B will be obtained.

The histogram representation does not preserve the distinctiveness of the image, i.e. two different images can have the same histogram, especially when low granularity bin intervals are chosen. This is because a histogram maps the two dimensional space of the image into a one dimensional vector representation, losing the order of the pixels. Nevertheless, histograms have been found to be widely acceptable and useful in the image processing and image retrieval communities [21].

Another issue remaining is the intractability problem. A generalization technique is needed in order to accommodate the intractability of the state space. More precisely, generalization is needed in order to approximate the value for a state that has never been visited before, through previous visits to similar states. A natural way to do so is to use function approximation techniques such as artificial neural networks.

B. Defining the goal (home) location

The feature representation approach does not give a direct indication of the distance to the goal location. Although the assumption that the goal location is always in the robot's field of view will not be made, however by comparing the current view with the goal view(s) the properties of distinctiveness, distance and orientation can be embodied in the RL model to an extent.

Since the home location can be approached from different directions, the way it is represented should accommodate for those directions. Therefore, a home (or a goal) location is defined by m snapshots called the stored views. A few snapshots (normally $m \geq 3$) of the home location are taken at the start of the learning stage, each from the same fixed distance but from a different angle. These snapshots define the home location and are the only requirement for the

model to allow the agent to learn to reach its home location starting from any arbitrary position in the environment (including those from which it cannot see the home, i.e. the agent should be able to reach a hidden goal location).

C. Features vectors and radial basis representation

The main aim of establishing the feature vector representation, from views comparison perspective, is to maximize the variation of the feature vectors for any two views, so that the changes to these features are maximized with respect to view changes. Consequently, the greater the difference between the views, the greater the difference between the corresponding vectors of features should be.

To maximize the distinctiveness of each feature from others the aim should be to maximize the sum of the differences from their means. In other words the features should be normally distributed. This should be done proportional to the variance, and preferably be scaled to the unit interval [0, 1]. In order to be able to reach the goal from any location in the environment, the mean of each bin (in the current view histogram) will be regarded to be the value of this same bin for the home location stored view. Then by assuming a normal distribution of those bins, a radial basis representation is obtained.

$$\phi_i(s_t(c,j)) = \exp\left(-\frac{(h_i(s_t(c)) - h_i(v(c,j)))^2}{2H^2\hat{\sigma}^2}\right) \quad (1)$$

Index t stands for the time step, j for the j^{th} stored view, and c is the index of the channel, where the RGB representation of images is used. Accordingly, $v(c,j)$ is the image of channel c of the j^{th} stored view, $h_i(v(c,j))$ is the histogram's bin of image $v(c,j)$, and $h_i(s_t(c))$ is histogram's bin i of channel c of the current (t) view. The number of bins will have an effect on the structure and richness of this representation and hence on the model. H is a normalization factor where $H = \sum_i^n h_i(v(c,j))$. It should be noted that this feature representation has the advantage of being in the interval [0, 1], which will be beneficial, as discussed in the next section.

The feature vector of the current view (state) is a union of all of the features for each channel and each stored view as follows:

$$\Phi(s_t) = \bigoplus_{j=1}^{m}\bigoplus_{c=1}^{3}\bigoplus_{i=1}^{B}\phi_i(s_t(c,j)) = (\phi_1,\ldots,\phi_i,\ldots,\phi_n) \quad (2)$$

where m is the number of stored views for the goal location, 3 channels are used, and B is the number of bins to be considered. Different bin sizes give different dimensions, which in turn give different numbers of parameters $\vec{\theta}$ that will be used to approximate the value function.

D. Advantages and limitations of the features representation

Using this radial basis representation (normalized or not) means that the model takes into account that the agent might 'want' to reach the home location form any start location; this is robust when the complexity of the vision medium is taken into account. The advantage of normalization will become apparent when using it in combination with the linear network, as all multiplications are guaranteed to be the unit weight interval. The other advantage is that this scaling allows the use of those features as a similarity measure. This will be discussed further in the next section.

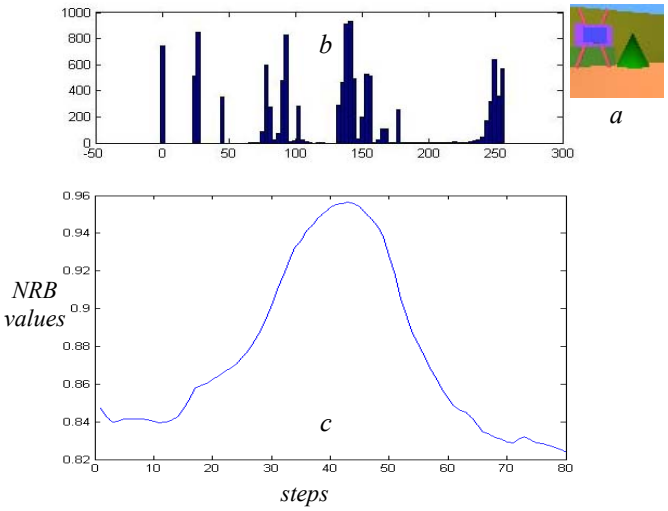

Fig. 1. (a): Current view of the agent's camera in a simulated environment. (b): its associated histograms (c): the plotted values of the normalized radial basis similarity measure or NRB of a sample π rotation.

The limitation of this representation is that the agent needs to be completely retrained when the goal location is changed. This retraining can be reduced by using an off-policy learning method. This issue, however, is outside the scope of this work and hence will not be covered. One could also store the once trained goals and use it later. It is still worth mentioning that in a higher level of control, each different goal location could be assigned a different set of networks and the agent can reach this home from any location in the environment according to some verbal or desired input instructing it to go to a particular location. The relatively small amount of information (in the form of stored views) required by this novel model to enable the agent to reach different goal locations is especially attractive.

