Vision-Based Landing of a Simulated Unmanned Aerial Vehicle with Fast Reinforcement Learning

Marwan Shaker, Mark N.R. Smith, Shigang Yue and Tom Duckett
Lincoln University, Lincoln, UK, LN6 7TS.
{mshaker,mnsmith,syue,tduckett}@lincoln.ac.uk

Abstract

Landing is one of the difficult challenges for an unmanned aerial vehicle (UAV). In this paper, we propose a vision-based landing approach for an autonomous UAV using reinforcement learning (RL). The autonomous UAV learns the landing skill from scratch by interacting with the environment. The reinforcement learning algorithm explored and extended in this study is Least-Squares Policy Iteration (LSPI) to gain a fast learning process and a smooth landing trajectory. The proposed approach has been tested with a simulated quadrocopter in an extended version of the USARSim (Unified System for Automation and Robot Simulation) environment. Results showed that LSPI learned the landing skill very quickly, requiring less than 142 trials.

1 INTRODUCTION

It is well known that landing is one of the most problematic stages for both manned and unmanned airplanes [9]. Landing an airplane is a complex task because it requires a large amount of kinetic and potential energy of the airplane in the presence of various dynamic constraints, such as sudden changes in winds, carried weight, height and velocity at each landing [9]. In unmanned air vehicles (UAVs), a significant number of accidents happens during the landing phase due to inexperience of pilots or sudden changes in the weather dynamics, such as winds. Thus, automatic landing systems are required to land UAVs safely.

In autonomous control of systems, the system to be controlled requires an accurate model of the environment and the agent in order to create controllers with optimal performance. However, obtaining an accurate model of the environment and UAV is difficult, as the dynamics and aerodynamics of the system are non-linear and complex. Furthermore, if for some reason the modelled system changes, e.g. due to heavy air turbulence in the case of aircraft, the model will no longer represent the actual aircraft. An alternative to enable UAV agents to learn and adapt their behavior is required. One of the most common and general frameworks for this type of learning and adaptation is reinforcement learning (RL). Reinforcement learning enables an agent to learn from scratch by interacting with the environment [19]. We will use a RL algorithm as the learning algorithm for the UAV to learn landing skill.

To land an UAV on the target area, a solution based on visual servoing [15] is applied, where the camera is used to keep track of the target while the UAV is steered to the desired configuration. An image-based visual servoing method is used, where the control law is computed directly from visual features, without explicit pose estimation. In our case, we used image processing techniques to extract features that represent the current state of the system, then we applied a reinforcement learning algorithm known as Least-Squares Policy Iteration (LSPI) [12] to obtain the desired control law. Least-Squares Policy Iteration is designed to solve control problems [12, 13], and uses value function approximation to cope with large state spaces and batch processing for efficient use of the training data. In addition, LSPI converges faster with fewer samples than Q-learning and no initial tuning of parameters is required [21, 12]. In this paper, we extend LSPI so that it will work on a continuous state-action space.

Prior to investigations with real robots, it is very useful to implement any algorithm in simulation. The benefits of simulations include the fact that simulations are easier to setup, less expensive, faster and more convenient to use. Some learning algorithms, such as genetic algorithms, are computationally expensive, and it would take a very long time to compute the learning on a real robot. Simulation provides the facility of transferring a learned controller from simulation and then applying it on a real robot. In our experiments, we used an extended version of the USARSim (Unified System for Automation and Robot Simulation) environment as the testing environment.

The rest of the paper is organized as follows. After discussing related work in Section 2, we explain the simulation setup and the visual servoing process in Section 3. Then, we describe the methodology of the implemented approach for the learning process in Section 4. In Section 5 the experimental results are presented, followed by our conclusions in Section 6.

2 Related Work

Many approaches have been introduced to make the landing task safer in unmanned aerial vehicles. Jiang et

