Learning State Grounding for Optimal Visual Servo Control of Dynamic Manipulation

Daniel Nikovski DANIELN@CS.CMU.EDU
Illah Nourbakhsh ILLAH@CS.CMU.EDU
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 USA

Abstract

We present an experiment in sequential visual servo control of a dynamic manipulation task with unknown equations of motion and feedback from an uncalibrated camera. Our algorithm constructs a model of a Markov decision process (MDP) by means of grounding states in observed trajectories, and uses the model to find a control policy based on visual input, which maximizes a prespecified optimal control criterion balancing performance and control effort.

1. Introduction

Most robots currently in use have limited sensing capabilities and achieve their objectives by following preprogrammed routines or by employing immediate feedback controllers. Building such controllers usually requires a detailed kinematic and dynamic model of the robot and the task it is performing, as well as significant design effort. In comparison, humans have remarkable natural abilities to acquire skills in highly complex dynamic tasks involving multi-stage decision making guided by visual input, and perform them with minimal energy expense. Much of robotic research has sought to emulate these abilities and endow robots with visual perception, optimal planning over long horizons, and adaptive acquisition of control policies.

Incorporating richer sensing, such as vision, into traditional feedback control schemes, further complicates the design effort, because camera models and state observers have to be built as well. The field of visual servo control has emerged as an alternative, aiming to control the system on the basis of features directly computed from images without state estimation (Hutchinson et al., 1996). Learning visual servo control has been explored too, primarily for the case of immediate feedback (fixed set-point controllers).

Optimizing a criterion involving a balance between performance and control cost over an infinite horizon has been studied in the field of optimal control (Bertsekas, 1995). When the equations of motion of the system are linear and the control criterion is quadratic in both the distance to a goal point and the control cost, linear quadratic regulators have been used with much success. However, nonlinear dynamics and more complicated control criteria usually result in intractable systems of differential equations. Learning the equations of motion further complicates the solution. The field of reinforcement learning is specifically concerned with optimal sequential decision making when the dynamics of the controlled system is unknown (Sutton & Barto, 1998). Balancing an inverted pendulum and the Acrobot task have been studied extensively and also implemented on real setups (Jervis & Fallside, 1992; Spong, 1995). The state of these systems, however, is assumed to be completely known, or (in the case of real robots) precisely measurable from joint encoders.

In the experiment reported in this paper, we have attempted to learn a visual servo controller for a dynamical system with rich dynamics, without prior knowledge of either its kinematics and dynamics, or precise camera calibration and lighting models. The algorithm we present accumulates training data in the form of features, extracted from images of sampled trajectories of the system's behavior. Aggregated discrete states of a Markov decision process are grounded in these features, and the transition and cost structure of the MDP is computed based on the state aggregation. The optimal policy for the resulting MDP is found by means of policy iteration and subsequently used for controlling the system.

The control task and feature extraction process are described in section 2. Section 3 describes the formalism of Markov decision processes and the procedure of grounding states in features, building an MDP model, and using it for finding a good control policy for the underlying dynamical system. Section 4 presents experimental results and discusses the performance of the learned controller with respect to open-loop policies, and section 5 proposes future directions of improvement and concludes.

2. Control problem

The task we are considering is related to the Acrobot and inverted pendulum tasks, as well as to the original centrifugal governor problem of James Watt, which started the theoretical analysis of control systems in the 18^{th} century. A hollow ball of radius $75mm$ is attached to the second link of a simple 2-degrees-of-freedom (DOF) planar robot manipulator with links of length $l_1 = 265mm$ and $l_2 = 115mm$, respectively. The two joints move in approximately parallel planes with link offset of $25mm$ and are actuated by two direct-drive servo motors (attached directly to the links without gear boxes) with torques $44.4oz/in$ and $19oz/in$, and speeds $230ms/60^o$ and $90ms/60^o$, respectively. The cord connecting the attachment point on the second link and the ball is $215mm$ long and has a rigid hollow tube around it, which essentially turns it into a third link attached to the ball and the second link via universal unactuated joints. Taking into account the rotational symmetry of the ball, the system has 6 DOF, only two of which are actuated. The quiescent state of the ball at settings of zero for both joint angles is shown in Fig. 1.

Figure 1. Experimental setup in resting state.

The objective of the controller is to move the manipulator so as to keep the ball as high as possible with minimum control effort. The effort is measured by the amount of movement of the joints in servo-motor notches per control step.

