Avatars That Learn How To Behave

Adam Szarowicz and **Paolo Remagnino** [1]

Abstract. It is possible to model avatars that learn to simulate object manipulations and other complex actions. A number of applications may benefit from this technique including safety, ergonomics, film animation and many others. Current techniques *control* avatars manually, scripting what they can do by imposing constraints on their physical and cognitive model. In this paper we show how avatars in a controlled environment can learn behaviors as *compositions* of simple actions. The avatar learning process is described in detail for a generic behavior and tested in simple experiments. Local and global metrics are introduced to optimize the selection of a set of actions from the learnt pool. The performance for the learnt tasks is qualitatively compared with a human performance.

1 Introduction

Anthropomorphic avatars are useful in a number of applications, for example to simulate object manipulations, but also interactions between individuals [13][12] [9][5]. Current research techniques involve the modeling of avatars, including their physical and cognitive features [7], and the modeling of the environment they inhabit [8]. Simple to complex animations can then be staged by imposing physical constraints, such as gravity, obstacle avoidance and above all anthropomorphic limitations (for example the limited movements of body limbs).

This paper shows that it is possible to devise automatic learning of behaviors in terms of simpler elemental actions. An avatar can learn how to achieve incrementally more difficult tasks by choosing and executing actions, making mistakes and recovering from its mistakes. The avatar learns to complete tasks by combining simple actions in an optimal order, taking into consideration the internal (physical/biomechanical) and external (environmental) constraints. If an indication of the ideal behavior of solving a task is known, then mistakes can be associated with rewards: the better the performance of the avatar, the larger the rewards. Reinforcement Learning lends itself optimally to this purpose: avatars, or parts of their physical and cognitive being, and a set of possible actions define a state space which can be explored to find the best combination of actions to satisfy a more or less complex goal. The paper is organized as follows: Section 2 introduces the avatar model drawing an analogy from robotics, Section 3 recapitulates on the used Reinforcement Learning techniques, in the light of the avatar task. Section 4 illustrates some examples of learning tasks. Section 5 discusses how an optimal choice of actions can be made when more are available. A number of metrics are introduced to calculate a distance between sequences of actions, and an analogy to information theory is drawn. Section 7 concludes the paper.

[1] Digital Imaging Research Centre, Kingston University, Kingston, UK email: {a.szarowicz,p.remagnino}@kingston.ac.uk

2 The avatar model

The used avatar model borrows its biomechanical characteristics from robotics. An avatar has a set of joints whose movements can be either *prismatic* (movements constrained on a 3D plane) or *revolute* (movements involving a rotation about an axis in 3D space). Then the kinematics of manipulators [4] rules all possible movements of joints as combinations of prismatic and revolute elemental movements. Needless to say a large amount of literature exists which explores a variety of methods to implement an optimal movement of an ensemble of joints to perform simple and complex actions. A standard goal usually includes more or less complex object manipulations. This paper demonstrates that using forward and inverse kinematics for a simple but articulated avatar the optimal sequence of actions can indeed be learnt. The forward kinematics (FK) solves the motion of the end effector as a function of the joint angles, the opposite task - generating the joint angle values knowing the position of the end effector - is known as inverse kinematics (IK). Learning is implemented by creating a suitable state space and applying Reinforcement Learning techniques to learn the optimal movements to reach an object of interest. Figure 1 illustrates the concepts of forward and inverse kinematics and the current model of the avatar.

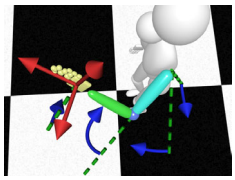

Figure 1. Position of the end effector can easily be calculated when all joint rotations are given (forward kinematics), the opposite task is the problem of inverse kinematics.

3 Reinforcement Learning for autonomous avatars

All avatars presented in this paper are anthropomorphic. They can perform a number of actions. The standard way of adding new actions and behaviors (seen as compositions of actions) to an avatar *repertoire* is to manually script them. Ideally, an avatar should allow for new actions but should also have a form of automatic generation of new actions. The Reinforcement Learning technique lends itself very well to the automatic acquisition of actions, and behaviors.

The implemented avatars use the Q-learning technique. All standard Reinforcement Learning techniques, and Q-learning in particular, do assume a scene evolving along a discrete time line, indicated by the t variable. A suitable state space is defined as well as

all available actions for each defined state. The reader should refer to [14],[10] for a detailed discussion on Reinforcement Learning techniques. The quality of an action is kept up to date either using a table of quality values $Q(s_t, a_i)$ or a neural network [2] or more stable alternatives (for instance [1]).