E. NRB similarity measure and the termination condition

To measure the similarity between two images, the sum of all the Normalized Radial Basis (NRB) features defined above can be taken and then be divided by the feature dimension. The resultant quantity is scaled to 1 and it expresses the overall belief that the two images are identical:

$$NRB(s_t) = \frac{\sum_{i=1}^{n}\bar{\phi}_i(s_t)}{n} \quad (3)$$

For simplicity the notation $NRB(s_t)$ and NRB_t will be used interchangeably. Other measures can be used [21]. In previous work the Jeffery divergence measure was used. However the above simpler measure was adopted because it is computationally more efficient for the proposed model since it only requires an extra sum and a division operations. While JDM has its own logarithmic calculation which cost additional computations. Another benefit of NRB is that it is scaled to 1 – a uniform measure can be interpreted more intuitively.

Fig. 1 shows a sample of a view representation of a home location taken in a simulated environment, it associated histogram and the NRB measures for this view and different orientation of the agent while it is rotating in the same location. The bell shape of this measure in relation to changes of orientation is evident. It tells us that we can expect it to normally increase when the orientation of the current view allies more with the reference view of the home location.

F. The termination condition

An episode is the collective steps of starting from any location and navigating in the environment until its end is marked by reaching the home location. The agent is assumed to finish an episode and be in the home location (final state) if the agent's similarity measure indicates with high certainty $\geq \psi_{upper}$ that its current view is similar to one of the stored views. This specifies the episode termination condition of the model: *If* $NRB(s_t) \geq \psi_{upper} \Rightarrow$ *Terminate Episode.*

Similarly, the agent is assumed to be in the see the home location: *If* $NRB(s_t) \geq \psi_{lower}$ where $\psi_{upper} \geq \psi_{lower}$. This situation when the agent is in the neighborhood of the home location with the desired orientation is called *goal-on-sight* and the interval $[\psi_{lower}, \psi_{upper}]$ is called the *goal-on-sight confidence interval*.

G. The action space

In order to avoid the complexity of dealing with a set of actions each with infinite speed values (which in effect turns into infinite number of actions), the two differential wheel speeds of the agent were assumed to be set to certain values, so that a set of three actions with fixed values is obtained. The set of actions is A = [Left_Forward, Right_Forward, Go_Forward]..Although the acceleration of continuous action space cannot be obtained using this limited set of actions. Nevertheless, by using actions with a small differential speed (i.e. small thrust rotation angle) the model can still get the effect of continuous rotation by repeating the same action as needed, this is done at the cost of more action steps obviously.

One could use a different set of actions than the limited one used here, to enhance the actions performance. For example, another three actions [*Left_Forward, Right_Forward, Go_Forward*] with double the speed could be added, which might give better performance, although more training would be a normal requirement in this case. One can also add a layer to generalize towards other actions by enforcing a Gaussian activation around the selected action and fade it for other actions, as in [27]. In this work, however, the action set was kept to minimum to concentrate on the effect of other components of the model.

H. The reward function

The reward function r depends on which similarity or dissimilarity function was used, and it consists of three parts:

$$r_{NRB} = \text{cost} + \Delta NRB_{t-1} + NRB_t \tag{3}$$

- The main part is the *cost,* which is set to -1 for each step taken by the robot without reaching the home location (reaching a termination state). The other two parts are to augment the reward signal to provide better performance. They are:

- Approaching the goal reward. This reward is the maximum increase in similarity between the current step and the previous step. This signal is called the *differential similarity signal* and it is defined as*:*

$$\Delta NRB_{t-1} = (NRB_t - NRB_{t-1}) \tag{4}$$

- The Position signal, which is simply expressed by the current similarity NRB_t. Thus, as the current location differs less from the home location, this reward will increase. Hence, the reward can be rewritten in the following form:

$$r_{NRB} = \text{cost} + 2NRB_t - NRB_{t-1} \tag{5}$$

The two reward's components above will be considered only if the similarity of *t* and *t-1* steps are both beyond the threshold ψ_{lower}, to ensure that goal-on-sight is satisfied in both steps. This threshold is empirically determined, and is introduced merely to enhance the performance.

I. The eligibility traces and update rule for TD(λ)

An eligibility trace constitutes a mechanism for temporal credit assignment. It marks the memory parameters associated with the action as being eligible for undergoing learning changes [28]. For the visual homing application, the eligibility trace for the current action a is constructed form the feature vectors encountered so far. More specifically, it is the discounted sum of the feature vectors of the images that the robot has seen in each time the same action a had been taken. The eligibility trace for other actions which have not been taken while in the current state is simply its previous trace but discounted, i.e. those actions are now less accredited as it is demonstrated in the following equation.

$$\vec{\mathbf{e}}_t(a) \leftarrow \begin{cases} \gamma\lambda\vec{\mathbf{e}}_{t-1}(a) + \vec{\boldsymbol{\varphi}}(s_t) & \text{if } a = a_t \\ \gamma\lambda\vec{\mathbf{e}}_{t-1}(a) & \text{otherwise} \end{cases} \tag{6}$$

λ is the discount rate for eligibility traces $\vec{\mathbf{e}}_t$ and γ is the rewards discount rate . The eligibility traces components do not comply with the unit interval i.e. each component can be more than 1. The reward function also does not comply with the unit interval. The update rule that uses the eligibility trace and the episodically changed learning rate α_{ep} is as:

$$\vec{\boldsymbol{\theta}}(a_t) \leftarrow \vec{\boldsymbol{\theta}}(a_t) + \alpha_{ep} \cdot \vec{\mathbf{e}}_t(a_t) \cdot \delta_t \tag{7}$$