The performance of the controller is inferred from visual input. The manipulator and the ball are observed from a QuickCam Pro camera placed $1380mm$ above the plane of the motion of the attachment point of the cord to the second link, and pointed approximately perpendicular to that plane. The camera and the servo control board are connected to a computer with a Pentium Pro processor running at 200MHz. An image is acquired, processed, and a new control position for the servos is determined every $90 \pm 1.45ms$ – the reason for the $1.45ms$ average deviation is that the operating system we are using can only meet soft real-time constraints. The ball is painted in bright yellow color, and a patch of bright pink color is placed at the second link, centered at the attachment point of the ball cord. By tuning the color gains, exposure time, light sensitivity, brightness, and contrast of the camera, it is possible to separate the ball and the patch from the background by thresholding the resulting pixel intensities without any modifications to the regular office lighting conditions. Separating the ball pixels from the patch pixels is based on color information – the red component is larger than the green component for the patch pixels, and vice versa for the ball pixels. Classification, though, is not perfect: a certain number of ball pixels (usually less than 10%) are always classified as patch pixels, apparently due to the ball's specular reflection, which looks red after filtering by the camera electronics. Nevertheless, the centroids of the detected ball and patch pixels, even when computed with these false positive patch pixels, coincide fairly precisely with the attachment points of the cord to the ball and second link, respectively, and can identify reliably the position of these points on the image plane.

It is the Euclidean distance between these two centroids in a particular image frame that we take as the measure of how high the ball is with respect to its resting state. The exact height is not measurable directly, but is proportional (non-linearly) to the projection of the third link on the image plane, so the latter is a reasonable substitute measure.

If the measured height in an image frame t is H_t, and the sum of the displacement of the servo motors is C_t, the total performance for that frame is $R_t = H_t - \alpha C_t$, where $\alpha \geq 0$ is a user-specified multiplier expressing the relative cost of control. The control objective then is to find a control policy π^* that maximizes the average performance of the system over an infinite number of control points:

$$\pi^* = argmax_\pi \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} H_t - \alpha C_t$$

3. Controller design

Designing a general-purpose controller for this system, which can bring the ball to any state in its state space is an extremely complicated problem: too few of the joints are actuated, the links are not completely rigid and the significant mass of the ball in comparison with the servo torques interacts with the links in ways, which are hard to capture in a dynamical model. In addition, even though the kinematics of the arm are known, the camera is not calibrated and the actual state of the system cannot be inferred; such a model is not even possible, because estimating the state of a ball flying in three-dimensional space from a single camera image is an ill-posed problem. Even if estimating the position was possible, the frame rate (11Hz) is too low to estimate the translational and rotational velocities of the ball by frame differencing. Finally, there are occasional delays in control periods, which, although small and infrequent, can have significant impact on the flight of the ball.

Instead of building a general-purpose controller, our strategy is to force the ball into self-stabilizing trajectories and choose the one, which is optimal with respect to both height and control effort. Forcing the ball to circle steadily is a suitable choice of a set of trajectories: we found that for a certain class of open-loop control schedules, the movement of the ball is self-stabilizing around a central orbit and not too sensitive to small control delays. Such trajectories also keep the projection of the ball quite far from the attachment point on the second link, which results in good performance.

One open-loop control, which always brings up the ball and holds it in a steady circle is an ellipsoidal motion in joint space with amplitude of the first joint at least 20 notches (14^o) and amplitude of the second joint at least 30 notches (21^o) and period less than $800ms$. The resulting shape of motion of the attachment point on the second link in Cartesian space is quite far from circular and is even further distorted by the significant centrifugal forces exerted on the ball and transferred to the link via the connecting cord; nevertheless, the resulting motion of the ball is roughly circular. The problem with this control is that it is expensive in terms of control effort and a lot of energy is expended even when the ball is in high orbit and has enough momentum to go on circling with little or no effort. However, starting with controls with lower amplitude never succeeds in bringing the ball into a steady orbit and instead results in chaotic jerky movements close to the starting state. The problem then becomes how to switch efficiently between a powerful, but expensive control, which almost always succeeds in bringing the ball to a high orbit, and a weak control, which is inexpensive, but affects the height of the ball very differently in different situations.

The two controls we considered were a_1, the ellipsoidal motion with amplitudes 20 and 30 notches for the two joints, which completes a revolution on the ellipse in $720ms$ (8 control frames), and a_0, the motion with zero amplitude, when the joints of the arm remain at their previous positions. Switching between controls is done only after completion of a full revolution, i.e., after 8 control points with the same amplitude. When the ball is on a stable orbit, this switch occurs at approximately the same phase of the orbit.

