Reinforcement Learning for Appearance Based Visual Servoing in Robotic Manipulation

UMAR KHAN, LIAQUAT ALI KHAN, S. ZAHID HUSSAIN
Department of Mechatronics Engineering
AIR University
E-9, Islamabad
PAKISTAN

Abstract: - The objective of this paper is to develop a new *appearance* based visual servoing method that needs no prior structuring of the environment and also eliminates the correspondence problem associated with conventional visual servoing methods. Detailed description of object appearance and its generation are provided in this paper. In addition, owing to the non-linear and high dimensional nature of the object appearance a Machine Learning method known as Reinforcement Learning is to generate the controller.

Key-Words: - Visual Servoing, Reinforcement Learning, Machine Learning, Robotic Manipulation

1 Introduction

The integration of visual feedback into robotic manipulation systems is becoming an increasingly important area of research. Although the first systems that used visual feedback to control a robotic system appeared in the early 80's [1], progress was slow. However, the pace of research has increased significantly in the last ten years, due to the development in several areas, especially computer vision and the availability of greater processing power. Therefore, it is now possible to run control loops with visual feedback at reasonable rates (approximately 50 Hz). Earlier on, slower hardware caused larger delays, thus making it impractical to run visual servo control loops in real time.

Two major advantages over conventional sensing mechanisms are offered by the integration of vision into robotic systems. Firstly, vision sensors offer more information, and secondly they are cost effective. Conventional sensors, for instance 3D laser scanners can determine the distances of objects in the workspace. But this information cannot be used for object recognition. The second major advantage offered by vision sensors is that they are cost effective. Until recently, researchers had to manage with specialized expensive pixel processing hardware [2]. However, now with the advancement of microelectronics the vision sensors themselves along with the necessary data processing hardware are available at low prices.

Formally, the above mentioned integration of visual feedback into a robotic system for the accomplishment of a manipulation task is known as visual servoing. The visual servoing method presented in this work, **Appearance Based Visual Servoing** eliminates the need for pose estimation and structuring of the environment as is needed in conventional visual servoing. Instead of estimating the pose of the object to be manipulated, or extracting any features from the image, the "appearance" of the object is used. This is done by using **Angular Colour Co-occurrence Histograms**, first proposed by [3]. This method also eliminates the correspondence problem associated with image based visual servoing. Owing to the nonlinear nature of the object appearance signal, the controller is generated using a machine learning strategy known as **Reinforcement Learning.**

2 Visual Servoing

Formally visual servoing can be defined as "*the use of one or more cameras and a computer vision system to control the position of the robot's end-effectors relative to the work piece as required by the task.*" [2]. Visual servoing systems can be divided into two classes: **Position Based Visual Servoing** and **Image Based Visual Servoing**. Position Based Visual Servoing estimates the pose of the object to be manipulated, relative to the robot end effector. This method has two disadvantages. Firstly, model acquisition and calibration is never error free. Secondly, pose estimation requires a significant amount of computation, thus causing the system to get slower. Image Based Visual Servoing systems on the other hand, do not estimate the pose of the object. The end effector is controlled directly with the help of the features extracted from the image. This eliminates the errors arising from pose estimation, and also reduces the computational requirements.

The approach considered in this work can be classified as an image based visual servoing method. The difference is that instead of extracting features from the image, the "appearance" of the object is used as an input by the controller. The basic idea behind extraction of the appearance of an object is derived from Plenoptic Functions [4].

The key advantages offered by this scheme are that no pose estimation is necessary, and secondly no features have to be extracted. Thus no special color segments on the object, LED's, or structuring of the environment in any other way are required. The object only needs to have some arbitrary texture on it. Thus clearly, this scheme is more robust.

3 The Visual Servoing Task

The manipulator in this work is a KATANA 6M five degree of freedom anthropomorphic arm. The object with respect to which the control loop must function is placed on a turntable in front of the katana. The rotation of the object **object Roll** is not known. The camera is placed in the center of the gripper. The gripper always looks towards the center of the object, at an elevation angle referred to as **gripper Elevation**, and with a rotation referred to as **gripper Roll**. These terms are illustrated in Figure 1.

