# Reinforcement Learning for Platform-Independent Visual Robot Control

David Muse, Kevin Burn and Stefan Wermter

Centre for Hybrid Intelligent Systems
School of Computing and Technology, University of Sunderland
www.his.sunderland.ac.uk

***Abstract*—This paper proposes a new architecture for robot control. A test scenario is outlined to test the proposed system and enable a comparison with an existing system, which is able to fulfil the scenario and thus be used as a benchmark. The scenario is a navigation task, to allow a robot to approach a specified landmark. The proposed architecture will make use of two control units, one to allow a pan/tilt camera to track the landmark as the robot moves, and a second to control the robots drive motors. These units will be trained via reinforcement learning, and provide the potential for platform-independent robot control.**

## I. Introduction

IN the past several different systems have been developed employing reinforcement learning for robot control. These systems have used different input modalities, from coordinates provided in simulated environments [1], to visual input for simulated or real world robot control [2]-[4]. Reinforcement learning is a technique, which uses reinforcement signals to train the 'agent' (a network, robot etc. to learn sequence policies). The reinforcement signal is given when the 'agent' achieves or misses the goal. Initially the 'agent' explores its environment randomly until it achieves the set goal, then as the 'agent' learns, the exploration becomes more structured (see [5] for a good introduction to some reinforcement techniques).

The following Section provides an overview of several systems developed to control mobile robots using reinforcement learning. All of these systems have been designed to control a specific robot platform. To control a different robot platform the systems would require redevelopment. However, there are commonalities with many robots; these can potentially be used in conjunction with reinforcement learning to develop platform-independent control systems.

D. Muse is a PhD student at the University of Sunderland and is a member of the Hybrid Intelligent Systems Research Group (phone: +44-191-515-3291 fax: +44-191-5153-461 e-mail: david.muse@sunderland.ac.uk).

Dr. K Burn is a Senior lecturer at the University of Sunderland (e-mail: kevin.burn@sunderland.ac.uk)

Prof. S. Wermter is the Chair of the Hybrid Intelligent Systems research group at the University of Sunderland (e-mail: stefan.wermter@sunderland.ac.uk)

The focus of this paper is to develop a platform-independent control system for a robot navigation task, which couples pan/tilt cameras to gather visual information and reinforcement learning. The pan/tilt camera will be used to locate a predefined landmark. The centre point of this landmark will then be used to allow the camera to keep it in the centre of its 'gaze'. With the landmark centred in the image the pan and tilt angles of the camera can be used to deduce the relative position of the landmark to the robot, and be fed into a motor control unit to allow the robot to approach the landmark. Reinforcement learning will be used to train two control units, one to keep the landmark in the centre of the camera image and a second to control the robots drive motors.

The remainder of the paper is structured as follows. Section II provides an overview of existing systems developed for robot control tasks using reinforcement learning. Section III highlights the scenario for the control task, followed by a discussion of an existing system developed for the scenario, complete with results. In Section IV a discussion of the proposed architecture is provided, starting with the motivation for the work. This is followed by a discussion of the architecture to address the defined scenario. The final part of section IV discusses the robotic platforms on which the proposed architecture will be tested. Section V provides a discussion on the possibilities of the proposed architecture with possible extensions. A conclusion to the proposed work is provided in section VI.

## II. Overview Of Existing Systems

Reinforcement learning has been successfully applied to different robot control tasks, and an overview of some of these systems is given here. It has been used to control a simulated robot in the field of robot soccer [1]. The paper concentrates on the results Hafner and Riedmiller obtained from the simulated environment, however they also state that they are working on adapting the system to work on their omnidirectional robot. They highlight a common problem which is the length of time it took to learn the defined tasks. This problem needs to be considered when using reinforcement learning on robots.

As part of a European funded project to study Mirror

Neurons we developed a visual system; the system allowed a PeopleBot robot to pick an object from a table [2]. This system used visual input and reinforcement learning to position the robot with the object between its grippers. However, it had a limited range, as the object needed to be in the visual field of the camera at all times during the run. The camera was also held in a fixed position, so was unable to track the object if it moved out of view. Work has been undertaken to extend this system [3],[4].

We extended the system by allowing the pan/tilt camera to move [3]. This increased the range of the robot vision, as the camera could track the object. We focused on the network developed to perform a coordinate transform to deduce the angle and distance to the object in robot centred coordinates. These values were used as the inputs to the reinforcement network. However, the system has not yet been tested on a real robot. The use of the coordinate transform network introduces possible errors to the system and increases the complexity.