J. Variable eligibility traces and update rule for TD-conj(0)

As it was shown in [29] the conjugate gradient TD-conj(0) method is translated through an equivalency theorem into a TD(λ) method with variable λ denoted as TD($\lambda_t^{(conj)}$) with the condition that $0 \le \lambda_t^{(conj)} \le 1$. Therefore, to employ conjugate gradient TD, the same formula (above) can be applied to obtain the eligibility traces for TD-conj. The only difference is that λ is varying according to one of the following possible forms:

$${}^1\lambda_t^{(conj)} = \frac{\left(\delta_t\vec{\phi}_t - \delta_{t-1}\vec{\phi}_{t-1}\right)^T\vec{\phi}_t}{\gamma\left(\delta_t\vec{\phi}_t - \delta_{t-1}\vec{\phi}_{t-1}\right)^T\vec{e}_{t-1}^{(conj)}},\quad {}^2\lambda_t^{(conj)} = \frac{\delta_t\left\|\vec{\phi}_t\right\|^2}{\gamma\delta_{t-1}\left\|\vec{\phi}_{t-1}\right\|^2}, \tag{8}$$

$$\text{or}\quad {}^3\lambda_t^{(conj)} = \frac{\left(\delta_t\vec{\phi}_t - \delta_{t-1}\vec{\phi}_{t-1}\right)^T\vec{\phi}_t}{\gamma\delta_{t-1}\left\|\vec{\phi}_{t-1}\right\|^2}$$

TD-conj(0) (and any algorithm that depends on it such as Sarsa-conj(0) [29]) is a family of algorithms, not because its λ is changed automatically from one step to the other, but because λ can be varied using different types of formulae. Some of those formulae are outlined in (8).

The eligibility traces can be written as:

$$\vec{\mathbf{e}}_t^{(conj)}(a) \leftarrow \begin{cases} \gamma\lambda_t^{(conj)}\vec{\mathbf{e}}_{t-1}^{(conj)}(a) + \vec{\boldsymbol{\varphi}}(s_t) \; if \;\; a = a_t \\ \gamma\lambda_t^{(conj)}\vec{\mathbf{e}}_{t-1}^{(conj)}(a) \qquad otherwise \end{cases} \tag{9}$$

For episodic tasks γ can be set to 1 (absence). The reason is –that there is no need to discount the sum of rewards $R_t = \sum_{k=0}^{T}\gamma^k r_{t+k+1}$ because the sum is bounded, since the task ends after reaching final state at time T. Therefore, in this case there is no concern that R_t goes to infinity as is the case for tasks with infinite horizon $R_t = \sum_{k=0}^{\infty}\gamma^k r_{t+k+1}$. Finally the update rule is identical to (7), where the conjugate eligibility trace is used instead of the fixed λ eligibility trace:

$$\vec{\boldsymbol{\theta}}(a_t) \leftarrow \vec{\boldsymbol{\theta}}(a_t) + \alpha_{ep} \cdot \vec{\mathbf{e}}_t^{(conj)}(a_t) \cdot \delta_t \tag{10}$$

K. The policy used to generate actions

A combination of ε-greedy policy and Gibbs soft-max [28] policy is used to pick up an action and to strike a balance between exploration and exploitation.

$$\pi_{\varepsilon+Gibbs}(a_i, \vec{\varphi}(s_t)) = Gibbs(a_i, \vec{\varphi}(s_t)) + \Pr(a_i, \vec{\varphi}(s_t)) \tag{11}$$

Using ε-greedy probability allows exploration to be increased as needed by initially setting ε to a high value then decreasing it through episodes.

$$\Pr(a, \vec{\varphi}(s_t)) = \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|A|} & if \;\; a = \underset{i}{\operatorname{argmax}}\left[\vec{\varphi}(s_t)^T \cdot \vec{\theta}(a_{(i)})\right] \\ \dfrac{\varepsilon}{|A|} & otherwise \end{cases} \tag{12}$$

The Gibbs soft-max probability,

$$Gibbs(a_{(i)}, \vec{\varphi}(s_t)) = \frac{\exp\left[\vec{\varphi}(s_t)^T \cdot \vec{\theta}(a_{(i)})\right]}{\sum_{j=1}^{|A|}\exp\left[\vec{\varphi}(s_t)^T \cdot \vec{\theta}(a_{(j)})\right]} \tag{13}$$

supports the chances of picking the action with the highest value when the differences between the values of it and the remaining actions are large, i.e. it helped in increasing the chances of picking the action with the highest action-value function when the robot is sure that this value is the right one.

L. The learning method and actor-only structure

The last model component to be discussed is the learning algorithm. The basis of the model learning algorithm is the Sarsa(λ) control algorithm with linear function approximation [28]. However, this algorithm was adapted to use the TD-conj(0) instead of the TD(λ) update rules. Hence, it was denoted as Sarsa-conj(0). From a theoretical point of view, TD-conj(0) [29] –and any algorithm depending on its update such as Sarsa-conj(0)– uses conjugate gradient direction in conjunction with TD(0) update. While, from an algorithm implementation point of view, according to the equivalency theorem [29], TD-conj(0) and Sarsa-conj(0) have the same skeleton of TD(λ) and Sarsa(λ) [28] with the difference that TD-conj(0) and Sarsa-conj(0) use the variable eligibility traces $\lambda_t^{(conj)}$ [29]. The benefit of using TD-conj(0) update is to optimize the learning process (in terms of speed and performance) by optimizing the depth of the credit assignment process according to the conjugate directions, purely through automatically varying λ in each time step instead of assigning a fixed value to λ manually for the duration of the learning process.

Sarsa is an on-policy bootstrapping algorithm that has the properties of (I) being suitable for control, (II) it provides function approximation capabilities to deal with huge state space and (III) it learns on-line. These three properties are considered ideal for the visual robot homing (VRH)

problem. The ultimate goal for VRH is to control the robot to achieve the homing task, in VRH the state space is huge because of the visual input, and on-line learning was chosen because of its higher practicality and usability in real world situations than off-line learning.