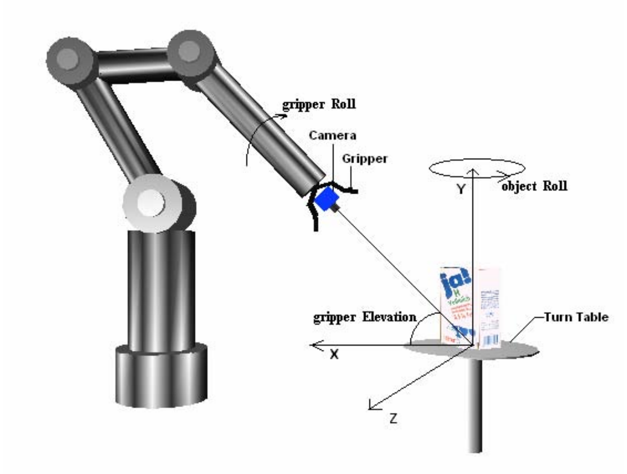

Fig-1. An illustration of the task parameters

As can be seen, the origin of the co-ordinate system is placed at the center of the turn table. The camera is placed in the gripper, and looks towards the origin. The purpose of the controller is to grasp the object that lies in front of the manipulator, using only the images acquired from the camera. This can be achieved by moving the gripper to a pose in which, the gripper Elevation is 90° and the gripper Roll is equal to the unknown object Roll. The strategy proposed in this work is simulated in virtual reality, using the Matlab Virtual Reality Toolbox.

4 Image Processing

The most important aspect of this thesis is the formal definition of the "appearance" of the object. This "appearance" is defined using the angular color co-occurrence histograms (ACCH).
The ACCH's are an extension of color co-occurrence histograms (CCH). The basic idea

underlying CCH's is that they count, starting with a reference pixel that is subsequently shifted across the image, the frequency of pixel pairs in a local environment of the reference pixel. In contrast to CCH's, entries in ACCH's are not only related to color but in addition to the orientations of the considered pixel pairs. Again, starting from the reference pixel, an angle to a pixel in the local environment is computed and assigned to a discrete angular interval.

4.1 Color Co-occurrence Histograms

CCH's encompass a statistical description of the geometric color distribution of an object. A CCH count starting with a reference that is subsequently shifted across the image, the frequency of pixel pairs in a local environment of the reference pixel. The entries of the CCH contain the frequency of pixel pairs of a distinct color combination. The reference pixel together with its local environment is shifted pixel by pixel across a region of interest (ROI) for which the overall geometric color statistics are recorded. The counts in the CCH are normalized, such that the obtained statistics are independent of the scale of the object. One major problem with this approach is that the number of color pairs increases with the square of the number of colors. It is therefore necessary to limit the number of colors used for the generation of the histograms. This is done using the *k*-means color clustering algorithm. The color centers are distributed according to the pixel density in the color space. The k-means clustering algorithm is described in detail in the next section. In addition, the histograms are made robust towards changing illumination by normalizing the color space.

4.2 Angular Color Co-occurrence Histograms

In contrast to CCH's, entries in ACCH's are not only related to color but in addition to the orientations of the considered pixel pairs. ACCH's constitute an extension to so-called normal CCH's as they store additional geometric relations. In addition to the two pixel colours, they also contain information on the angle between both pixels.

Again, starting from the reference pixel an angle to a pixel in the local environment is computed and assigned to a discrete angular interval. Figure 2 shows an object with two distinct colours. The local environment is partitioned into two angular sectors. Therefore, the histogram consists of six separate bins, namely blue-blue, blue-red and red-red pixel pairs, divided into two sectors with angles smaller or larger than 45°. Notice, that the ACCH does not distinguish between blue-red pixel pairs and red-blue pixel pairs.