We provided an alternative extension, using an omnidirectional camera [4]. This was successfully implemented on the PeopleBot and used a sequence of reinforcement subsystems to allow the robot to reach the object from a greatly increased range. Here, the original network was unaltered and a new network was used to locate the object via omnidirectional vision and a specified landmark. Once at the landmark the object was visible in the pan/tilt camera. However, as the original network was used, the camera remained stationary and if the object moved out of sight, the system would still be unable to track the it. To avoid a lengthy online training time, the network was initially trained on a simulator. This was then exported to the robot for the final training, which fine tuned the network to control the motor system of the robot. This principle will be used by the system proposed in this paper.

Sierra et al describe their work, where a simulated robot is controlled using various reinforcement learning algorithms [6], coupled with the control architecture described in [7] for the sharing of the visual component. Here the vision system is competed for by the different subsystems such as the navigation system to allow the robot to move to the landmark or the pilot system which performs obstacle avoidance.

Gaskett et al developed a control system for a robot using a fixed camera, and used reinforcement learning to remove the need for camera calibration [8]. All of these systems have demonstrated the applicability of reinforcement learning to robot control problems. However, the main problem faced by reinforcement learning is the 'curse of dimensionality': increasing the number of inputs increases the dimensionality of the state space, which is used to model the input variables. We avoid this problem in our proposed architecture as the subsystems have been designed to only require two inputs to avoid the curse of dimensionality.

## III. Scenario and Initial Work

### A. Scenario

The initial scenario that will be used to test the proposed architecture is similar to that used by Gaskett et al [8]. The robot will be placed at a random starting position with a direct line of sight to the specified landmark and will be required to move to the landmark. To allow tracking of the landmark as the robot moves, the camera will be free to pan/tilt. With the camera being free, two control units are required: one for the camera to keep the landmark in the centre of the image and another for motor control, allowing the robot to move to the landmark. Once the robot is able to move to the specified landmark the system can be extended, by compiling a library of landmarks could be used to allow the robot to perform topological navigation [9], moving from one landmark to the next.

### B. Learning Robot Control using an Omnidirectional Camera

A system has been designed based on omnidirectional vision and reinforcement learning [4]. A brief overview of the system follows.

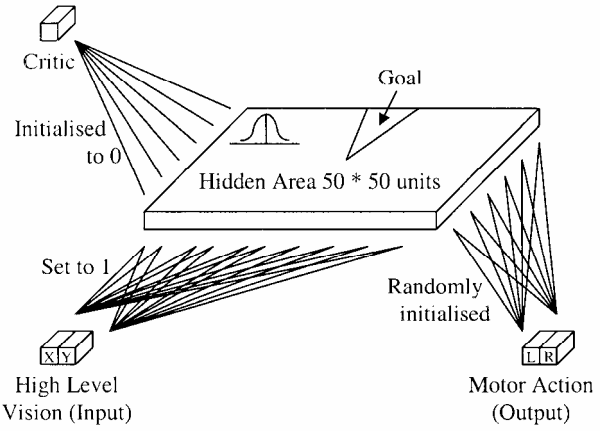


Fig. 1. Architecture used in the initial work.

Fig. 1 illustrates the architecture used for the network allowing the robot to move to a predefined landmark. There are two input units, which are the normalised (x,y) coordinates of the landmark in the omnidirectional image. The trained network then produces the required action to ensure the landmark is ahead of the robot, which then moves forward towards it.

The network was trained using the actor critic reinforcement learning algorithm [6]. Equation (1) is used to calculate the activation of the hidden units to find which unit encodes the position of the landmark in the omnidirectional image. The activation of the hidden units are defined as:

$$f_i(p) = \exp\left(-\frac{\|p - s_i\|^2}{2\sigma^2}\right) \quad (1)$$

Each node in the hidden layer is covered by a Gaussian hill of activation, which is described by (1). $p$ is the

perceived position of the landmark in the omnidirectional image, $s_i$ represents the coordinates of the centres of each of the Gaussians and $\sigma$ is the standard deviation of the Gaussian. The activation of the critic is calculated using (2) and is a weighted sum of the activations of the place cells.