This is a sequential decision problem, because several actions are required to move the system between different orbits: even when applying a_1 only, it takes at least 4 revolutions (32 control points, or 2.88 seconds) to enter a stable high orbit from the quiescent state of the ball. Conversely, if the ball is on a stable orbit, it takes at least 8 revolutions of no action (a_0) for it to drop down close to its resting state. Furthermore, when the ball has moved down past a certain height of the orbit, subsequent applications of a_1 appear to produce initially only erratic movements, and it takes relatively longer time to bring it back up, in comparison to the case when the ball is starting from the quiescent state. Consequently, the task of the controller is to bring the ball up initially by applying a_1 several times, and once it is high enough, use the momentum of the ball to conserve energy by applying action a_0. However, as soon as the ball drops down to a level, which is likely to result in chaotic behavior, the controller should bring it back up by applying action a_1.

The decision of when to switch the actions could potentially be based on many characteristics of the ball's trajectory, measured from the 8 images taken during one revolution. One possibility is to use the 8 ball locations directly; another one is to use derived statistics such as the average radius of the trajectory, or some estimate of its phase. Clearly, the ball's trajectory during the next revolution depends on all of these variables, but if a compact decision rule is to be found, it is essential to find one or more features to base the decision upon.

Since the performance criterion to be optimized includes the estimated distance H between the center of the ball and the attachment point on the manipulator, we chose to use the average value $\bar{H}$ of H over the last 8 frames as a description of the state of the ball. Clearly, this introduces some perceptual aliasing, because important parameters of the ball's flight are ignored, such as the phase of the trajectory, as well as the location of the individual points along it.

Failure to consider informative aspects of the true state results in decreased determinism of the observed evolution of the state, and higher uncertainty about the effect of actions. In order to deal with this problem, we rely on an algorithm for stochastic optimal control, which samples the system by a consistent exploration policy, builds a stochastic MDP model of the state evolution and state rewards, and then uses policy iteration to find the policy that maximizes the average expected reward.

During exploration, the system attempts to cover the whole state space of the problem (in this case, all possible heights of orbit), and try all actions everywhere. The result of exploration is a database of transitions of the form $(\bar{H}_i, a_i)$, $t = 1, N_o$, which are assumed to be representative of the dependency of $\bar{H}_i$ on $\bar{H}_{i-1}$, and a_i. The performance R_i at time t does not have to be stored explicitly, because it can always be computed as $R_i = \bar{H}_i - \alpha C(a_i)$.

In order to find an optimal policy, we construct an MDP from the database of transitions. An MDP is described by the tuple (S, D, A, P, R), where S is a set of states of size N_s, D is an initial probability distribution over these states, A is a set of actions of size N_s, P is a transition function that maps $S \times A$ into probability distributions over S, and R is a reward function that maps $S \times A$ into a scalar. The expected average performance of following a particular policy $\pi(s)$, when starting in state $s \in S$ can be computed by solving the linear system of equations for all states

$$V(s) = R[s, \pi(s)] - \hat{R} + \sum_{s' \in S} Pr[s'|s, \pi(s)]V(s')$$

where s' ranges over the whole state space, and the scalar $\hat{R}$ is the expected average reward of the policy π (Puterman, 1994). If the number of states in S is finite, the value function for each state can be uniquely determined – even though there is an extra variable $\hat{R}$ to be found, the value function of the states in S is unique up to an additive constant, so by setting $V(s_0)$ to 0 for some arbitrary state s_0, we can obtain a completely determined linear system of equations.

The policy iteration algorithm starts by choosing randomly an initial policy π, estimating its value function as described above, and then improving the policy by choosing for each state s the action, which maximizes its expected performance if started at this state:

$$\pi'(s) = argmax_a[R(s, a) + \sum_{s' \in S} Pr(s'|s, a)V(s')].$$

If $\pi'(s) = \pi(s)$, the algorithm terminates, and the optimal policy $\pi^*(s) = \pi(s)$; otherwise $\pi(s)$ is set to $\pi'(s)$, and the algorithm proceeds with another sequence of policy estimation and improvement steps, until termination. The policy estimation step has relatively high complexity $(O(N_s^3))$ if implemented directly, but for state spaces with less than a couple of thousand states, this is not a problem.

In order to apply this algorithm to the current control problem, however, we need a discrete state space with relatively few states, and good estimates of the transition function of the MDP under all actions. All we have so far is a database of transitions in continuous feature space, as represented by the feature $\bar{H}_i$, which are hopefully representative of the behavior of the system in different modes under different actions.