Initializa tion

initialize b, m, *and final_epis ode*

$n \leftarrow\approx 3 \times \frac{256}{b} \times m$

$\vec{\theta}_0(a_i) \leftarrow \vec{1} \qquad i = 1 : |A|$

$a_0 \leftarrow 2$

Repeat for each episode

$\vec{\mathbf{e}}_0(a_i) \leftarrow \vec{\mathbf{0}} \quad i = 1 : |A|$

$s_0 \leftarrow$ Initial robot view , $t \leftarrow 1$

Generate a_0 using sampling of probabilit y $\pi(\vec{\varphi}(s_0), a)$

Repeat (for each step of episode)

Take action a_t , Observe r_{t+1}, $\vec{\varphi}(s_{t+1})$,

Generate a_{t+1} using sampling of probabilit y $\pi(\vec{\varphi}(s_{t+1}), a)$.

$\delta_t \leftarrow \left[r_{t+1} + \gamma \vec{\varphi}(s_{t+1})^T \cdot \vec{\theta}(a_{t+1}) - \vec{\varphi}(s_t)^T \cdot \vec{\theta}(a_t)\right]$

$^1\lambda_t^{(conj)} \leftarrow \frac{(\delta_t \vec{\varphi}_t - \delta_{t-1}\vec{\varphi}_{t-1})^T \vec{\varphi}_t}{\gamma(\delta_t \vec{\varphi}_t - \delta_{t-1}\vec{\varphi}_{t-1})^T \vec{\mathbf{e}}_{t-1}}$

$\vec{\mathbf{e}}_t^{(conj)}(a) \leftarrow \begin{cases} \gamma\lambda_t^{(conj)} \vec{\mathbf{e}}_{t-1}^{(conj)}(a) + \vec{\varphi}(s_t) & if \ a = a \\ \gamma\lambda_t^{(conj)} \vec{\mathbf{e}}_{t-1}^{(conj)}(a) & otherwise \end{cases}$

$\vec{\theta}(a_t) \leftarrow \vec{\theta}(a_t) + \alpha_{ep} \cdot \vec{\mathbf{e}}_t^{(conj)}(a_t) \cdot \delta_t$

$\vec{\varphi}(s_t) \leftarrow \vec{\varphi}(s_{t+1}), \vec{\varphi}(s_{t-1}) \leftarrow \vec{\varphi}(s_t)$

$\delta_{t-1} \leftarrow \delta_t, \quad a_t \leftarrow a_{t+1}$

until $\frac{1}{n}\sum_{i=1}^{n} \phi_i(s_t) < \psi_{upper}$

until *episode* $==$ *final_epis ode*

Fig. 2. Conjugate gradient Sarsa control, with RBF features extraction, linear action-value function approximation and dynamic-Policy Improvement. The approximate Q is implicitly a function of $\vec{\theta}$. $\lambda_t^{(conj)}$ can be assigned to any of the three forms calculated in the preceding step.

The action-value function was used to express the policy, i.e. this model uses a critic to induce the policy. Actor-critic algorithms could be used. Using such a structure has its own simplicity advantage but has the disadvantage of high variance in the TD error [30]. This can cause both high fluctuation in the values of the TD error and divergence. This limitation was addressed in this model by carefully designing a suitable scheme to balance exploration and exploitation according to a combination of Gibbs distribution and ε-greedy policy. The Gibbs exponential distribution has some important properties which helped in realizing the convergence. According to [31] it helps the TD error to lie in accordance with the natural gradient.

In that sense this model is rather a hybrid model. It is like any action-value model uses the action-value function to induce the policy but not in a fixed way. It also changes the policy preferences (in each step or episode) towards a more greedy policy like any actor-critic models. So it combines and mixes between action-value and actor-critic models. It is felt that this hybrid structure has its own advantages and disadvantages, the convergence properties of such algorithms need to be studied further in the future.

TD-conj(0) learns on-line through interaction with software modules that feed it with the robot visual sensors (whether from simulation or from real robot). The algorithm coded as a controller returns the chosen action to be taken by the robot, and updates its policy through updating its set of parameters used to approximate the action-value function Q. Three linear networks are used to approximate the action-value function for the three actions. $\vec{\theta}(a_{(i)}) = (\theta_1^{a(i)}, \ldots, \theta_i^{a(i)}, \ldots, \theta_n^{a(i)}) \qquad i = 1,..|A|$.

The current image was passed through an RBF layer, which provides the feature vector $\vec{\varphi}(s_t) = (\phi_1, \ldots, \phi_i, \ldots, \phi_n)$. The robot was left to run through several episodes. After each episode the learning rate was decreased, and the policy was improved further through general policy improvement theorem (GPI). The overall algorithm is stated in Fig. 2.

M. The neural network architecture

From a neural network point of view, each action network is a radial basis network that uses RBF feature input and a linear layer output. The RBF networks work collectively together with an output layer that contains the Gibbs and greedy probabilities neuron. This architecture is a probabilistic architecture that calculates the probability of picking up a certain action conditional to the given state.

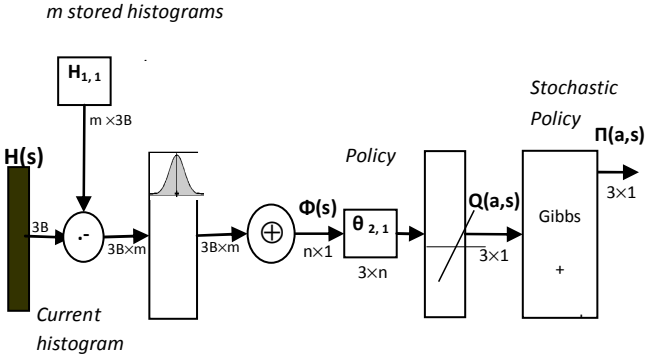

Fig. 3. Radial Basis Function Neural Network with stochastic policy estimation capability.