$$C(p) = \sum_i w_i f_i(p) \tag{2}$$

Equation (3) calculates the predicted errors of the possible actions taken by the robot. The calculated value is then used during the training of the critic. $R_t$ (the reward at time t) only equals 1 when the robot is at the goal. As there will be no next time step at this point $C(p_{t+1})$ is 0. At all other times $R_t$ equals 0; thus they are never included in the calculation at the same time. $\gamma$ is the constant discounting factor. $p_{t+1}$ is calculated by predicting the change in position cause by the action to be performed. This varies from action to action.

$$\delta_t = R_t + \gamma C(p_{t+1}) - C(p_t) \tag{3}$$

Equation (4) is used in the training of the critic and this uses the predicted error calculated in (3).

$$\Delta w_i \propto \delta_t f_i(p_t) \tag{4}$$

To calculate the activation of the actor nodes (5) is used. The activation is a weighted sum of the activations of the surrounding hidden units to the current location.

$$a_j(p) = \sum_i z_{ji} f_i(p) \tag{5}$$

During the training of the network, exploration of the environment was made possible using (6) and a random variable. The probability of each action is calculated and incrementally summed; when the result crosses the generated variable that action is executed.

$$P_j = \frac{\exp(2a_j)}{\sum_k \exp(2a_k)} \tag{6}$$

The actor units are trained using (7) in a modified form of Hebbian learning. Here the weight is only updated if the action is performed. This is achieved by setting $g_j(t)$ to 1 if the action is chosen or to 0 otherwise.

$$\Delta z_{ji} \propto \delta_t f_i(p_t) g_j(t) \tag{7}$$

The results gained from the experimentation undertaken on the robot are shown in table I. Here, the PeopleBot robot was placed in visual range of the landmark and was required to move to it. The robot was required to find the landmark in the omnidirectional image, turn and approach the landmark. Each trial was deemed successful if the robot approached the landmark to a range of 10cm or less. An unsuccessful trial occurred if the robot didn't approach to the required range, or it did not find the landmark. Eight starting positions were tested during the experiments, each starting position was given ten trials and the success of each trial was recorded. These trials took between 20 and 60 seconds to complete. This time depended on the amount the robot was required to turn to position the landmark ahead of itself. Also, as the test was conducted on an actual robot there were some external factors affecting the time, such as the motors occasionally stalled as the robot was turning. For training statistics of the network see [4].

TABLE I
RESULTS FROM ROBOT TRIALS

| Landmark Position | % Success |
|---|---|
| Rear Left | 80 |
| Front | 70 |
| Right | 70 |
| Rear Right | 70 |
| Left | 70 |
| Front Left | 30 |
| Front Right | 30 |
| Rear | 30 |
| Overall | 56.25 |

The experiments provided positive results for the system, illustrating the system is capable fulfilling the scenario. However, a problem occurred when the landmark was initially in one of the following three positions: *(i)* to the front left of the robot, *(ii)* to the front right of the robot, and *(iii)* to the rear of the robot. Fig. 2 illustrates the cause of this problem, which is the design of the omnidirectional camera. There are three blind spots present, where the supporting pillars for the conical mirror are. When the landmark was initially behind one of these pillars the system often failed to find them and produced unsuccessful trails. However, the results produced when the landmark was in view from the start had a success rate of 72%. In the remaining trials the landmark may have become occluded by the supporting pillars and thus lost from sight resulting in the unsuccessful trials. This problem may have been solved by further training on the robot.

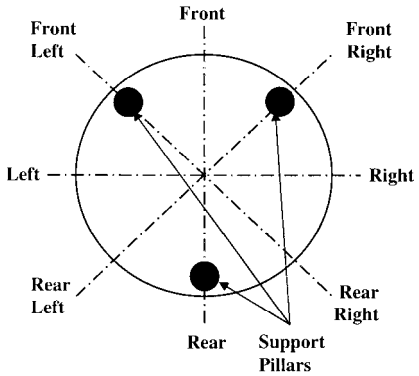


Fig. 2. Top down view of the omnidirectional camera with the support pillars and relative positions of the landmarks are shown.

This system successfully allowed a PeopleBot robot to move to the predefined landmark using reinforcement leaning. However, it is not possible to transfer the trained control system to a different robotic platform. The experience gained during the development of this system has led to the architecture proposed in the following Section.