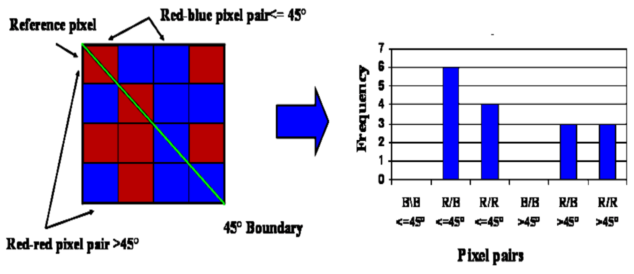

Fig-2. Generation of an ACCH

For the purpose of this work the images are colour segmented using 20 colours. 10 angular bins are used for the generation of the ACCH's. Thus each ACCH has 2100 elements.

5 Reinforcement Learning

This section gives a basic overview to reinforcement learning theory. The first section explains the reinforcement learning problem along with an explanation of the necessary terms. The second section discusses two approaches to solve the reinforcement learning problem namely: Monte Carlo methods and Temporal Difference methods, respectively.

5.1 Reinforcement Learning Problem

Reinforcement learning is a machine learning method used to solve problems involving sequences of actions to achieve a desired objective. There are two classes of machine learning methods: supervised learning methods and unsupervised learning methods. Supervised learning methods use training data to learn a function. The training data consists of a set of inputs and the corresponding set of correct outputs. In the case of unsupervised learning however, no training data is provided. The purpose of the learning algorithm is to learn to classify the input data. Two examples of unsupervised learning tasks are clustering and dimensionality reduction.

Reinforcement Learning lies in the middle of these two extremes. No training data is provided as in the case of supervised learning. Although, a "hint" is provided regarding how good or bad a selected action is. This "hint" is given using a numerical reward signal. The two fundamental elements of a reinforcement learning mechanism are the agent and the environment. The agent is the decision maker that decides which actions to take. The environment is every thing that the agent can perceive using its sensors. Figure 3 depicts the reinforcement learning scheme.

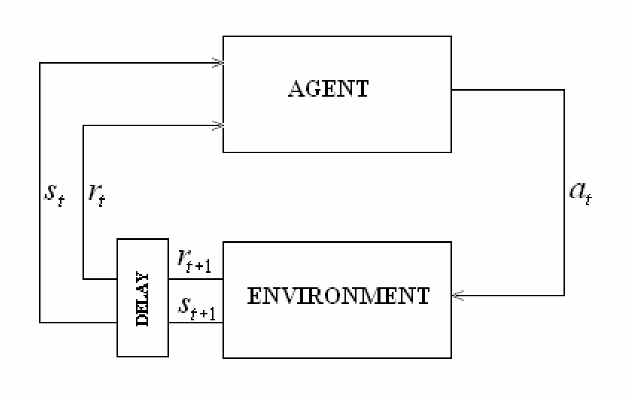

Fig.-3: Basic reinforcement learning scheme.

Formally, let S be the set of all possible discrete environment states, and let A(S) be all the possible discrete actions for every state. The agent maintains a mapping from states to actions. This mapping denoted by $\pi(s,a)$ is known as the policy. $\pi(s,a)$ represents the probability of choosing action *a* in state *s* [5]. At any given time *t*, the agent receives form the environment a state s_t (Figure 3). Based upon the policy $\pi(s,a)$, the agent generates an action a_t. This is received by the environment and at the next time step, it responds with the new state s_{t+1} and the reward r_{t+1}. The objective of the agent is to maximize the received reward. If all actions in all states are executed infinitely often, then the agent always learns the optimal policy. By following the actions recommended by the optimal policy π^* *(s, a)*, the received reward is maximized. All reinforcement learning algorithms must finally be able to generate this policy.

5.2 The Solution Strategies

This section gives an introduction to two methods to solve the reinforcement learning problem, namely Monte Carlo methods and temporal difference methods. None of these methods need a model of the environment. Monte Carlo methods however have the disadvantage that they require complete episodes of agent environment interaction in order to generate the optimal policy. Temporal difference methods on the other hand do not need to wait until the end of the episode and can learn after every time step.

5.2.1 Monte Carlo Methods

The agent learns his policy from scratch. Thus the policy that he follows initially is random. The basic idea underlying Monte Carlo methods is that this arbitrary policy is evaluated and then improved in alternating steps. The process of evaluation and improvement is continued until the optimal policy is generated.