al. [10] applied reinforcement learning to altitude control for airplanes, in particular the Boeing 747 with 20 state variables. They used a coefficient-based policy search method combined with genetic algorithms to learn altitude control for airplanes, where reinforcement feedback is the only information used to update the fitness value of each chromosome. Lin [14] introduced an approach for learning control of missiles (much faster and more mission-oriented tasks). They provided a framework to control bank-to-turn missiles, which combined a fuzzy basis function network and an adaptive critic. Saripalli et al. [16] designed and implemented a hierarchical behavior-based landing algorithm for an autonomous helicopter. They used an AVATAR helicopter to navigate from an initial position to a final position in a partially known environment based on GPS and vision. Valasek et al. [20] developed an adaptive reinforcement learning control methodology for the morphing air vehicle control problem. A structured adaptive model inversion was used as the controller for tracking trajectories and handling time-varying properties, parametric uncertainties, un-modeled dynamics, and disturbances. In addition, a reinforcement learning module using Q-learning was used to learn how to produce the optimal shape at every flight condition. Barber et al. [4] proposed vision-based landing for small fixed-wing UAVs, where a visual marker is used to generate the roll and pitch commands to the flight controller. Bourquardez and Chaumette [5] introduced a visual servoing algorithm to control an airplane during landing. Visual features are used to build the landing control providing a linearized model of the airplane dynamics. The landing manoeuvre is divided into three phases for the purpose of simplification. Huh and Shim [9] introduced an automatic landing algorithm for a blended wing body shaped fixed-wing UAV based on a vision system.

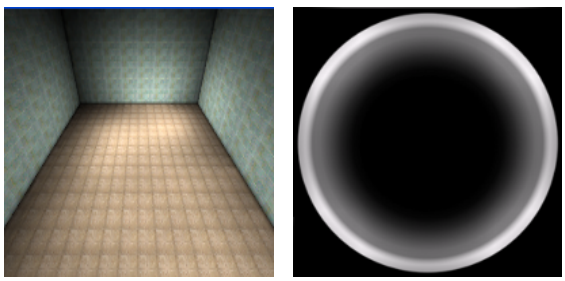
Our approach provides a learning controller that adapts its behavior through direct interaction with the environment. The proposed approach converges faster than Q-learning with fewer samples [21, 17, 12].

3 Simulation

3.1 USARSim Simulation

The USARSim (Unified System for Automation and Robot Simulation) system [8] was created to provide a realistic low cost simulation tool for robots in real environments. It is based on the commercial games platform Unreal Tournament [1] which is customized to provide models of various robots and environments. Additional software components are provided to support image and sensor data acquisition and implement robot actions. In our experiments, we used an extended version of the USARSim software known as altURI, which was developed by one of the authors [18].

The altURI software makes it easy to use the USARSim [8] robot simulation environment. Furthermore, the altURI software provides images through the widely used OpenCV vision processing library [6], as well as networked web images in several formats. Many robots can be simultaneously controlled in the simulator environment. The altURI software consist of two components and two support programs. The vision component captures images from graphics engines and makes them available as OpenCV, web images or a Matlab array. This component is loaded into the graphics process and, on request, intercepts calls made by the simulator to display a frame. The vision component supplies images at the same resolution as the USARSim game environment but using parameters can provide smaller or partial images (multiview). The control component of altURI controls an instance of a robot using information specified in a configuration file. This component allows the action commands to be called using calls to a simple programming interface. A support program to load the vision component into a graphics process is provided.



(a) The environment used for learning the UAV landing skill. (b) The landing target, detected by OpenCV.

Figure 1. Simulated environment

The modular approach allows the vision and command components to be substituted for real robot vision and control components, or components that interact with other simulator environments. The system removes the requirement for any direct simulator programming and makes it possible to have a working robot control program in just a few minutes. The environment used in our experiments is shown in Fig. 1(a)

3.2 Visual Servoing

To make the automatic landing possible, the UAV must rely on special sensors, such as vision, GPS or laser. However, a single GPS is not useful because single GPS without a differential correction typically provides position accuracy of at most a few meters from the ground, which makes the last few meters in the landing process uncontrolled. By contrast, a laser sensor gives an accurate distance measure to the ground. However, it consumes too much energy. Thus, we selected a vision sensor to obtain the state variable for the landing stage.

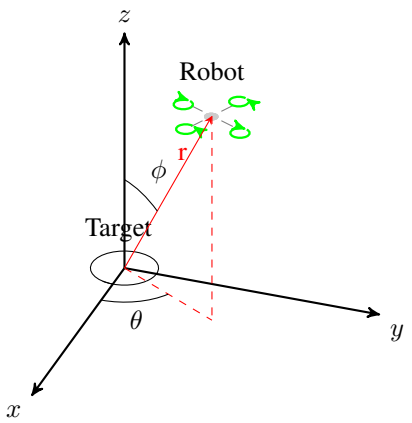


Figure 2. State space representation.

A camera is fixed at the central bottom of the UAV, and provides images of the landing target. In our experiments, the landing target is a full black circle surrounded by circles with grey color range (starting from black and ending with white) as shown in Fig. 1(b). The target was selected because a circle will be detected as a target even the lighting is changed. The state space variables are extracted from the image of the target.