## IV. Overview of the Architecture

### A. Motivation for Architecture

It is well known that developing control software for robots is a very time consuming and difficult process when the system is hard-coded [10]. Hard-coding here means the robot is manually programmed with required actions given the state of the environment. To design a robust system all possible occurrences need to be considered and coded where appropriate. An alternative to hard-coding the control is to learn it over time, from experience via interaction with the environment. Wolpert et al discuss motor learning and the different approaches that can be used (supervised, unsupervised and reinforcement learning) to learn the motor control from a Neurologists perspective [11]. The learning approach adopted by this paper is reinforcement learning as discussed in Section III. This method of training has been successfully used for robot control [1]-[4].

All of the systems discussed have been designed to control one specific platform for either a real or simulated robot. There has been a trend to design the more advanced systems in robotic simulators to avoid problems faced in the 'real world'. It is the aim of the remainder of this paper to discuss the possibility of developing a control architecture trained by the actor critic reinforcement learning rule [12] to control two very different robot architectures, a PeopleBot and Sony Aibo robot. With the use of reinforcement learning the networks can be partially trained in a simulator for general camera and motor control. These partially trained networks can then be used to control the robots. Further training is then carried out on the robots to specialize the networks with the actual movement of the individual robots and cameras.

As already discussed different systems tend to work in different coordinate systems. For example we worked with a robot-oriented coordinate frame [3]. A network was successfully trained to produce this coordinate transform. However, this limits the applicability of this system to other robotic platforms and the coordinate transform would have to be trained on every new platform. The goal of the transform used was to get the position of the target relative to the robot. This information can be gained directly from the position of the camera if the target is in the centre of the camera image. By using the pan and tilt angles of the camera to get the relative position of the target, in principle a system can be developed that is platform-independent.

### B. Proposed Architecture

The core of the architecture will be the two control networks; one to control the pan and tilt of the camera and a second to control the motors. Two networks will be used to avoid the curse of dimensionality. Using one network would require a high dimensional hidden layer to model the state space and different input / output modalities combined. The input would be a combination of the location of the landmark and the camera alignment. The output of the network would have to combine camera/motor action. We avoid this complexity with the modular use of the two networks.

The visual recognition of the landmark will be decoupled from the core control allowing flexibility of the landmark to be approached. Fig. 3 highlights the proposed architecture.

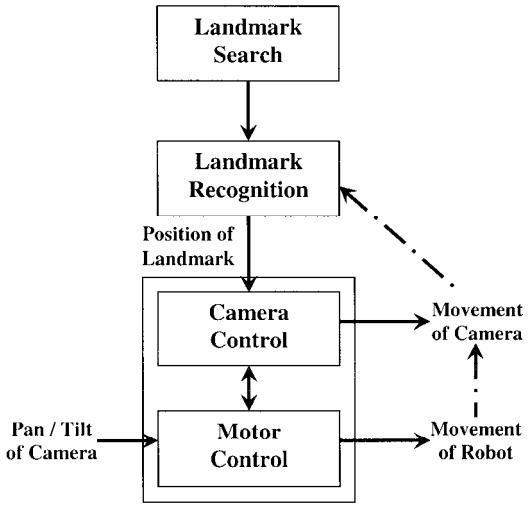


Fig. 3. Overview of the architecture. The core components are the 2 control units, which enable platform independency. The Landmark search / recognition may need altering slightly for each platform.

In the testing scenario initially there will be a direct line of sight from the robot to the landmark. However, the landmark may not be in the visual field of the robot's camera, requiring a search mechanism to be present in the overall architecture. This unit is to move the camera and robot until the landmark is present in the camera image. As this architecture is being developed to allow a platform-independent control system, the landmark detection unit will be kept simple; it is not in the scope of this paper to test advanced object recognition systems, but the portability of the proposed architectures between robotic platforms. Hence the landmark will be a unique object in the environment and will not be occluded by other objects.

Once the landmark is detected, control will be passed to the two control units to move the robot to the required location. To allow the camera control unit to track the landmark as the robot moves it will need updating with the position of the landmark in the camera image. This will be achieved via the landmark recognition unit. The function of this unit is to recognise the landmark in the camera image and get the pixel values of the centre of the landmark to pass

to the camera control unit. As the camera and robot moves, this unit will be invoked to produce the new coordinates of the landmark. With the landmark centred in the camera image the following will occur as the robot moves: *(i)* if the robot rotates to the left, the landmark will move to the right of the image causing the camera to pan to the right to track the landmark, *(ii)* if the robot rotates to the right, the landmark will move to the left of the image causing the camera to pan to the left to track the landmark, *(iii)* if the robot moves forward, the landmark will move to the bottom of the image causing the camera to tilt down to track the landmark and *(iv)* if the robot moves backward, the landmark will move to the top of the image causing the camera to tilt up to track the landmark.

