Towards a deep feature-action architecture for robot homing

Altahhan, A.

Original citation & hyperlink:
Altahhan, A. (2015) 'Towards a deep feature-action architecture for robot homing' in 2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM) (pp: 205 - 209). IEEE
http://dx.doi.org/10.1109/ICCIS.2015.7274621

ISBN 978-1-4673-7337-1
DOI 10.1109/ICCIS.2015.7274621

Towards A Deep Feature-Action Architecture for Robot Homing

Abdulrahman Altahhan, *Member, IEEE*

***Abstract*— This paper describes a model for robot navigation that uses an architecture similar to an actor-critic reinforcement learning architecture. Contrary to the abundance of models that use two neural networks one for the actor and one for the critic, this model sets up the actor as a layer seconded by another layer which deduce the value function. Therefore, the effect is to have similar to a critic outcome combined with the actor in one network. Hence, the model paves the way for a deep reinforcement learning architecture for future work. The reward signal is back propagated through the critic then the actor. At the same time, the features layer have been deeply trained by applying a simple PCA on the whole set of images histograms acquired during the first running episode. The model is then able to shrink the whole architecture to fit a new reduced features dimension. Initial experimental result on real robot shows that the agent accomplished good level of accuracy and efficacy in reaching the goal.**

I. Introduction

Robot homing is considered one of the important special cases of navigation. It pertains to all animals naturally and is a must for most of the commercial and entertaining robotics application. Central to this ability is the skill of orienting towards the home and recognizing it once the agent is around it. Animals do that by wiring the surrounding visual memory somehow to their neural map of the environment. How they do that is yet to be discovered. Traditionally it has been linked to distinctive position or places in the environment i.e. landmarks. However, the way animals do its navigation and find their home suggests something more subtle than only landmarks [1, 2, and 3]. In this paper it is argued that a plausible proposition is that the brain hard wire the scenes to itself and compare it with the look of the home. It forms a frame of reference which is used to compare all information passed through to obtain internal map of the home as opposed to the different paths to its home. It is argued that the animal cannot remember all paths to its home neither it can remember the links between the landmarks consciously. It simply react to what it sees once it set back home and it dose not normally plan it unless something novel happened. It uses this internal representation to guide itself along with other visual aids such as landmarks for the long distance. However for the short distance it uses this internal map solely.

Abdulrahman Altahhan is with the Computing Department of Coventry University, CV1 5FB, UK (phone: +44 2477653088; e-mail: abdulrahman.altahhan@ coventry.ac.uk).

II. Reinforcement Learning

A. Reinforcement learning framework

Reinforcement Learning, or RL, is considered ideal for learning novel scenario where it is impractical to obtain a model of the environment [4]. This is the norms for animals once they face a new situation. In that moment the actual learning begins, and we are interested to know what could be going on at that moment.

RL does not require planning, but planning-like behavior comes as a natural consequence of it [4]. Actor-critic architecture have been proposed and studied in plenty of scenarios [5, 6 and 7] that ranges from simulation to real problems and recently an interesting development has been proposed [8]. The actor-critic architecture is interesting due to the fact that it allows for explicit natural separation of concern between a performer, that tries to learn the best set of actions in certain situations, and a critic that tries to maximise overall future gain strategically. For the learner everything can be summed up in terms of a reward signal (food, shelter etc.) and the sensory data that feedback to it, as well as the actuators that the animal needs to use in order to reach its target and maximize the long and short term rewards. Shelter is considered a complex but important urge in animals. It comes after food as it is less urgent in the short term however it could be very important in the long term and could be vital in some dangerous situations. Therefore, animals have developed a complex behavior around the shelter need and through evolution it becomes intrinsic in the brains of higher animals. To trace all that back one needs to understand how primitive rewards form skills that governs complex behavior.

B. Deep learning and RL

Deep learning has been shown to overcome the bottleneck representation problem that has long set back the success of machine learning applications [9, 10]. It is especially important for RL since RL normally takes a long time to converge due to the fact that there is no direct answer to the input, the answer is just a signal that indicates how good or bad the current action is (in the long run), and hence how good or bad the overall behavior is [14]. Deep learning showed better results when combined with supervised learning [11, 12]. When combined with RL it is also believed to have a good potential.