5.2.2 Temporal Difference Learning

The second method used to solve the reinforcement learning problem is the temporal difference method. The major difference between this and the Monte Carlo method is, that it is not necessary to wait until the end of the episode to update the value functions. Rather, the updates can be made every time the agent executes an action and moves into a new state. This is a major improvement over Monte Carlo methods. In most real applications the agent environment interaction can get very long, and thus it is not feasible to wait until the end of the episode. In some applications, there may be no episodes at all.

In order to generate the optimal policy starting from a completely arbitrary one, the arbitrary policy is improved gradually. Unlike the Monte Carlo case however, where the action values $Q^{\pi}(s,a)$ are updated at the end of the episode, updates are made at every step.

The temporal difference method used in this work is known as **Q Learning.** This method makes use of the action value function to generate the optimal policy. Once the agent is in a particular state s_t, it chooses the action a_t. The action selection is done using an ε -greedy strategy described in [5]**.** Afterwards the reward received r_t, and the maximum action value for actions in the next state s_{t+1} is stored. The action value of the state action pair (s_t, a_t), is updated using the following equation:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)],$$

where α denotes the learning rate. This is the rate at which the current action value of the state action pair (s_t, a_t) is moved towards its new updated value $r_t + \gamma \max_a Q(s_{t+1},a)$. This is due to the fact that the action value function for all the state action pairs is initially arbitrary. Thus, the action values of the next state s_{t+1} cannot be "trusted" completely. Therefore the action values are gradually moved towards their estimates. The factor γ determines the rate of propogation of the Q values backwards from the goal state to the starting state.

The Q values for all possible state action pairs are stored in an array Q**.** The number of elements in this array is equal to the number of state action pairs. This array is initialized with random values. In order to guarantee convergence to the optimal policy, all state action pairs must be visited infinitely often. Therefore, the exploration loop must go on forever. At the start of the loop, the agent observes the current state of the environment $s_{current}$. The agent then chooses the action to be executed in this state. The action selection is done in exactly the same way as in the case of the Monte Carlo method. The chosen action referred to as a' is executed and the new state s_{new} and reward *r* are observed. The Q value for action a' in state $s_{current}$ is then updated using the standard Q learning equation. These steps are then repeated forever.

6 Simulation Setup and Results

6.1 Simulation Setup

This section describes in detail the implementation of the reinforcement learning method for the solution of the visual servoing task.

6.1.1 The Agent

The agent consists of a five degree of freedom manipulator, the Katana 6M (Figure 1) from Neuronics AG, along with a CCD camera mounted in the center of the gripper. In order to reduce exploration time the arm always looks towards the center of the object. Only the **gripper Elevation** and **gripper Roll** of the manipulator can change.

6.1.2 The Environment

The environment is an unmodeled textured object, in this case a MilkPack (Figure 1). It is placed on a turn turntable, which is rotated by an unknown rotation **object Roll**.

6.1.3 Action Space

The Katana is capable of looking upon an object from elevation angles of 45° to 90°, and the gripper can be rotated 360°. Therefore the pose of the TCP is defined by two parameters, gripper Elevation and gripper Roll (Figure 1). The action vector therefore has two elements, the change in the elevation ΔElevation of the arm and the change in the rotation of the gripper ΔRoll. The actions are discredited. The ΔRoll can have fixed values of 5°, 0° ,-5° , whereas the ΔElevation can have values of 0° and 5°. This combination generates five possible actions.

Once the reinforcement learning algorithm decides which action to take, the final joint motor commands are generated using the inverse kinematics of the arm. For the purpose of bringing the simulations closer to the real environment, noise is also added to the joints. No noise is added to the acquired images or the histograms currently.

6.1.4 State Space

The state is defined by the ACCH for a particular pose. The image size is 200×200 pixels. It is color segmented with a resolution of 20 colors. Finally it is normalized and then the corresponding ACCH is generated. 10 angular bins are used for the generation of ACCH's.