The two control units require coupling, which will allow them to work in unison. There are several options available for the coupling of the units. Two possible couplings are *(i)* to have both control units running concurrently, altering the alignment of the camera and the robots motors simultaneously. The main drawback to this option would be if the camera control unit was not fast enough to keep track of the landmark, it could be lost from sight. This could be addressed during experimentation by reducing the speed of the drive motors, allowing more time for the camera to realign to the new position of the landmark; *(ii)* to only initiate the motor control unit once the landmark was within a threshold distance from the centre of the camera image. However, this approach may lead to a 'stop-start' system, with the robot moving slightly then stopping until the camera realigns itself with the landmark. Therefore, for the remainder of the paper it is assumed that *(i)* will be used for the coupling of the control units.

The two control units will be trained using the actor critic reinforcement learning rule shown in (1)-(7). Each network will be partially trained in a simulator to learn the general control required for the camera and robot movements. These partially trained units can then be exported to the robotic platform to be controlled. As the units are only partially trained, further training will be required on the robot to get the specific control for the individual robots. Having already learned the basic control for the landmark tracking and motor control in the simulator the time required for the training on the robot will be greatly reduced. This training will thus fine-tune the control units to work with the actual movements of the camera and robot.

Fig. 4 illustrates the architecture for the camera control unit. There are two input units one critic unit and four output units. As cameras on different robots may have different resolutions the two inputs will be the normalised x y coordinates of the landmark in the image. As the landmark will cover many pixels the centre point of the landmark will be used as input to the camera control unit. The hidden area will in effect cover the image, and the node that contains the centre point of the landmark will be the node to produce the required camera action to move the landmark closer to the centre of the image. The hidden area is covered by Gaussian hills to find the node that encodes the landmark position. Here the coordinates are fed into the hidden layer and the node which produces the highest firing rate is the node encoding the landmark position.

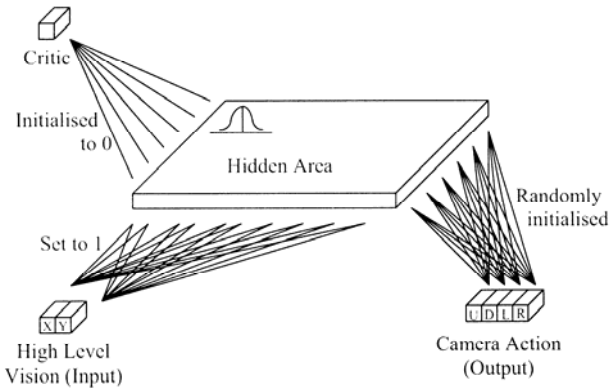


Fig. 4. Architecture to be used for the camera control unit.

To provide an example we will set the dimensions to 32nodes on the x-axis and 24nodes on the y-axis. With these dimensions, an image with a resolution of 640*480 can be easily normalised as each node would represent a grid of 20*20 pixels in the image. Fig. 5 provides an example of an image from a camera with the hidden layer superimposed to show which node would encode the position of the landmark.

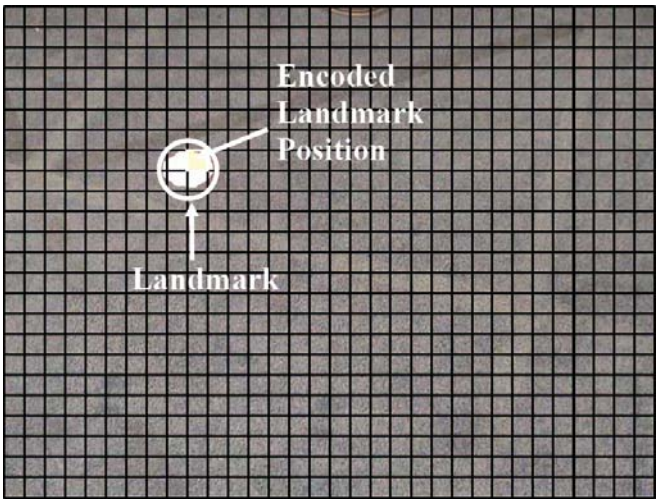


Fig. 5. Example camera image with a landmark highlighted and the node that encodes the landmark position is illustrated.