Results on both deterministic and non-deterministic approaches are described in the next sections. In all the experiments the state-space and the goals of the agent are explicitly defined. The Q-learning was implemented by discretising the space into states and using a Q-table. The following list describes more details of the current implementation:

- An avatar can perform a number of simple actions including arm, forearm and hand motion illustrated in Figure 2 and textually described in Table 1,
- The state-space is different for each mode of control but in both cases it is discretised defining a number of degrees of freedom for the used joints. In the case of forward kinematics the degrees of freedom of the arm were defined as rotations around spatial axes (see first four illustrations of Figure 2), all rotations were discretised and constrained to realistic physical movements. In the case of inverse kinematics the state-space, the discretisation is performed on the 3D space location of the end effector of the avatar, that is its hand (see last illustration of Figure 2).
- In both forward and inverse kinematics, walking along one dimension is considered as an additional action. Descretisation here is implemented along one axis in the two main directions of the ground plane, where the avatar moves. Similarly grabbing an object is considered to be an additional action.
- Other external objects (such as the door shown in the experiments) were represented as additional variables.
- For each possible state-space dimension there are always two possible actions, indicating a movement of a body part (i.e. an arm, a forearm, a hand etc.) along such dimension in the two opposite directions. Examples include the avatar walking forward and backwards or moving its hand along the vertical axis resulting in lowering or raising the hand.
- Successful fulfilment of a goal is rewarded, collisions with the environement and violence of the biomechanical constraints are punished

Table 1. Low-level actions used to train the avatar

Forward Kinematics	Inverse Kinematics
Rotate arm up/down by α	Move palm by x
Rotate arm forward/backward by α	Move palm by y
Rotate forearm by α	Move palm by z
Rotate hand along Z axis by α	
Rotate shoulder along Z by α	
Perform the grabbing action	Perform the grabbing action
Move forward/backward by x	Move forward/backward by x

Although Q-learning convergence is not affected by the initial state, for optimization reasons all animation experiments were biased towards a realistic starting state (for instance with the avatar up-right and both arms aligned with the body).

4 Learning Tasks

Two learning tasks have been defined and executed using Reinforcement Learning. Each task was *trained* using both forward and inverse kinematics control modes.

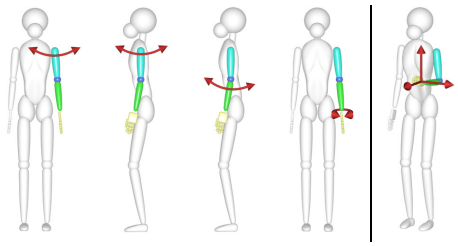

Figure 2. Avatar degress of freedom for the teapot task: FK (first four) and IK (last) control

4.1 The door opening task

For this task the goal of the agent was to get through a locked door. The door would be unlocked upon touching the door handle. The avatar would then have to push the door and pass through it. The agent was rewarded whenever its position was behind the door. The simple actions available to the agent were selected from Table 1, for the FK these were actions 1,2,3,7 (Experiment $\mathcal{A}$) and 1,2,3,4,5,7 (Experiment $\mathcal{B}$), α was set to 20 degrees, step size (x in action 7) was 35 cm, in all experiments $gamma$ = 0.95. Experiment $\mathcal{B}$ differed from Experiment $\mathcal{A}$ in that two additional degrees of freedom for the arm motion were added. Therefore the state space for the first experiment consisted of approximately 12000 states distributed across 4 dimensions (2 degrees of freedom for the left arm, 1 for the left forearm and 1 for backward/forward walk, the sizes of these dimensions were 12, 16, 11 and 6 respectively). Eight simple actions were available to the agent at each time step. These were three rotations - two for the arm and one for the forearm - in two opposite directions and walk along one (2*3+2). The two additional degrees of hand freedom (hand and arm rotation around the z-axis, 13 different positions for each) added for the second experiment made the total number of states of over 2 million and 12 actions per state. In the first experiment the solution was usually found in only about 250 iterations, the second experiment required at least 1500 iterations. The lengths of the shortest solutions in both experiments were 5 simple actions.