III. The Model

Our model starts by learning a concise and reduced feature representation. The model obtains a reduced representation in the first episode by applying a simple PCA on the whole set of images acquired during this first

episode(s). This explorative episode aligns with how animals normally explore a place by looking around. Then the model shrinks the whole architecture to fit the new reduced number of features, and this concludes the deep learning phase for the feature representation layer. It should be noted that other methods could be better for object detection or recognizing a certain pattern such as a hand-written digits [13]. However, for the purposes of our model we need something fast and low-level and we do not need to obtain features that could be used for recognizing an object. We just need a set of features that is good enough to distinguish the goal from other locations.

A. Model Architecture and Components

Next is the turn of Actor-Critic architecture [8]. After the Actor layer takes its input form the PCA layer, it then decides to do a certain action, accordingly the Critic layer punishes or rewards the Actor depending on the reward it receives form the environment.

Formally, the presented model uses the following components/stages:

- Goal representation: As opposed to many models the goal or the home is represented by just a snapshot taken for that location with the desired orientation of the robot. In fact the method used to identify the goal is transitionally invariant. Hence the goal location could in deed by identified by the agent from an angle different to the one it has originally taken from as we shall emphasize later by our stopping condition.

Feature vector is calculated in two stages

- The preliminary stage is to learn a reduced feature vector by deducing a representative Eigen vectors that gives results similar to a simple autoencoder. This is done by running the robot in an explorative episode(s) and analyzing the different scenes images in each step in an online fashion where the mean of the entire Eigen vector set is calculated at the end of the episode. And the max dimension is taken in case of more than one explorative episode is performed by the agent. This stage can be extended to span more than one episode to give a stable acceptable dense sample of the environment. However almost always the agent had to spend relatively long time in the first episode due to the fact that the weights are initialized to completely random small values (close to zero), hence they encouraged disoriented behavior that is explorative by nature. Then the model shrinks the whole architecture to fit the new reduced number of features (by picking Eigen vectors that corresponds to Eigen values of certain significance). This concludes the deep learning stage for the feature representation layer. This stage is done once and will not be repeated.

- The primary step in which the Eigen vectors deduced in the preliminary stage is used to calculate a reduced features vector. This step helped in focusing the policy learning step which is inevitably much longer.

In both stages the initial features used are differential Radial Basis features [14] that make the goal image its referential point and make all the views relative to that goal. This is consistent with home-aware localization and allows the agent to view the world from the perspective of its current homing task. The initial features are given by:

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

Where $h_f\left(v(c,j)\right)$ is histogram, bin f of channel $v(c,j)$, and $h_i\left(s_t(c)\right)$ is histogram bin f of channel c of current image. Using the mean Eigen vector that has been calculated in the explorative episode, the new reduced features are given by

$$F_k = \Omega_{i,f} \times \phi_f\left(s_t(c,j)\right) \quad (2)$$

The dimension of the features ϕ_f is $d_1 \times d_2 \times 3$ where d_1 and d_2 are the dimensions of the images and 3 coming from having three channels. The dimension of the reduced features F_k is $n << d_1 \times d_2 \times 3$.

- Another component of the model is a similarity measure that specifies the termination of the episode and is given by

$$NRB(s_t) = \sum_{k=1}^{n} \vec{F}_k\left(s_t\right) \Big/ n \quad (3)$$

- This measure has been used along with two thresholds to set the stopping and approaching conditions for the agent. In particular, circular statistics should be used for this purpose as we want the agent to stop regardless of the angle that it faces the goal with [15]. Von Mises distribution provides such framework; it has nice properties of being easier to deal with and being an approximation to the warped normal distribution (the Gaussian counterpart in circular statistics) [16]. It is capable of dealing with directions and rotation in multidimensional spaces.

$$\psi_{upper} = 1 - \left(\pi\sqrt{mult}\right)^{-1}, \quad \psi_{lower} = 1 - \left(\pi\sqrt{mult - 1}\right)^{-1} \quad (4)$$

We chose multipliers 9 and 10 (through trial and error) that yield the values 0.894% 0.879% respectively.

- The reward signal is calculated using the weights of the critic which is constituted of three parameters in accordance with the number of actions allowed in this model. These actions are forward, left and right respectively, where left and right have been set to equal speeds. The reward function is given as a combination of step cost in addition to a reward for going towards (approaching) the goal as well as a reward for reaching the goal itself (which is proportional to how fast the agent reached the goal in terms of number of steps) as in [14]. Further a higher cost has been associated with turning actions. I.e. when the agent turns it will acquire higher costs than

when it goes strait. This had the desired consequence of supressing unnecessary turns and emphasising going strait. Also a punishment for taking any action that leads to a reactive behaviour has been set. This is also to help reduce the costal behavior and to encourage going directly towards the goal.