In fig. 5 the centre of the landmark is at (190, 152). This results in the node located at (10, 8) to encode the position of the landmark. It is this node which would produce the camera action required to move the landmark to the centre of the image. The output from the network is the direction to move the camera and is one of the following four actions; *(i)* tilt upward, *(ii)* tilt downward, *(iii)* pan left, or *(iv)* pan right.

Once the camera has been moved, the new coordinates of the landmark would be calculated and fed into the camera control unit, say (11, 8). This will be repeated until the landmark is positioned in the centre of the image. This unit can be tested by moving the object through the camera image and seeing if the camera is able to track the object. It could work independently from the motor control unit to track an object.

Unlike the camera control unit, the motor control unit cannot work in isolation. This needs the camera control unit to track the landmark as the robot moves through the environment towards the landmark. The architecture of the motor control unit is very similar to that of the camera and is shown in fig. 6.

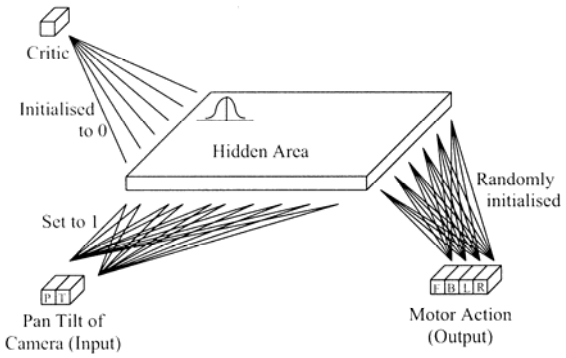


Fig. 6. Architecture to be used for the motor control unit.

The architecture has two input nodes, one critic node and four output nodes. The hidden area encodes the input to the network of the pan and tilt angles of the camera. The network produces the required action to get the landmark to the base of the robot. The actions available are: *(i)* move forward, *(ii)* move backward, *(iii)* rotate to the left, and *(iv)* rotate to the right. The dimension of the hidden layer should be defined in such a way that the different pan/tilting capabilities of different robots can be represented. These angles will be normalised to feed into the network to produce the required motor action to move the robot towards the goal location.

Fig. 7 provides an example of the camera position at two different stages in the process of approaching the landmark. Fig. 7 (a) shows the camera at a position indicating that the landmark is to the front right of the robot. The motor control unit then needs to move the robot in such a way to get the landmark at its base and results in the camera position shown in Fig. 7 (b).

The goal of the motor control unit is to control the robots movement so the camera control unit tracks the landmark and produces the camera alignment shown in fig. 7 (b). With the camera 'looking' down the landmark is at the base of the robot. The motor control unit needs to learn the effect that robot movement has on the relative position of the landmark and thus the movement of camera alignment. This should be learned in such a way that the robot can move to manipulate the alignment of the camera so that it is aligned as illustrated in fig. 7 (b). In the example provided in fig. 7, the robot would need to rotate to the right and move forward to produce the transition from fig. 7 (a) to (b).

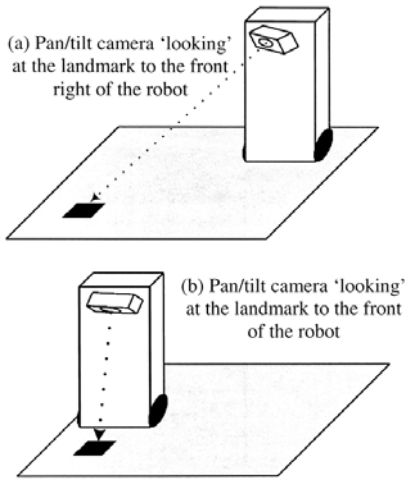


Fig. 7. Example of different camera alignments (a) position of the camera with the landmark located to the front right of the robot and (b) position of the camera when the robot is at the landmark.

### *C. Test Platforms*

As the architecture is designed to allow platform independency it is vital it is tested on different robot platforms. The first robot that is used for testing id a PeopleBot robot shown in fig. 8. This is a wheel-based robot and was the platform used in our earlier system [4]; this allows a direct comparison with the performance proposed system. A direct comparison will be available on the performance of the proposed architecture and that of [4]. The second platform to be used is a Sony Aibo dog shown in fig. 9. These different robots were chosen since they belong to very different platforms. If the architecture is successful in controlling both robots it will provide some validation that the new architecture supports platform-independence.