Our general approach is to ground a small set of discrete states into feature space by means of clustering the starting states in the transition database, and subsequently estimating the transition probabilities between clusters under all actions. We have pursued this approach also for the problem of navigation of mobile robots (Nikovski & Nourbakhsh, 2000). The rationale behind this approach is that a clustering algorithm naturally produces clusters of states aggregated in regions with high occupancy, which results in approximately even distribution among clusters of the samples, from which transition matrices are estimated, and thus in higher precision of the estimates. On the contrary, a uniform discretization of feature space, which is not driven by the actual experienced trajectories, is likely to result in imprecise frequency counts and thus in high variance of the estimates.

Currently, the algorithm uses a fixed predetermined number N_s of clusters. This number depends on the number T of sampled transitions in the database – if we assume a relatively even distribution of starting states among clusters, then an average of T/N_s states are likely to appear to start from each cluster. Since there are a total of N_s clusters, where these transitions can lead to, and N_a actions to try, we can expect that each entry in the transition probability tables would be estimated from an average of $T/(N_a N_s^2)$ samples. The larger this number, the less variance will exist in these estimates.

The performance of each aggregated state (cluster) is taken to be the average of the performance for those transitions in the database, whose starting states belong to this cluster. The initial distribution in this case is deterministic – the system always starts in the cluster, to which the quiescent state of the ball is assigned.

4. Experimental results

We experimented with different exploration policies, trying to find one, which can sample fairly evenly all modes of the system. Choosing randomly actions a_0 and a_1 with equal probabilities is not such a policy – under it, the ball stays mostly in two states: either in high steady orbit, or near the resting state, moving chaotically. The system spends the least time in the most interesting part of state space – the boundary between steady high and steady medium-high orbits, where it is very likely to dwell under optimal energy-conserving policies.

To alleviate this bias of completely random exploration towards unimportant parts of state space, we devised an exploration policy, which persistently probed the interesting regions of state space. The policy was of the form $a_1^4 a_0^k a_1^k a_0^{10}$, $k = 1, 10$, run over six runs to produce a total of 504 state transitions. Its purpose was to bring the ball up to a steady orbit, and then test for how many revolutions it was safe to let the ball move on momentum only, so that the same number of revolutions with a_1 could bring it back up, instead of entering a chaotic low orbit, from which it could not recover without expending too much effort. The last 10 revolutions of a_0 ensured that the ball came to a rest before trying the next deeper probing sequence. The resulting transitions are shown in Fig. 2. It can be seen that while action a_0 fairly consistently reduces the orbit, the consequences of action a_1 have much more uncertainty, although in general it increases the orbit until the height reaches a limit.

We used the transitions experienced during exploration to cluster the starting heights into N_s different discrete states by means of k-means clustering, also known as learning vector quantization (LVQ). The resulting cluster centers (LVQ codebook) were used to label both the starting and successor states of each transition (each successor state is the starting state

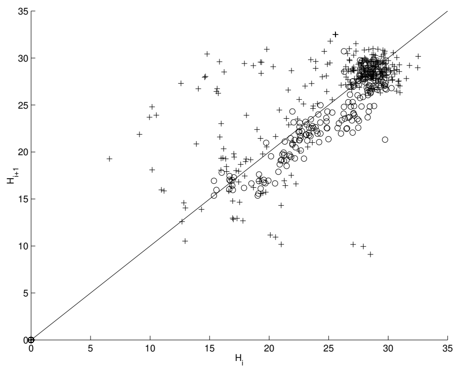

Figure 2. Observed transitions for the exploration policy. Action a_0 is denoted by 'o', and action a_1 is denoted by '+'. Transitions below the solid line reduce the height, while those above the line increase it.

of the next transition in the database). The transition matrices of the MDP were computed by frequency counting: the probability of moving from state s_i to state s_j under action a was computed as the number of transitions starting in s_i and ending in s_j under action a, divided by the total number of transitions out of s_i under a. The aggregated performance for state s under action a was the average height during all transitions starting in s under action a, less the cost of action a multiplied by the factor α, with $C(a_0) = 0$ and $C(a_1) = 5$.

The MDP was used to compute the optimal action for each of the 10 states by means of policy iteration. Not surprisingly, the optimal policy was threshold-based: for all orbits lower than the threshold θ, the optimal action was a_1, while for orbits higher than the threshold, the optimal action was a_0. The threshold θ lies halfway between the centers of the two clusters, where the optimal action changes, and in order to minimize the effect of the random cluster initialization of the LVQ algorithm, five runs were performed and the resulting thresholds averaged. (Note that the number of clusters on the two sides of the threshold is in general different between runs.) The computed threshold was used to control the arm for various values of α. The results are shown in Table 1.