C. Actor-critic Combined Network with Double Eligibility Traces

In this section we show the derivation of the learning formulae for the layered actor-critic architecture. The error function [4] can be written as

$$Er_t^2(s) = \left[V^{\pi}(s) - V_t(s)\right]^2 \tag{5}$$

Using two layered perceptron (one hidden layer and an output layer) and two sigmoid activations we have:

$$V_t(s_t) = \frac{1}{1+e^{\sum_{i=1}^{I} Q_t(i) w_t(i)}} \tag{6}$$

$$Q_t(i) = \frac{1}{1+e^{\sum_{k=1}^{K} F_t(k) \theta_t(i,k)}} \tag{7}$$

The update rules for weights that go opposite to the gradient direction are:

$$\Delta w_t(i) = \alpha_t \left(V^{\pi}(s_t) - V_t(s_t)\right) \frac{\partial V_t(s_t)}{\partial w_t(i)} \tag{8}$$

$$\Delta \theta_t(i,k) = \alpha_t \left(V^{\pi}(s_t) - V_t(s_t)\right) \frac{\partial V_t(s_t)}{\partial \theta_t(i,k)} \tag{9}$$

Two layers with forward view using R_t^{λ} as an approximation of $V^{\pi}(s_t)$ [4] are:

$$\Delta w_t(i) = \alpha_t \left(R_t^{\lambda} - V_t(s_t)\right) \frac{\partial V_t(s_t)}{\partial w_t(i)} \tag{10}$$

$$\Delta \theta_t(i,k) = \alpha_t \left(R_t^{\lambda} - V_t(s_t)\right) \frac{\partial V_t(s_t)}{\partial \theta_t(i,k)} \tag{11}$$

By bootstrapping and using the $r_{t+1} + \gamma V_t(s_{t+1})$ as an approximation for $V^{\pi}(s_t)$ we have:

$$\Delta w_t(i) = \alpha_t \delta_t \frac{\partial V_t(s_t)}{\partial w_t(i)} \tag{12}$$

$$\Delta \theta_t(i,k) = \alpha_t \delta_t \frac{\partial V_t(s_t)}{\partial \theta_t(i,k)} \tag{13}$$

Where: $$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \tag{14}$$

Two layers with two eligibility traces $e_t(i)$, $e_t(i,k)$ and backward view we have:

$$\Delta w_t(i) = \alpha_t \delta_t e_t(i) \tag{15}$$

$$\Delta \theta_t(i,k) = \alpha_t \delta_t e_t(i,k) \tag{16}$$

$$e_t(i) = \gamma \lambda e_{t-1}(i) + \frac{\partial V_t(s_t)}{\partial w_t(i)} \tag{17}$$

$$e_t(i,k) = \gamma \lambda e_{t-1}(i,k) + \frac{\partial V_t(s_t)}{\partial \theta_t(i,k)} \tag{18}$$

It should be noted that eligibility traces in reinforcement learning framework is similar to the momentum for supervised learning. It constitutes a way to accommodate previous updates into current updates to guide the search for the local optima. In RL it traces blame of current decision back to older decisions that lead to the current situation. In addition, two regularizers have been multiplied by the two parameter sets to discount the old values of the parameters (hence prevent ovefitting). Also the updates have been committed in a mixture of online and off-line fashion.

B. Deep Blended Actor-critic Architecture

By setting the second layer to three parameters; one for each action and calculating the error signal for the actor layer (which is responsible for taking the actions; its decision is the one that is being carried out). And by allowing this second layer to act as a critic that contemplates the consequences of the actions of the actor layer and sends a signal to it to indicate how well its current policy is. And by making the two layers to work as hidden and output layers of a one neural network, we are creating deep blended actor-critic architecture in one sound system that depends on two eligibility traces. The value function layer (we are calling it critic layer but it is not in the strict sense as it does not take input directly from the states so one can call it an evaluator layer as well) itself is taking its feedback form the reward function. The action layer can learn independently form the critic layer by utilizing an action-value function approach while the critic layer cannot act independently, it still needs the actor layer to calculate the estimated value function. In that sense the learning process can be thought of as a layer by layer learning or deep learning enabled. In the future we will explore training in each layer independently by freezing learning in each layer and then fine-tune by utilizing the presented approach. So this model is deep in terms of its feature representation and has the potential to be deep in terms of its action representation. All training is done using backpropagation. Finally the policy has been formulated as a simple greedy police.