Fig. 8. PeopleBot robot

The PeopleBot is a relatively large wheel based robot with a width of 47cm, length of 50cm and a height of 124cm [13]. The robot is supplied with a Canon VC-C4 Camera which is roughly 100cm above floor level. The resolution of the camera is 460*350. It is capable of panning $100^O$ to the left and right, and can tilt $30^O$ up and $90^O$ down [14].

The Sony Aibo is a 4 legged robot and is much smaller than the PeopleBot. It has a width of 152mm, a height of 281mm and length of 250mm [15]. The camera is a 100 000 pixel CMOS sensor positioned in the nose of the robot and produces an image of 172*143. As the camera is located in the nose it is pan/tilted by movement of the robot's head. It is capable of tilting $20^O$ up, tilting $65^O$ down, and panning $90^O$ to the left and right (these angles are approximate values).

Fig. 9. Sony Aibo robotic dog

Even though the two test platforms are completely different, the architecture makes use of the similarities that are present. This is due to the availability of a pan/tilt camera in both robots, because although they are situated at totally different heights, they can still be used in the same way. Both robots will be at the landmark when the cameras have a 0 pan/tilt angle and the camera is pointing downward with the landmark in the centre of the camera images highlighted in fig. 10 (b) and (c). The need for a coordinate transform mechanism has been avoided by the use of this principle and enabled the platform independency.

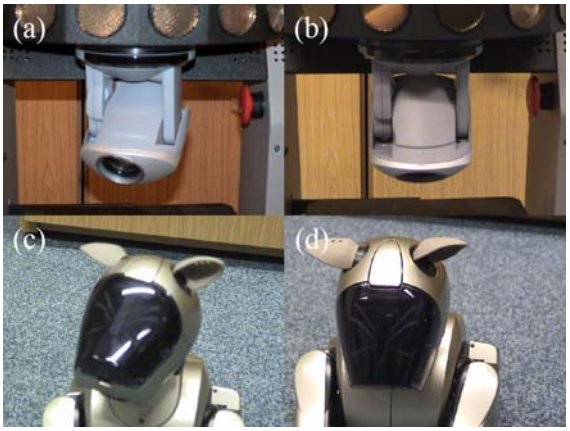


Fig. 10. Example of the camera alignments for the two test robots (this is the same as the example provided in fig. 7).

Fig. 7 provided an example of general camera alignments when the landmark was detected to the front right of the robot and when it was at the base of the robot. Possible camera alignments on the test platforms are shown in fig. 9 for these situations. Fig. 10 (a) and (c) shows the camera alignment of the PeopleBot and the Sony Aibo with a landmark located to the front right. Fig. 10 (b) and (d) show the camera alignments when the landmark is located at the base of each of the robots. From fig. 10 it can be seen that the two robots have very similar camera alignments in these situations, and it is this similarity that will allow the platform-independent control. Although the cameras work on different scales, the actions required by both robots to move to the landmark would be very similar.

## V. Discussion

The proposed architecture tackles several aspects of robot control. Traditionally robot control has been hard coded, and is highly platform dependant. Not only is hard coding the control mechanism time consuming, it is also not flexible. The architecture presented will be capable of learning the robot control, allowing a robot to move to a specified location in an environment, marked by a unique predefined landmark.

The architecture designed will be trained via reinforcement learning, but the traditional problems associated with reinforcement learning have been avoided. The first of these is the 'curse of dimensionality'. This has been avoided by the use of two modular networks to control the camera motors and the drive motors of the robots. To control both sets of motors from one network would require a four-dimensional hidden layer to model the state space. Also more output units would be required to control all motors from the one network. However, by using the two networks the need for the high dimensional hidden layer has been removed. Each of the control networks only need a two-dimensional hidden layer.

The networks are trained via the actor critic reinforcement learning rule which can have lengthy training times. This can cause problems for training on-line due to the physical limitations of robots (battery life etc.). The two control units will be initially trained in a simulator to circumvent this problem. Once a rough control mechanism is learned the network can then be exported to the test platforms for the final training. As the rough control is already learned, the random exploration of the environment required to initially locate the rewards will be removed. It is expected that the time needed for the final training will be greatly reduced as it will be the fine tuning of the control of the different platforms, which will be learned not the entire control mechanism. It is in applying this theory of initial training and then fine tuning on the different robots that will allow the platform-independent control. This is made possible by the use of reinforcement learning, producing a general reward structure during the initial training, then adapting this during the fine tuning on the different robots.