The process of detecting the target is done in three steps using OpenCV. The first step is to capture a frame and implement a Canny edge detection operator. The second step is to detect a number of circles using a Hough transform and calculating their center. The third step is to use the original image to select the circle that has a black center and white circumference.

3.3 Experiments Setup

In our experiments, we used a quadrocopter UAV to test our proposed approach on the landing task as shown in Fiq. 2. The landing station is shown in Fig. 1(b). The state space is represented by three state variables (r, θ, ϕ), where r is radial distance of that agent in the space from the center of the target, θ represents the azimuthal angle in the xy-plane from the x-axis of the target with $0 \leq \theta < 360$, and ϕ represents the polar angle from the z-axis with $-90 \leq \phi \leq 90$. All the state variables are estimated from the captured image. Before starting the experiments, we captured two images with the UAV at a distance of 2 and 4 meters from the target, respectively. We used these images to interpolate the relative radial distance, r, during subsequent processing. The state space is shown in Table 1.

4 Methodology

Many RL methods are time consuming, especially for learning a complex task with a large state-action space from scratch [7, 2]. Nevertheless, many methods have been tried

Table 1. State Space Parameters

Parameter	Parameters Range
r	$[0, 10]$meters
θ	$[0^o, 360^o]$
ϕ	$[-90^o, 90^o]$
Reward	(1) 120 if it gets to the goal, (2) -1500 if it finishes outside state space, (3) equal to a value, this value is decrease as ϕ or θ increase
Goal State	r = $[0, 0.1]$m, θ = $[0^o, 10^o]$, ϕ = $[-5^o, 5^o]$
Control	Lateral Velocity: $[-5, 5]$ Linear Velocity: $[-5, 5]$

to accelerate the reinforcement learning process by combining it with different methods, such as neural networks, planning, etc. However, the high computational complexity or tuning of the initial parameters has limited the potential of such techniques to solve many problems. We propose to address this challenges by using and extending a reinforcement learning algorithm called Least-Squares Policy Iteration (LSPI) [12].

4.1 Least-Squares Policy Iteration

In this work, we applied least-squares policy iteration (LSPI) [12, 13] as the learning algorithm. LSPI converges faster with fewer samples than traditional approaches, since the samples are used more efficiently. This property comes from the fact that LSPI evaluates the policy with a single pass over the set of samples, and all the samples can be used in each iteration to evaluate the policy. LSPI is particularly suited to mobile robot applications because it does not require careful tuning of initial parameters, e.g., learning rate. As it has no learning rate parameters to tune, and does not take gradient steps, there is no risk of overshooting, oscillation or divergence, which are difficulties many other algorithms have to face. This property comes from the fact that the policy is evaluated over a history of samples. Thus, LSPI is insensitive to initial parameter settings. In the next paragraphs, we will explain the theoretical part of LSPI and how it was extended to work on a continuous action space.

LSPI approximates Q-values, Q^π, for a given policy, π, with a parametric function approximation instead of evaluating the optimal state-action values function directly to find the optimal policy. More precisely, the value function is approximated as a linear weighted combination of k basis functions (features) as:

$$Q(s,a) \approx \hat{Q}^\pi(s,a,w) = \sum_{i=1}^{k} \phi_i(s,a) w_i = \Phi(s,a)^T W, \quad (1)$$

where ϕ_i is the i^{th} basis function and w_i is its weight in the linear equation, k is the number of basis functions (fea-

tures). The k basis functions (features) represent information extracted from the state-action pairs, and were designed manually.

With the help of Eq. 1, the TD update equation given in [19] can be re-written as $\Phi W \approx R+\gamma P^{\pi}\Phi W$, where Φ is a $(|S||A|\times k)$ matrix, representing the basis functions for all state-action pairs. This equation can be reformulated [12] as follows: $\Phi^T(\Phi - \gamma P^{\pi}\Phi)w^{\pi} = \Phi^T R$, where P is a stochastic matrix that contains the transition model of the process, and R is a vector that contains the reward values..