Since noise is added to the joints of the manipulator, there is some uncertainty in the actual position of gripper. If no noise is added the states will always have fixed values. But if noise is added some randomness is always associated with them. This makes the states continuous.

6.1.5 The Goal State

The goal state is defined by the ACCH which is generated when,

gripper Elevation = 90°
gripper Roll = object Roll

6.1.6 The Reward Function

The reward function is purely image based. Ideally the reward scheme could be such that a positive reward is given upon reaching the **goal** state and nothing otherwise. But the learning can be made faster if negative rewards are given for the **crash** states. Crash states occur, if the gripper goes all the way to the top of the object, however its rotation (gripper Roll) is still not equal to the rotation of the object(object Roll). Therefore three kinds of states are possible: goal state, crash states and **intermediate** states.

6.2 Simulation Results

6.2.1 Q-Learning

The controller is tested for all values of object Roll from 0° to 359°. For every value, the gripper Elevation and gripper Roll are both set to 0°. The controller then moves the gripper to its final position. Ideally, in the final position, the gripper Elevation should be 90° and the gripper Roll should be equal to the object Roll. In this position, the gripper can easily grasp the object.

After testing the controller for all values of object Roll**,** it is observed that when the gripper reaches the top of the object, its rotation gripper Roll comes close to the object Roll, although not perfectly. Figure 4 shows these errors,

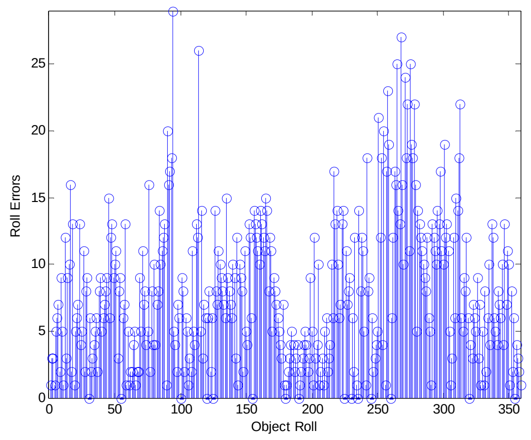

Fig.-4. Roll errors.

The average error is 7.86°. With such errors it is still possible to grasp the object. In order to visualize the learned Q values the object Roll is set to 50°. The gripper Elevation and gripper Roll are then set to all possible values between 0° and 90° at intervals of 5°. At each of these poses, the ACCH's are generated. Then the closest match is searched in amongst the stored ACCH's. The Q values are then stored. For interpolation purposes they are passed through a guassian kernel. Figure 5 below show the learned Q values for action 3 (ΔElevation=5^o,ΔRoll=0^o). The goal state in this case corresponds to a gripper Roll of 50° and gripper Elevation of 90°. The gripper always starts from a gripper Roll of 0° and gripper Elevation of 0°. As can be seen, the highest values are in the region of the goal state and get lower towards the start state. This is because close to the goal state, action 3 nearly always yields a positive reward.

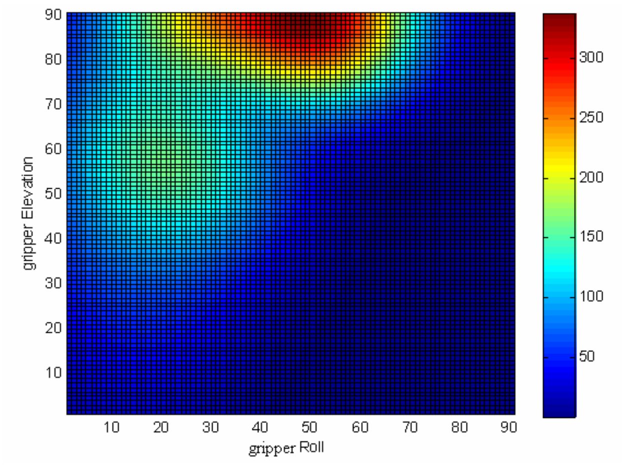

Figure 5: Learned Q values.