The neural network structure used in this model is shown in Fig.3. Number 3 in the components to the left half of the figure is coming from and RGB channels representation, while number 3 in the right half is coming from having 3 actions. The unification operation is a simple transfer from matrix to vector by reordering the matrix elements in a queue. The stochastic policy is given to a sampler that uses a pool of 1000 samples that are divided between the actions according to their probabilities. A uniform distribution is then used to choose the action. In effect the action is chosen according to the given action probabilities.

III. EXPERIMENTAL RESULTS

The model was applied using a simulated Khepera [32] robot in Webots™ [33] simulation software. The Khepera is a miniature real robot, 70 mm diameter and 30 mm height, and is provided with 8 infra-red sensors for reactive behaviour, as well as a colour camera extension.

A 1.15x1 m2 simulated environment has been used as a test bed for our model. The task is to learn to navigate from any location in the environment to a home location (without using any specific object or landmark). For training, the robot always starts from the same location, where it cannot see the target location, and the end state is the target location. While, after learning the robot can be put in any part of the environment and can still find the home.

Fig. 4. shows the environment used. A cone, ball and TV are included to add more texture to the goal location, i.e. to enrich it and make it different from the other environment locations. We re-emphasize that no object recognition technique was used, only the whole image measures were used (whether JDM or NRB). This allows the model to be applied to any environment with no constraints and with minimal prior information about the home. The controller is written as a combination of C++ code and Matlab Engine code.

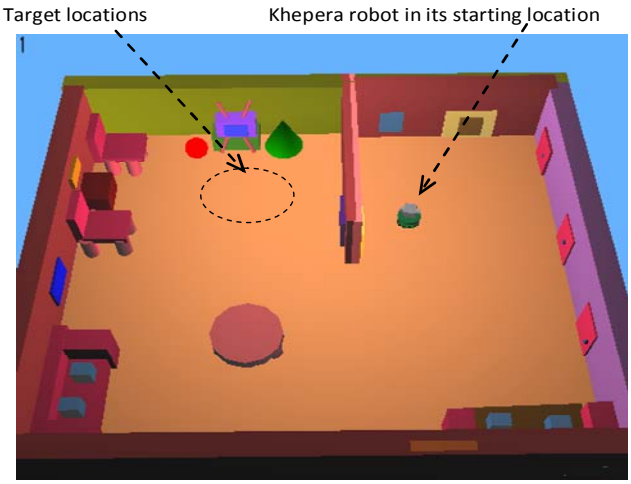

Fig. 4. A snapshot of the realistic simulated environment. The home location is in front of the TV, while the robot starts in a place where it cannot see the home.

The robot starts by taking (m=3) snapshots for the goal location. It then goes through a specific number (EP) of episodes that are collectively called a run-set or simply *a run*. In each *episode* the robot starts from a specific location and is left to navigate until it reaches the home location. The robot starts with a random policy, and should finish a run set with an optimized learned policy.

A. Practical settings of the model parameters

Table 1 gives a detailed description of the different constants and parameters used in Sarsa-conj(0) algorithm and their values/initial values and updates. Each run lasts for 500 episodes (EP=500), and the findings averaged over 10 runs to insure validity of the results. The feature space parameters were chosen to be b=3, m=3. Hence, $n = 3\times(round(256/3)+1)\times 3 = 774$. This middle value for b, which gives a medium feature size (and hence medium learning parameters dimension), together with the relatively small number of stored views (***m=3***), were chosen mainly to demonstrate and compare the different algorithms on average model settings, different setting could have been chosen.

TABLE I
THE MODEL DIFFERENT PARAMETERS, THEIR VALUES AND THEIR DESCRIPTION.

Symbol	Value	Description
EP	500	Number of episodes in each run
α_0	$\alpha_0 = 2\times10^{-6} \approx (1/EP)\times(1/n)$	Initial learning rate
α_{ep}	$\alpha_{ep} = \alpha_0((n_0\times EP+1)/\ (n_0\times EP+ep))$	Episode learning rate
n_0	$75\%EP$	Start episode for decreasing α_{ep}
ε_0	0.5	Initial exploration rate
ε_{ep}	$\varepsilon_{ep} = \varepsilon_0((n_0\times EP+1)/\ (n_0\times EP+ep))$	Episodic exploration rate
n0ε	$50\%EP$	Start episode for decreasing ε_{ep}
γ	1	The reward discount factor
m	3	Number of snapshots of the home
b	3	Features histograms bin size
ψ_{upper}	0.96	Termination threshold
ψ_{lower}	$\psi_{lower}=\psi_{upper}$-0.02=0.94	NRB trust region threshold

The initial learning rate α_0 was set to $\alpha_0 = (1/EP)\times(1/n) \approx (1/500)\times(1/1000) = 2\times10^{-6}$ in accordance with the features size and the number of episodes. This to divide the learning between all features and all episodes to allow for good generalization and stochastic variations. The learning rate was decreased further from one episode to another, to facilitate learning and to obtain convergence for TD according to [34]. This is necessary in accordance with the study of TD convergence conducted by [35], which pointed out that decreasing the step size learning rate α is necessary so that the policy parameters $\vec{\theta}$ do not diverge. Although factor $\lambda_t^{(conj)}$ in TD-conj(0) is a counterpart of fixed λ in conventional TD(λ), however in contrast with λ it varies from one step to another to achieve better results. The discount constant was set to γ=1, i.e. the rewards sum does not need to be discounted through time because it is bounded given that the task ends after reaching the final state at time T.

ψ_{upper} ,ψ_{lower} are determined empirically and were set to 0.96, 0.94 respectively when using the NRB measure and settings b=m=3. They were set to 1.7 and 2 respectively when using the Jeffery Divergence Measure and when b=m=3. All the results obtained using the NRB measure will be stated; for results regarding the Jeffery Divergence Measure the reader is referred to [36]. Setting those

constants to 0.96 and 0.94 simply indicates that to terminate the episode the agent should be $\geq$ 96% sure (using the NRB similarity measure) that its current view corresponds with one or more of the stored views for the home to assume that it has reached the home location. Furthermore, it indicates that the agent should be $\geq$ 94% sure that its current view is similar to the stored views to assume that it is in the goal-on-sight region.