The weights W of the linear function $\hat{Q}^{\pi}$ can be extracted by solving the following linear system of equations [12]:

$$W = A^{-1}b, \quad (2)$$

$$A = \Phi^T(\Phi - \gamma P^{\pi}\Phi), \quad (3)$$

$$b = \Phi^T R, \quad (4)$$

but the values of P and R will be unknown or too large to be used in practice. To overcome this problem, LSPI learns A and b by sampling from the environment. A sample is defined as $\{s, a, s', r\}$, where s, a, s', r are the current state, action, next state, and immediate reward respectively. Given a set of samples, $D = \{\{s_i, a_i, s'_i, r_i\}|i = 1, 2, \ldots, L\}$, an approximate form of Φ, $P^{\pi}\Phi$ and R can be constructed as follows:

$$\hat{Q} = \begin{pmatrix} \phi(s_1, a_1)^T \\ \cdots \\ \phi(s_L, a_L)^T \end{pmatrix}, \quad (5)$$

$$P^{\hat{\pi}}\Phi = \begin{pmatrix} \phi(s'_1, \pi(s'_1))^T \\ \cdots \\ \phi(s'_L, \pi(s'_L))^T \end{pmatrix}, \quad (6)$$

$$\hat{R} = \begin{pmatrix} r_1 \\ \cdots \\ r_L \end{pmatrix}. \quad (7)$$

With Φ, $P^{\pi}\Phi$ and R, the optimal weights can be found using Eq. 2. Thus, by combining the policy-search efficiency of the approximate policy iteration with the data efficiency of approximate estimation of the Q-value function, we obtain the Least-Square Policy Iteration (LSPI) algorithm [13]. The aim of LSPI is to learn a policy, π, that maximizes the corresponding Q-function by taking advantage of the efficient search of the approximate policy iteration.

The policy evaluation step of the approximate policy iteration depends on the Q-value function estimation described in Eq. 1. So, whenever a new sample is collected, the weights of the approximation are updated. After the policy evaluation phase has finished processing, the policy improvement starts by selecting a policy that maximizes the approximate representation of the Q-value, as follows:

$$\pi(s|w) = \arg\max_a \phi(s, a)^T w, \quad (8)$$

Lagoudakis and Parr [12, 13] showed that these estimates converge on the optimal weights of the linear function approximation as the number of samples increases.

In traditional LSPI [12, 13], the action space was represented by a fixed number of actions (discrete action space). In this work, we extend the approach to work with a continuous action space (controlling the lateral and linear velocity of the UAV). The agent has to learn the correct direction (right and left for the lateral velocity, or forward and backward for the linear velocity) and optimal value for turning. The optimal value for the lateral and linear velocity is calculated from the optimal w^{π}. After w^{π} is calculated from solving the linear system of Eqs. 2, we calculate $\hat{Q}^{\pi} = \phi \times w^{\pi}$ for each ϕ in Φ. Then, the action value will be equal to the average value of w^{π} that gives maximum $\hat{Q}^{\pi}$.

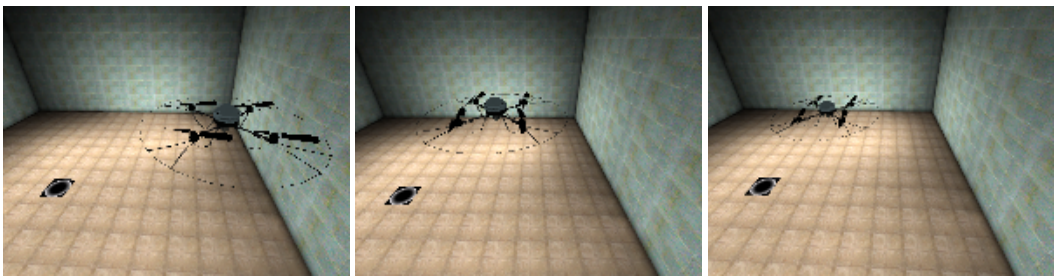
4.2 Computational Complexity

A standardized measurement of the computational time complexity of an algorithm is the number of elementary computer operations it takes to solve a problem, in the worst case. The number of computer operations depends on the "size" of the problem. In the following table, we will give the time complexity of some basic reinforcement learning approaches and the implemented approach, where the equations for calculating the number of elementary computer operations required by value iteration (VI) and policy iteration (PI) is taken from [11].