The task of a third experiment (Experiment $\mathcal{C}$) was the same as for the previous ones but the mode of control and the state and state-action spaces were changed. The simple actions available to the agent were 1,2,3 and 7 (Table 1, inverse kinematics column), x = 35 cm for walk (the size of a single step) and x = y = z = 5 cm for the motion of a hand $gamma$ = 0.95. Thus a 3-dimensional cube of x,y,z positions around the avatar's hand has been defined, the last dimension was walk along one axis. Therefore the agent could choose from 8 simple actions - hand motion along 3 spatial axes in two opposite directions for each axis plus walk (2*3+2). The total simulated state space was 1 296 000, however this initial number was highly redundant and could be reduced to 6720 (8 states for the x-direction, 14 for y and 10 for z, plus 6 positions for the walk). This reduction will be demonstrated in the second task (see below). Similarly as before the solution was usually found in a few hundred iterations.

4.2 The teapot lifting task

The goal here was to lift a teapot (z co-ordinate of the teapot position had to increase). Therefore the agent was rewarded whenever the end position of the teapot was higher than the start position. The simple actions available to the agent were selected from Table 1, for the FK

these were actions 1,2,3,4,6,7 (Experiment $\mathcal{D}$ and $\mathcal{E}$). Unlike in the previous task, the agent was assigned an additional action (action 6 - grab an object) which improved the resulting animation and served as a means of representing the state of the teapot (grabbed/not grabbed). The learning parameters were set as follows: alpha was set to 20 degrees in Experiment $\mathcal{D}$ and to 10 degrees in Experiment $\mathcal{E}$ and $gamma$ = 0.95. The difference between experiments $\mathcal{D}$ and $\mathcal{E}$ was not in the size of the state-space but in the sampling of it - the space axes were sampled more densely and thus the state space size of the second task was larger. Thus the state-space in the first experiment consisted of about 13000 states, and for the second one was about ten times bigger (121 000 states). The dimensionality of both tasks was 5 - 2 degrees of freedom for the left arm, 1 for the left forearm, 1 for hand rotation and 1 for the state of the teapot. The respective size of these dimensions were 7, 12, 11, 7, 2 for Experiment $\mathcal{D}$ and 12, 20, 18, 14, 2 for Experiment $\mathcal{E}$. Ten simple actions were available to the agent at each time step (2 for each state-space dimension, as described earlier). The minimum number of iterations for the first experiment was about 4 000 and 20 500 for the second one. Thus, despite the tenfold growth of the state-space, the minimum number of iterations increased by a factor of about 5. The lengths of the best solutions were 9 and 14 respectively.

Similarly an experiment with biped control using inverse kinematics was also conducted (Experiment $\mathcal{F}$). The simple actions available to the agent in this case were actions from Table 1 (actions 1,2,3,7 of the inverse kinematics column), and x = y = z = 8 cm for the motion of a hand, $gamma$ = 0.95. Therefore the state-space was 4-dimensional and the agent could choose from 8 simple actions - hand motion along 3 spatial axes in two opposite directions for each axis plus the grabbing action. The total size of state space was 2240 (8*14*10*2). The algorithm needed about 800 iterations to find a solution, and the best solution found was 10 actions long.

For both tasks the number of states across each dimension was chosen to provide sufficient sensitivity but also to eliminate as many unnecessary states as possible. Therefore only reasonable angles for joint movements were selected, these were taken from human joint constraints: forearm can only rotate by about 180 degrees around the x-axis, arm 270 degrees around the x-axis (forward/backward) and 180 degrees around the y-axis (up/down). Two additional states were added for each joint to represent the illegal motions, so called forbidden states (e.g. for the forearm rotation -20 degrees and 200 degrees would be the forbidden states). For walking only the route through the door was represented as walking to the door could easily be achieved within the underlying cognitive system (FreeWill [11]). Finally in all experiments the Q-table was represented as a lookup table and the values were initialized to 0 before the simulation.

Results of all experiments are summarized in Table 2 and 3, the number of actions required to find a stable optimum solution would normally be 1.5-2 times higher than those presented in the two tables. The following two graphs show the convergence for both FK and IK (Figure 3).