B. Convergence results

Fig. 5. shows the learning plots for the TD-conj(0)≡ TD(λ_t^{conj}) where the ${}^1\lambda_t^{(conj)}$ was used. Convergence is evident by the exponential shape of all of the plots. In particular the cumulative rewards converged to an acceptable value. The steps plot resonates with the rewards plot, i.e. the agent attains gradually good performance in terms of cumulative rewards and steps-per-episode. The cumulative changes made to the policy parameters have also a regular exponential shape, which suggests the minimization of required learning from one episode to another. It should be noted that although the learning rate is decreased through episodes, if the model is not converging more learning can be seen in later episodes, which will deform the shape of the changes in the policy parameters plot.

λ can take any value in the [0 , 1] interval. It has been shown by [28] that the best performance for TD(λ) is expected to be when λ has a high value close to 1, such as 0.8 or 0.9 depending on the problem. It should be noted that λ=1 is not a good candidate as it approximates Monte Carlo methods and has noticeably inferior performance than smaller values [28]. It should also be noted that the whole point of the suggested TD-conj(0) method is to optimize and automate the selection of λ in each step to allow TD to perform better and avoid trying different values for λ.

The temporal error for a sample episode is shown in Fig. 6 (a). Fig. 6 (b) shows the trace decay factor ${}^1\lambda_t^{(conj)}$ for the same sample episode. The overall actual exploration versus exploitation percentage is shown in Fig. 6 (c). It can be seen that for ${}^1\lambda_t^{(conj)}$, most of the values are above 0.5. As has been stated in the Equivalency Theorem, there is no guarantee that $\lambda_t^{(conj)}$ satisfies the condition $0 \leq \lambda_t^{(conj)} \leq 1$.

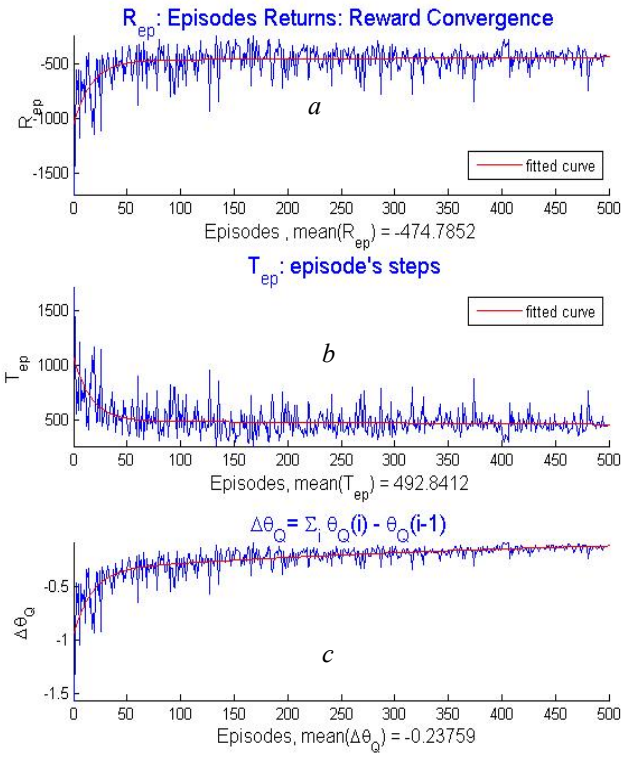

Fig. 5. TD-conj(0) for ${}^1\lambda_t^{(conj)}$ algorithms performance (a) The cumulative rewards. (b) The number of steps. (c): The cumulative changes in the learning parameters.

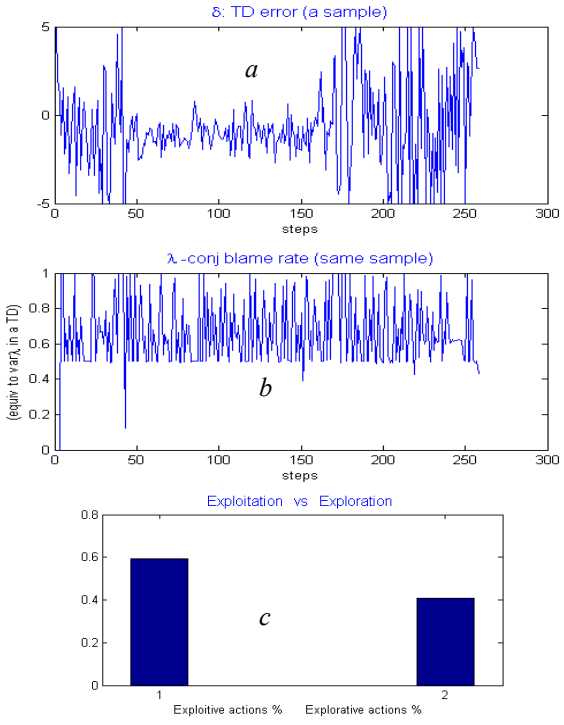

Fig. 6. TD-conj(0) algorithm internal variables (a): the temporal error for a sample episode. (b): the trace decay factor ${}_1\lambda_t^{(conj)}$. (c): the overall exploration versus exploitation.