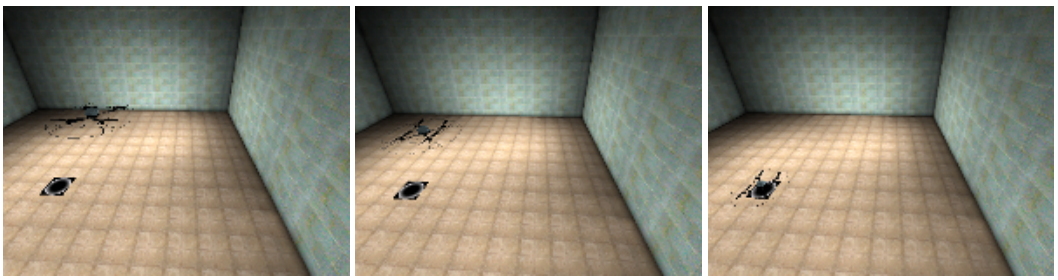
Value iteration and policy iteration approaches depend on the total number of states and actions, which make these approaches inapplicable for large or continuous state-action space. In our approach, the cost of each iteration of the approximate policy iteration in LSPI with continuous state-action space (LSPI-CSA) is:

$$O(NB^3 + NB^2 + (NB^2 * hm) + (NB * hm) + NB + hm) \quad (9)$$

where NB is the number of basis function used and $NB \ll |S| \times |A|$, where $|S|$ and $|A|$ represents the total number of states and actions respectively, and hm represents the number of samples collected. In our experiments, we investigated the number of samples, hm, required to reach the optimal behavior. We tested for $hm = 50$, 100 or unlimited and observed that $hm = 100$ or unlimited does not make a major difference compared to $hm = 50$. Therefore, we used $hm = 50$ in our experiments (meaning that at each step, we collect one sample and 49 samples from the previous learning). Thus, the time complexity of the approximate policy iteration is less than the value iteration and policy iteration approaches, and it is not affected by the number of states.



(a) 1st Sequence (b) 2nd Sequence (c) 3rd Sequence



(d) 4th Sequence (e) 5th Sequence (f) 6th Sequence

Figure 3. Image Sequence of the Landing.

Table 2. UAV Landing Complexity

Complexity			
parameter	VI	PI	LSPI-CSA
No. Of State Calculation	$\frac{\theta_{max}-\theta_{min}}{5} \times$ $\frac{\phi_{max}-\phi_{min}}{5} \times$ $(r_{max} - r_{min})$		θ, ϕ and r
No. Of State	25920		Continuous state
Action	Four actions	Four actions	Continuous action
	(right or left) (Forward or Backward)		
NB			20
Complexity (Number of Computer Operations)	Worst Case: 10,749,542,400	Worst Case: 17,425,008,230,400	Worst Case: for hm= 50 is 29,470 for hm= 100 is 50,520

5 Experiments

In our experiments, we used the quadrocopter UAV model AR100 [3]. Quadrocopters belong to the class of vertical take-off and landing (VTOL) aerial vehicles. A major advantage of this type of aerial vehicle is the ability to launch from the stowing position without the need for complex launching facilities. Furthermore, quadrocopters have a large number of degrees of freedom, and offer highly dynamic flight capabilities.

In our experiments, we implemented the modified LSPI using two different types of basis functions, i.e. polynomial basis functions (PBF) and radial basis functions (RBF) on the UAV landing task. Both approximations gave a smooth landing trajectory. However, to compare the performance of the proposed improvement using RBF and PBF, a number of experiments were done to check the number of iterations required by LSPI to converge to the optimal policy. We performed 100 experiments and the average results of the required number of iterations is shown in Table 3.

Table 3. Comparison of RBF and PBF

Basis Function	Order	Number of Iteration	Average Number of Iterations over 100 experiments
PBF	2	23-231	27
PBF	4	3-9	4.8
RBF	N/A	2-3	2.1

In the case of using PBF, we tested using polynomials of order 2 and 4. For PBF with order 2, LSPI required more than 27 iterations to converge for each state, while for the polynomial of order 4, LSPI required between 3 and 9 iterations to converge for each step. Thus, we used PBF of order 4 in our simulated experiments. In the case of using RBF, we used the Gaussian function. Furthermore, LSPI with PBF requires from 3 to 9 iterations with an average of 4.8 iterations to converge on the optimal policy, while LSPI with RBF requires from 2 to 3 iterations with an average of 2.1 iterations. We used PBF or RBF because we believe that it is a simpler notion and provides a more intuitive explanation than other function approximation schemes.