6.2.2 Monte Carlo Method

The controller is tested for all values of object Roll from 0° to 359°. For every value, the gripper Elevation and gripper Roll are both set to 0°. The controller then moves the gripper to its final position. Ideally, in the final position, the gripper Elevation should be 90° and the gripper Roll should be equal to the object Roll. In this position, the gripper can easily grasp the object. After testing the controller for all values of object Roll**,** it is observed that when the gripper reaches the top of the object, its rotation gripper Roll comes close to the object Roll, but not perfectly. Figure 6 gives these errors for all object rotations

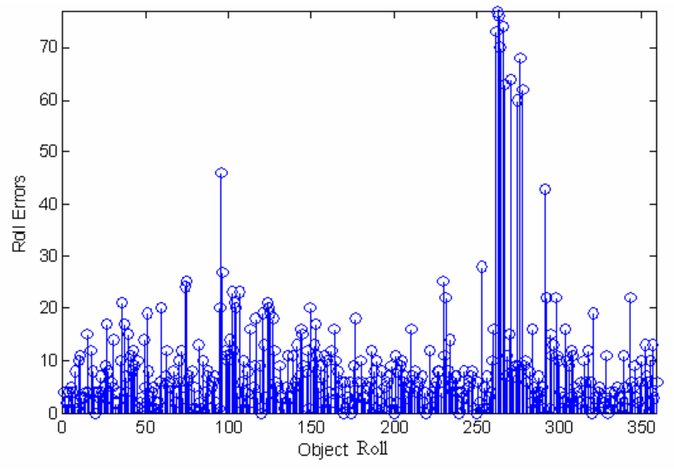

Figure 6: Roll errors.

The average error is 9.25°. For values of object Roll close to 270°, the errors are very high. However for the other object Roll values, they are low. With such errors it is still possible for the gripper to grasp the object.

Figure 7 gives the learned Q values for action 3. The plot is generated in the same way as in section 6.2.1.The goal state in this case again corresponds to a gripper Roll of 50° and gripper Elevation of 90°. The gripper always starts from a gripper Roll of 0° and gripper Elevation of 0°. Just like the values learned with the Q learning algorithm, the highest values are in the region of the goal state and get lower towards the start state.

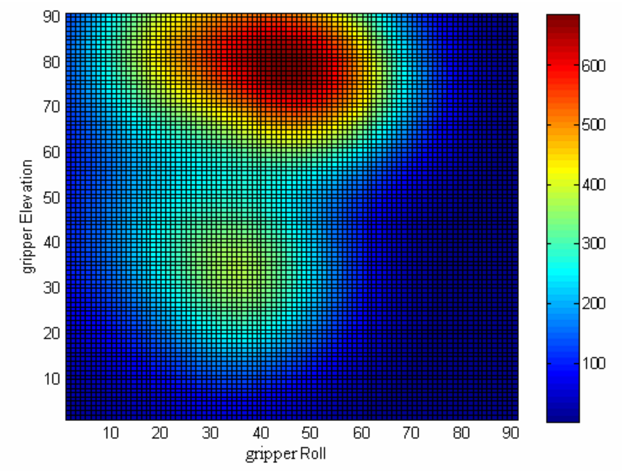

Figure 7: Learned Q values.

7 Conclusion

A new machine learning based visual servoing approach was investigated. The results clearly demonstrate that controller thus generated is adequate for most object grasping application. It remains however to be seen if the large number of trails necessary for the controller generation can be reduced so that the approach can be extended to a real manipulator and the approach be tested in reality.

References:

[1] Sanderson and Weiss, Adaptive Visual Sevo Control of Robots, *Robot Vision*, 1983.

[2] S. Hutchinson, G. Hager and P. Corke, A tutorial on Visual Servo Control, *IEEE Trans. On Automation and Robotics*, 12(5):651-670, October 1996.

[3] F. Hoffmann, S. Ekvall, D. Kragic, Object Recognition and Pose Estimation using Color Coocurrence Histograms and Geometric Modelling, *Image and Vision Computing*, 2005.

[4] Edward H.Adelson and James R.Bergen, The Plenoptic Function and the Elements of Early Vision. *Computational Models of Visual Processing*, Cambridge, MA:MIT Press 1991.

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