Nevertheless, for most of the values this form of $\lambda_t^{(conj)}$ does satisfy this condition. For those values that did not, it is sufficient to apply the following rule on them:

$$if(\lambda_t^{(conj)} > 1) \Rightarrow \lambda_t^{(conj)} \leftarrow 1, \quad if(\lambda_t^{(conj)} < 0) \Rightarrow \lambda_t^{(conj)} \leftarrow 0 \qquad (14)$$

It should be noted that better performance could have been achieved by the following rule:

$$if(\lambda_t^{(conj)} > 1) \Rightarrow \lambda_t^{(conj)} \leftarrow 1 - \xi, \qquad if(\lambda_t^{(conj)} < 0) \Rightarrow \lambda_t^{(conj)} \leftarrow 0 \qquad (15)$$

However, using this rule would mean that the results shown for TD($\lambda_t^{(conj)}$) might have been affected by the higher performance expected for TD update when $\lambda_t^{(conj)}$ is close (but not equal) to 1. This is because for one single update at some time step t TD(λ) and TD($\lambda_t^{(conj)}$) are identical for the same $\lambda_t^{(conj)}$ value. It is the collective variation form one TD(λ) update to another at each time step that makes TD($\lambda_t^{(conj)}$) different from TD(λ). Therefore, a better performance could have been achieved by the following rule (15). Hence the performance of TD($\lambda_t^{(conj)}$) could be dubious when rule (14) is used. Few values did not satisfy the Theorem of equivalence condition - the percentage was 0.0058% for TD($^1\lambda_t^{(conj)}$).

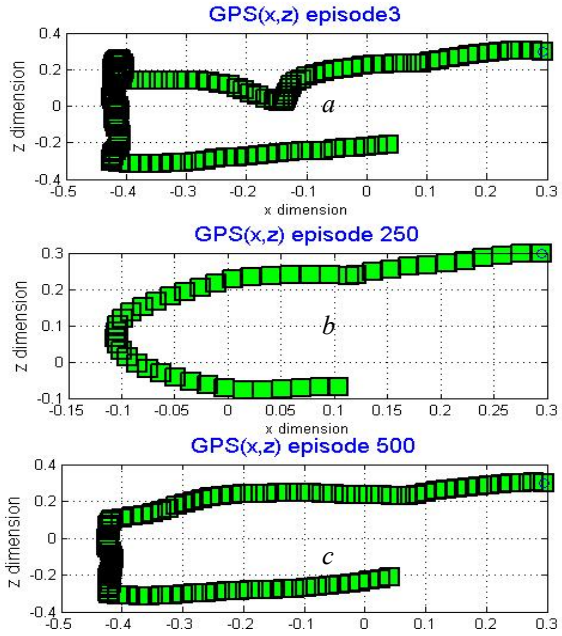

Fig. 7. GPS plots for early, middle and end episodes, that show the trajectory improvement that took place during learning for TD($^1\lambda_t^{(conj)}$) algorithm.

IV. Novelty Aspects

There are various aspects of novelty, but the two main ones are concerned with the learning method and the model. The new TD-conj method is used in an existing RL control algorithm, namely Sarsa. The algorithm is applied in a novel RL model designed for visual robot homing.

The use of a whole image measure as a means of termination and to augment the reward signal coming from the environment is one element of novelty. The simple NRB is a newly established measure shown to be effective. This aspect conforms with the latest finding in cognition which asserts that landmarks are not necessary for homing [19]. The home location is defined through *m* snapshots. This, together with the use of a whole image measure such as NRB, allows for robust task execution with minimal required knowledge about the target location and its environment.

On the other hand, it was realized that there is a need to boost RL algorithms that use function approximation. A new family of RL methods , TD-conj(λ), has been established [29] by using the conjugate gradient direction instead of the gradient direction in the conventional TD(λ) with function approximation. Since TD-conj(0) is proved to be equivalent to TD(λ) where λ is variable and is denoted as λ_t^{conj} [29], this family is used in the proposed model as the learning algorithm. The novelty of this work can be summarized in three main points. The use of a whole image measure in a reinforcement learning model to establish more general model that can be used in virtually any indoor environment. The combination of RL whole image measure and visual radial basis function. The use of the new TD(λ_t^{conj}) method to learn the homing task.

Some of the advantages of the novel method and model can be summarized in the following points. Simplicity of learning: the robot can learn to perform its visual homing (sub-navigation) task in a simple way, without a long process of map building. Only limited storage of information is required in the form of *m* stored views. No pre- or manual processing is required. No *a priori* knowledge about the environment is needed in the form of landmarks. An important advantage of the proposed model over MDP model-based approaches is that abduction of the robot is solved directly, i.e. the robot can find its way and recover after it has been displaced from its current position and put in a totally different position. Other models that use algorithms such as the particle filter [25] can recover from this problem but not as quickly as the proposed model. This is because the current view, alone, gives enough information for the trained neural network to decide immediately on which action to take, while multimode filters (such as the particle filter, or an unscented Kalman filter) may take two or more steps to know where the agent is, and then to take a suitable action depending on its location.

V. Conclusion and Future Work

A new robust learning model for visual robot homing (VRH) was developed, which reduces the amount of required *a priori* knowledge and constraints associated with

the robot's operating environment. Other models requires a landmarks [16], [17] or a separate stage and extensive assessment of tens of pictures in the environment before being able to operate in the environment [6]. While, the proposed model only requires taking 3 snapshots of the environment. This was done by employing a reinforcement learning method for control and appropriate linear neural networks, in combination with a reward signal and termination condition based on a whole image measure.

The proposed model is an attempt to contribute to filling the gap of the lack of models for homing that are practically adaptive to any indoor environment a robot operates in. It does not require human intervention in the learning process, neither it assumes that those environments should be artificially adapted for the sake of the robot. This study shows that visual homing based on RL and whole image techniques can offer generality and automated learning properties. In future research, a number of enhancements are planned to the model. Although setting up a suitable exploration/exploitation was automated in the model and needed only specifying ε_0 and $n_{0\varepsilon}$ prior to execution, finding the best balance between these parameters will be a topic for future research. Further, reducing the number of the learning parameters is another issue that is being looked at.

REFERENCES

[1] U. Nehmzow, *Mobile robotics: A Practical Introduction*: Springer-Verlag, 2000.

[2] R. C. Arkin, *Behavior-Based Robotics*: MIT Press., 1998.

[3] R. R. Murphy, *Introduction to AI Robotics*. Cambridge, Massachusetts.: The MIT Press, 2000.