USARSim software is used for simulating our problem setup. A bounded environment with no physical obstacles was chosen for clarity of results. We used a simple circular shape in the environment to represent the landing station. Using the experimental setup described in Section 3.3, the UAV can control the altitude, lateral, linear and rotational velocities, where altitude velocity controls the up and down movement of the UAV, lateral velocity controls the right and left velocity of the UAV, linear velocity controls the forward and backward movement of the UAV, and rotational veloc-

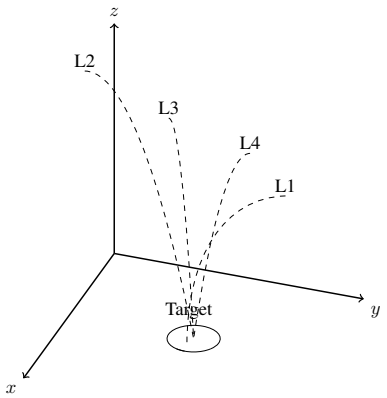


Figure 4. Samples of the learned path

ity controls the rotation movement of the UAV. In our experiments, the UAV learns to control the lateral and linear velocities, while the UAV's altitude velocity was set to decrease at a fixed rate. If the UAV falls below a certain height that causes it to lose track of the target, the RL controller in the UAV is stopped for safety purposes.

The learned path is shown in Fig. 4, where L1, L2, L3, L4 are different starting locations of the UAV. The UAV can land on the target from different starting locations with the learned visual servoing approach. Figure 3 shows a sequence of images displaying the landing process with the learned controller. With the extended version of LSPI, the UAV can learn the landing skill in less than 142 trials.

6 Conclusion

This paper describes the implementation of a visual servoing approach based on reinforcement learning to enable a UAV learn and improve the landing skill. Simulation results showed that, with LSPI as the learning algorithm, the quadrocopter UAV learned the landing skill very quickly, generating a smooth landing trajectory. Future work will include investigating the effects of wind to achieve a more faithful simulation, and transferring the learned skill from simulation to a real UAV.

References

[1] Epic Games, Unreal Tournament. http://www.unrealtournament2004.com, 2004.

[2] P. Abbeel and A. Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 1–8, New York, NY, USA, 2005.

[3] AirRobot. Quadrocopter UAV Model AR100. http://www.airrobot-uk.com/index.htm.

[4] B. Barber, T. McLain, and B. Edwards. Vision-based landing of fixed-wing miniature air vehicles. *Journal of Aerospace computing, Information, and Communication*, 6, March 2009.

[5] O. Bourquardez and F. Chaumette. Visual servoing of an airplane for auto-landing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, (IROS)*, pages 1314–1319, San Diego, CA France, 2007.

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] A. Epshteyn, A. Vogel, and G. DeJong. Active reinforcement learning. In *Proceedings of the 25th International Conference on Machine learning (ICML)*, pages 296–303, New York, NY, USA, 2008.

[8] S. Hughes and M. Lewis. Robotic camera control for remote exploration. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 511–517, New York, NY, USA, 2004.

[9] S. Huh and D. H. Shim. A vision-based automatic landing method for fixed-wing uavs. *J. Intell. Robotics Syst.*, 57(1-4):217–231, 2010.

[10] J. Jiang, H. Gong, J. Liu, H. Xu, and Y. Chen. Altitude control of aircraft using coefficient-based policy method. In *Proc. Canadian Conference on Electrical and Computer Engineering, (CCECE)*, pages 361 –364, 4-7 2008.

[11] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artifical Intelligence Research*, 4:237–285, 1996.

[12] M. G. Lagoudakis and R. Parr. Model-free least squares policy iteration. In *Proc. Advances in Neural Information Processing Systems (NIPS-14)*, 2001.

[13] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:2003, 2003.

[14] C.-K. Lin. Adaptive critic autopilot design of bank-to-turn missiles using fuzzy basis function networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(2):197 – 207, 2005.

[15] T. Martinez-Marin and T. Duckett. Fast reinforcement learning for vision-guided mobile robots. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4170–4175, 2005.

[16] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme. Vision-based autonomous landing of an unmanned aerial vehicle. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2799–2804, 2002.

[17] M. R. Shaker, S. Yue, and T. Duckett. Vision-based reinforcement learning using approximate policy iteration. In *Proc. 14th IEEE International Conference on Advanced Robotics (ICAR)*, 2009.

[18] M. N. R. Smith. altURI-a thin middleware for simulated robot vision applications. http://webpages.lincoln.ac.uk/mnsmith/altURI.htm, 2010.

[19] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[20] J. Valasek, J. Doebbler, M. Tandale, and A. Meade. Improved adaptive reinforcement learning control for morphing unmanned air vehicles. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(4):1014 – 1020, aug. 2008.

[21] P. Wang and T. Wang. Adaptive routing for sensor networks using reinforcement learning. In *CIT '06: Proceedings of the Sixth IEEE International Conference on Computer and Information Technology*, page 219, Washington, DC, USA, 2006.