The policy iteration correctly discovers that when there is no cost of control ($\alpha = 0$), the optimal policy is to apply always action a_2. When $\alpha > 0$, the optimal policy maintains a lower orbit, trying to economize effort according to its relative cost. Also, there is fairly good agreement between the predicted performance by the policy iteration algorithm ($\bar{R}$) and the actual performance measured from the test run,

Table 1. Results for three values of the cost multiplier α. For each value, the average height $\bar{H}$ and cost $\bar{C}$ measured over 1000 revolutions are shown, as well as the predicted total performance $\hat{R}$ as reported by the policy iteration algorithm, and the actual total performance $\bar{R} = \bar{H} - \alpha\bar{C}$.

α	θ	$\bar{H}$	$\bar{C}$	$\bar{R}$	$\hat{R}$
0	∞	28.36	5.00	28.36	28.43
1	27.86	26.54	3.05	23.49	24.77
2	27.18	25.13	2.28	20.09	21.44

which indicates that the stochastic model obtained by grounding discrete states into features is good. Another look at these data is shown in Fig. 3, where the total performance for all policies is plotted against the relative cost of control. It can be seen that each policy is optimal only around the value of α that it was computed for, and the learning algorithm takes into consideration that cost when computing a control policy. This, of course, does not necessarily mean that these are the exactly optimal policies and better ones cannot be found by some other algorithm.

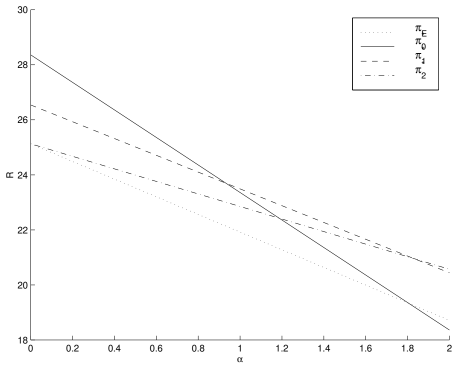

Figure 3. Performance under varying cost multipliers for the exploration policy (π_E) and the computed optimal policies π_0^*, π_1^*, π_1^*, for $\alpha = \{0, 1, 2\}$, respectively.

Fig. 4 shows a sample path of the height of the controlled system under the computed optimal policy for $\alpha = 1$, along with the control signal. It can be seen that the controller does not follow an open loop policy, but performs feedback control based on the measured height of the ball from visual input.

5. Conclusion and future work

The proposed algorithm for state grounding in image features is able to find a suitable real-time control policy based on visual input for a dynamic manipulation task performed by an underactuated manipulator. The algorithm takes into consideration the relative cost of control to adjust the control policy. It is likely, however, that including more features representative of the state of the system might produce even better policies. Another direction for improving the controller is to consider not only grounding of aggregated states into features, but also finding similar aggregations in action space.

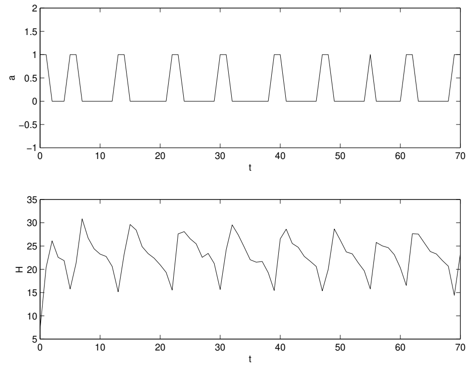

Figure 4. Control signal and measured height of the ball under the computed optimal policy for $\alpha = 1$.

References

Bertsekas, D. P. (1995). *Dynamic programming and optimal control.* Belmont, Massachusetts: Athena Scientific.

Hutchinson, S., Hager, G., & Corke, P. (1996). A tutorial on visual servo control. *IEEE Trans. Robot. Automat.*, *12*, 651–670.

Jervis, T. T., & Fallside, F. (1992). *Pole balancing on a real rig using a reinforcement learning controller* (Technical Report CUED/F-INFENG/TR. 115). Cambridge University, UK.

Nikovski, D., & Nourbakhsh, I. (2000). Learning probabilistic models for decision-theoretic navigation of mobile robots. *Proc. 17th International Conf. on Machine Learning* (pp. 671–678). Stanford: Morgan Kaufmann, San Francisco, CA.

Puterman, M. L. (1994). *Markov decision processes—discrete stochastic dynamic programming.* New York, NY: John Wiley & Sons, Inc.

Spong, M. (1995). The swingup control problem for the acrobot. *IEEE Control Systems Magazine, Feb.*

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning.* Cambridge, Massachusetts: The MIT Press.