Although the control system is platform-independent, interfaces will need to be developed for allowing interaction between the control units and the robot. The interfaces that are required are for the capturing of images from the cameras and the physical movements of the robot. The action to be performed by the robot will come from the platform independent control system and the actual movement of the robot will come from the interface. Also some form of normalisation will be needed for the camera dimensions and pan/tilt angles as these vary from robot to robot. This could be a simple allocation of a set number of pixels/range of angles to a hidden unit. An example of this would be if 20 hidden units were to encode $200^{O}$ of panning on the camera. Here each unit would encode $10^{O}$ of the panning. The same principle can be used for the tilting of the camera and for the allocation of pixels to hidden units.

The motivation for the proposed architecture came from the development of the system presented in [4]. Although the system was successful in allowing a PeopleBot robot to achieve the defined scenario, limitations were present. The system was highly dependant on the custom made omnidirectional camera which would hamper replication of the system.

## VI. Conclusion

This paper has presented a method for developing a platform-independent control system for mobile robots using reinforcement learning. The scenario to test the proposed architecture has been developed allowing the different robots to be tested in that same scenario. Here the robots are required to move from their starting position to a goal location, which is defined by a set landmark. The scenario has been designed in such a way that it will be extensible once the robots are able to successfully move to the goal location.

Initially the landmark will be a single stationary predefined object which is unique in the environment. With an image processing module, separate from the robot control units, a library of landmarks could be built to allow the robots to perform topological navigation. An alternative extension could be to have a moving object as opposed to a stationary landmark. The camera could track the object as the robot moves, this would thus allow a robot to track and follow a moving object.

## References


[1] R. Hafner and M. Riedmiller, "Reinforcement learning on an omnidirectional mobile robot" *IEEE/RSJ International Conference on Intelligent Robots and Systems for Human Security, Health, and Prosperity,* 2003

[2] C. Weber, S. Wermter, and A. Zochios, "Robot docking with neural vision and reinforcement" *Knowledge Based Systems*, Vol. 12, No. 2-4, pp. 165-72, 2004.

[3] C. Weber, D. Muse, M. Elshaw, and S. Wermter, "A camera-direction dependent visual-motor coordinate transformation for a visually guided neural robot" *Applications and Innovations in Intelligent Systems XIII - International Conference on Innovative Techniques and Applications of Artificial Intelligence,* pp. 151-164, 2005

[4] D. Muse, C. Weber, and S. Wermter, "Robot docking based on omnidirectional vision and reinforcement learning" *Research and Development in Intelligent Systems XXII - International Conference on Innovative Techniques and Applications of Artificial Intelligence,* pp. 23-36, 2005

[5] R.S. Sutton and A.G. Barto, "Reinforcement learning an introduction" *MIT Press*, 1998

[6] D. Busquets, R. Lopez de Mantaras, C. Sierra, and T.G. Ditterich "Reinforcement learning for landmark-based robot navigation" In *Proceedings of The International Conference on Autonomous Agents and Multiagent Systems,* 2002

[7] C. Sierra, R. Lopez de Mantaras, and D. Busquets, "Multiagent bidding mechanisms for robot qualitative navigation" *Lecture Notes in Computer Science,* Vol. 1986, pp. 198-205, 2002

[8] C. Gaskett, L. Fletcher, and A. Zelinsky, "Reinforcement learning for visual servoing of a mobile robot" *In Proceedings of the Australian Conference on Robotics and Automation,* 2000

[9] O. Trullier, S.I. Wiener, A. Berthoz, and J-A Myer "Biologically based artificial navigation systems: review and prospects" *Progress in Neurobiology,* Vol. 51, pp. 483-544, 1997

[10] W.D. Smart and L.P. Kaelbling, "Reinforcement learning for robot control" *In Proceedings of the SPIE: Mobile Robots XVI,* Vol. 4573, pp. 92-103, 2001

[11] D.M. Wolpert, Z. Ghahramani, and J.R. Flanagan, "Perspectives and problems in motor learning" *TRENDS in Cognitive Sciences* Vol. 5, No. 11, pp. 487-494, 2001

[12] D.J. Foster, R.G.N. Morris, and P. Dayan, "A model of hippocampally dependent navigation, using the temporal learning rule" *Hippocampus,* Vol. 10, pp. 1-16, 2000

[13] ActivMedia Robotics "Performance PeopleBot™ operations manual" pp. 65

[14] ActivMedia Robotics "PTZ Robotic Cameras" pp. 13

[15] Sony EntertainmentRobot Aibo operating instructions ERS-210 pp. 63