IV. Experimental Results

The robot was let to run for 30 episodes. Each episode starts by going from any location in the environment to the goal/home location. The size of the robot makes it relatively easier to run it form different locations. Hence, it was allowed to run for a 500 steps before the episode is considered a failure.

A. Hardware and Software Settings

Fig. 1 shows the used robot and its environment. It is basically an updated version of Lego Mindstorm that has been used with additional camera module and processing unit that was mounted and attached on top of it. This robot has relatively a low level of sophistication in terms of the motor commands, balance, senor reading as well as its shape. Yet the results were good, so it is expected to obtain higher performance once a finer robot it used. The Raspberry Pi has been powered by an off-the-shelf 1000AMP chargeable battery that was placed underneath the NXT brick. The sound sensor has been set up for external rewards/punishment but has not been used in the experiments presented her.

Matlab have been used throughout the model in the form of a set of library functions that have been written specifically for this model. In addition the RWTH-Mindstorms NXT Toolbox for MATLAB has been used to provide the sensory reading and the actuator commands form the NXT robot to Matlab.

Figure 1. *Left:A snapshot of the built robot with its sensors, actuators, and camera module. Right: The training envirnment.*

B. Model Hyper Parameters Settings

It should be noted that the robot abducts itself after each successful/unsuccessful episode by following a rigid set of backwards steps that formed a U-shape so that it is as far and disoriented form the goal as possible. Each episode stops by either reaching the goal (successful) or by exhausting the allowed budget (number of steps) that the agent is given (unsuccessful). In the case of the presented work it was 500 steps. The settings of the model hyper parameters are shown in Table1

Table 1. Parameters of The Model

Symbol	***Value***	***Description***
Max_e	30	Number of episodes in each run
α_0	0.1	Initial learning rate
ε_0	$0.3\times EP$	Initial exploration rate
ep_0	$0.3\times EP$	Start episode for decreasing α and ε
γ	1	The reward discount factor
m	1	Number of snapshots of the home
b	2	Features histograms bin size
ψ_{upper}, ψ_{lowe}	0.88 0.87	Goal_at_perspective thresholds
λ	0.8	Eligibility trace discount
Max_steps 1	500	steps before agent considered unsuccessful

Images with resolution of 160x120 were sent form the Raspberry PI via wireless network adaptor to an off-board computer for processing where learning is taking place, then the required commands is sent to the actuators of the robot via Bluetooth.

C. Convergence and Performance

Fig. 2 shows an intermediate stage where the robot was still learning. The number of episodes (upper right corner) is envisaged (as was evident in the simulation in [14, 15 and 16]) to show a pattern of convergence towards minimal number of step if the robot where left to run for a very long time.

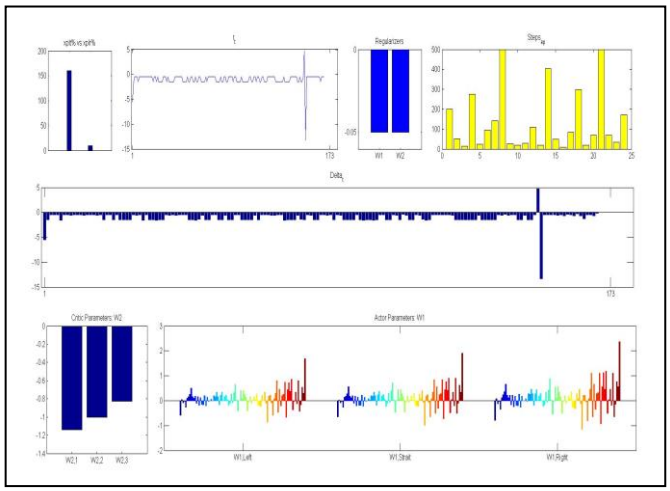
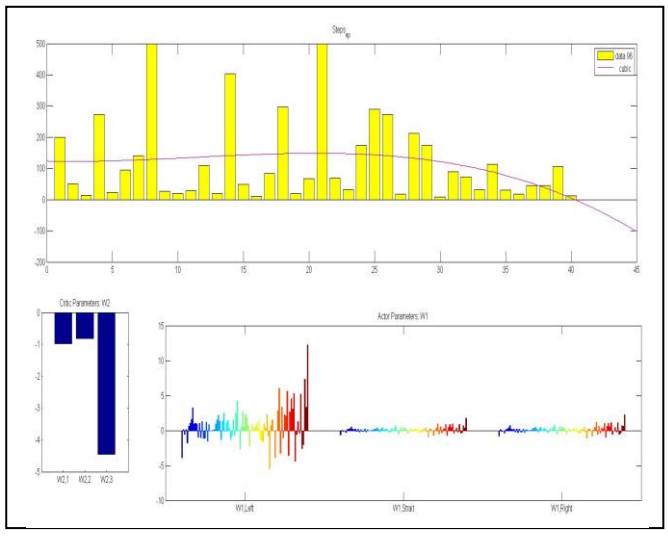