[4] M. Asadpour and R. Siegwart, "Compact Q-learning optimized for micro-robots with processing and memory constraints," *Robotics and Autonomous Systems, Science Direct, Elsevier*, 2004.

[5] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains.," *Artificial Intelligence*, vol. 101, pp. 99–134., 1998.

[6] M. Szenher, "Visual Homing with Learned Goal Distance Information," presented at Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005), Awara-Spa, Fukui, Japan, 2005.

[7] A. Vardy and R. Moller, "Biologically plausible visual homing methods based on optical flow techniques," *Connection Science*, vol. 17, pp. 47–89, 2005.

[8] S. Thrun, "Probabilistic Algorithms in Robotics," CMU-CS-00-126. , Technical Report 2000.

[9] N. Tomatis, I. Nourbakhsh, and R. Siegwart, "Combining Topological and Metric: a Natural Integration for Simultaneous Localization and Map Building," presented at Proc. Of the Fourth European Workshop on Advanced Mobile Robots (Eurobot 2001), 2001.

[10] R. Simmons and S. Koenig, "Probabilistic Robot Navigation in Partially Observable Environments," presented at Proc .of the International Joint Conference on Artificial Intelligence. , 1995.

[11] A. M. Anderson, "A model for landmark learning in the honey-bee," *Journal of Comparative Physiology A* vol. 114, pp. 335-355, 1977.

[12] B. A. Cartwright and T. S. Collett, "Landmark maps for honeybees," *Biological Cybernetics*, vol. 57, pp. 85-93, 1987.

[13] K. Weber, S. Venkatesh, and M. Srinivasan, "Insect-inspired robotic homing," *Adaptive Behavior*, vol. 7, pp. 65-97, 1999.

[14] A. Vardy and F. Oppacher, "A scale invariant local image descriptor for visual homing," in *Biomimetic neural learning for intelligent robots.* , G. Palm and S. Wermter, Eds.: Springer, 2005.

[15] A. A. Argyros, K. E. Bekris, and S. C. Orphanoudakis, "Robot Homing based on Corner Tracking in a Sequence of Panoramic Images," presented at 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01) 2001.

[16] C. Weber, S. Wermter, and A. Zochios, "Robot docking with neural vision and reinforcement," *Knowledge-Based Systems, Science Direct, Elsevier* vol. 17, pp. 165-172, 2004.

[17] D. Muse, C. Weber, and S. Wermter, "Robot docking based on omnidirectional vision and reinforcement learning," *Knowledge-Based Systems, Science Direct, Elsevier* vol. 19, pp. 324-332 2006.

[18] A. Vardy, "Long-Range Visual Homing," presented at IEEE International Conference on Robotics and Biomimetics, 2006. ROBIO '06., Kunming, 2006.

[19] S. Gillner, A. M. Weiß, and H. A. Mallot, "Visual homing in the absence of feature-based landmark information," *Cognition*, vol. 109, pp. 105-122, 2008.

[20] I. Ulrich and I. Nourbakhsh, "Appearance-Based Place Recognition for Topological Localization " presented at IEEE International Conference on Robotics and Automation San Francisco, CA, 2000.

[21] Y. Rubner and et al., "The Earth Mover's Distance as a Metric for Image Retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99-121, 2000.

[22] D. Sheynikhovich, R. Chavarriaga, T. Strosslin, and W. Gerstner, "Spatial Representation and Navigation in a Bio-inspired Robot," in *Biomimetic Neural Learning for Intelligent Robots*, S. Wermter, M. Elshaw, and G. Palm, Eds.: Springer, 2005, pp. 245-265.

[23] C. Weber, D. Muse, M. Elshaw, and S. Wermter, "A camera-direction dependent visual-motor coordinate transformation for a visually guided neural robot," *Knowledge-Based Systems, Science Direct, Elsevier* vol. 19, pp. 348–355, 2006.

[24] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *International Journal of Robotics Research*, vol. 23, pp. 693–716, 2004.

[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, Massachusetts; London, England: The MIT Press, 2005.

[26] P. Stone, R. S. Sutton, and G. Kuhlmann, "Reinforcement learning for robocup soccer keepaway," *International Society for Adaptive Behavior* vol. 13, pp. 165–188, 2005.

[27] A. Lazaric, M. Restelli, and A. Bonarini, "Reinforcement Learning in Continuous Action Spaces through Sequential Monte Carlo Methods," presented at NIPPS 2007, 2007.

[28] R. S. Sutton and A. Barto, *Reinforcement Learning, an introduction*. Cambridge, Massachusetts: MIT Press, 1998.

[29] A. Altahhan, "Conjugate Gradient Temporal Difference Learning for Visual Robot Homing," in *Faculty of Computing and Engineering*, vol. Ph.D. Sunderland: University of Sunderland, 2008, pp. 206.

[30] V. Konda and J. Tsitsiklis, "Actor-critic algorithms. ," presented at NIPS 12, 2000.

[31] J. Peters, S. Vijayakumar, and S. Schaal, "Natural Actor-Critic," *Proceedings of the Sixteenth European Conference on Machine Learning*, pp. 280–291, 2005.

[32] D. Floreano and F. Mondada, "Hardware solutions for evolutionary robotics," presented at First European Workshop on Evolutionary Robotics, Berlin, 1998.

[33] O. Michel, "Webots: Professional Mobile Robot Simulation," *International Journal of Advanced Robotic Systems*, vol. 1, pp. 39-42, 2004.

[34] J. A. Boyan, "Least-squares temporal difference learning " presented at Proceedings of the Sixteenth International Conference on Machine Learning San Francisco, CA 1999.

[35] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control* vol. 42, pp. 674–690, 1997.

[36] A. Altahhan, K. Burn, and S. Wermter, "Visual Robot Homing using Sarsa(λ), Whole Image Measure, and Radial Basis Function," presented at IJCNN HONG KONG, 2008.