Figure 2. *The model learned parameters; a tendency towards turning left is developed by the agent, which is what is expected when operating in an*

However, due to the time and physical constraints, this was difficult to do. Hence it has been let to run for a limited number of episodes. On the other hand, it should be noted that the number of steps varied between episodes due to the abduction of the robot to a random location after reaching the home, which resulted near/further position from the goal. Nevertheless, a cubic curve fitting is shown in Fig. 3 that



illustrates possible convergence, which is evident indeed.

Figure 3. *The model learned parameters and episodes number of steps*

Fig. 3 shows the learned parameters, a tendency towards turning left is developed by the agent, which is what is expected when operating in an open plan. It should be noted however that the agent did not just always turned left, the behavior depends on the current image/position.

V. CONCLUSION

Our results show that out of 40 times (30 trainings + 10 testing) it mixed the goal twice. This has been verified by looking into what the robot has registered as a target in each episode. In addition from figure 3 it can be seen that the agent runs out of time in two episodes also, i.e. it reaches maximum number of steps. These are different than the one which it has mixed. Hence, accuracy in identifying the goal (training and testing) is 1-2/38 = 947% and efficacy in reaching the goal in the allowed steps is (40-4)/40= 90%. As opposed to many models the goal or the home is represented by just a snapshot taken for that location with the desired orientation of the robot. The model uses deep learning for feature representation, which set its distinctive novelty. How practical is it, will be for future work to verify. Also it is intended to show some other interesting properties of the model such as convergence and the relationship between deep feature learning and deep action learning.

REFERENCES

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

[2] N. Tomatis et al, "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.

[3] Jochen Zeil, "Visual homing: an insect perspective, Current Opinion in Neurobiology", Volume 22, Issue 2, pp. 285-293, ISSN 0959-4388, April 2012

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

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

[6] O. Ziv and N. Shimkin, "Multigrid Methods for Policy Evaluation and Reinforcement Learning", presented at IEEE International Symposium on Intelligent Control, Limassol, 2005.

[7] C. Zhang at al, "Efficient multi-agent reinforcement learning through automated supervision", presented at International Conference on Autonomous Agents Estoril, Portugal, 2008.

[8] S. Bhatnagar et al, "Incremental Natural Actor-Critic Algorithms", presented at Neural Information Processing Systems (NIPS19), 2007.

[9] G. Hinton et al, "A fast learning algorithm for deep belief nets". Neural Computation,18(7):1527–1554, 2006.

[10] A. Coates et al, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning", in AISTATS 14, 2011.

[11] P. Vincent et al, "Extracting and composing robust features with denoising autoencoders". In ICML, 2008

[12] Andrew Ng et al (2010), Tutorial in Deep Learning: Stanford University [Online]. Available: http://ufldl.stanford.edu/tutorial/

[13] Y. LeCun et al, "Learning methods for generic object recognition with invariance to pose and lighting". In CVPR, 2004.

[14] A. Altahhan, "A Robot Visual Homing Model that Traverses Conjugate Gradient TD to a Variable λ TD and Uses Radial Basis Features", in Advances in Reinforcement Learning, A. Mellouk, Ed. Vienna: InTech Education and Publishing, 2011, pp. 225-254.

[15] A. Altahhan, "Robot Visual Homing using Conjugate Gradient Temporal Difference Learning, Radial Basis Features and A Whole Image Measure", International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, ISBN: 978-1-4244-6916-1, 2010.

[16] A. Altahhan et al, "Visual Robot Homing using Sarsa(λ), Whole Image Measure, and Radial Basis Function", International Joint Conference on Neural Networks (IJCNN), Hong Kong, 2008