Table 2. State space details for FK

Task	FK		
	State space	Actions/state	iterations
Door	1 296 000* (6720)** (4D)	8	700
Teapot	2240 (4D)	8	800

Table 3. State space detail for IK

Task	IK		
	State space	Actions/state	iterations
Door	12672 (4D)	8	200
	2 141 568 (6D)	12	1500
Teapot	12936 (5D)	10	4000
	120 960 (5D)	10	20500

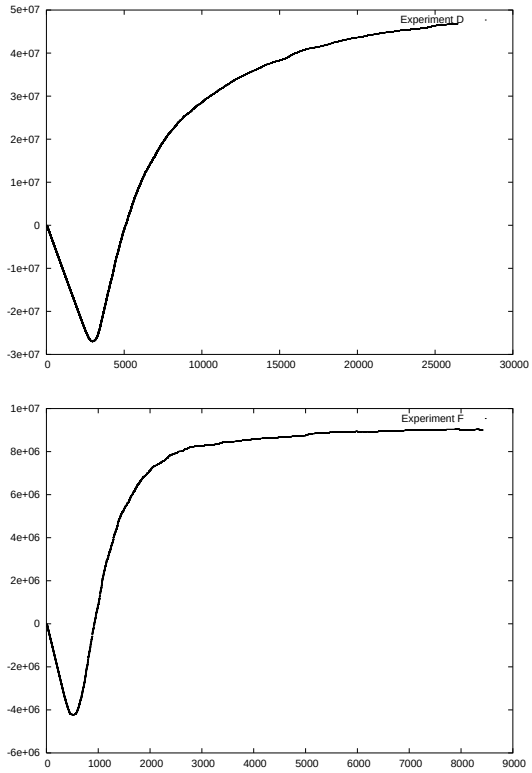

Figure 3. Convergence for forward and inverse kinematics examples

5 Metrics

The avatar learning task can be cast as an optimisation problem formulated over the number of actions necessary to fulfil the goal. This means that, apart from trying to find a set of transition states from the starting position to the goal state, the avatar also optimises the number of actions necessary to achieve its goal. The learning process might have more than one solution, all of which leading plausible and comprising the exact same number of steps (imagine the action of lifting a cup: many solutions are viable and plausible, all merely constrained by the position of the cup and by the bio-mechanical constraints of the arm). If the application is animation, and in any case if the set of solutions ought to be analysed, then if the number of such solutions is low, they can all be presented to a human operator who would select the 'best' ones. The human user/operator might use qualitative criteria, for instance how a specific sequence of actions might be perceived as realistic. On the other hand, when many

different solutions exist, it would be desirable to automate the comparison and selection process. The following sections describe valid metrics that might serve this purpose.

5.1 Local distance metric

The local distance metric (LDM) selects a pair of animation sequences and generates a distance value depicting the similarity of these sequences - the higher the distance value, the more dissimilar the two sequences are. The distance, which is commutative, is calculated on the basis of individual actions in the sequence, located on corresponding positions within the string. Each two such actions are compared and if they appear to be different the total distance for the sequence is increased by one, otherwise it remains the same. The resulting set of distance values for all permutations of the action sequence pairs in the input solution set is then searched for the maximum and the resulting subset can be generated as pairs of sequences for which the distance is equal to the maximum. It is also possible to add other pairs with the distance value lower than the maximum within a given margin, thus allowing the user to influence the size of the presented subset. The formal definition of the metric is given below.

Let us consider two learnt sequences of the same length N, A and B, consisting of a number of low-level composite actions each:

$$\mathcal{A} = a_1, a_2, ..., a_M \tag{1}$$

$$\mathcal{B} = b_1, b_2, ..., b_M \tag{2}$$

The length of each sequence, that is their cardinality, is thus given as $N = card(\mathcal{A}) = card(\mathcal{B})$. A distance between two sequences $\mathcal{A}$ and $\mathcal{B}$ can be now defined (similar to the Hamming distance [3])as:

$$d_{\mathcal{AB}} = d_{\mathcal{BA}} = \sum_{i=1}^{M} |sign(a_i - b_i)| \tag{3}$$

The local distance metric finds all pairs of sequences for which the distance is equal to the maximum distance found between any two sequences in a given set:

$$d_{max} = max\{d_{ab}\}, \forall a \in \mathcal{A}, \forall b \in \mathcal{B}, a \neq b \tag{4}$$

where A is a set of all sequences found for a given task.

5.2 Global distance metric

The local distance metric from the previous section finds sequences, which substantially differ from each other, but it does not guarantee that the resulting set will be well sampled. This is because one sequence very different from all other ones will affect the maximum distance and cause this particular sequence to be paired with a number of other solutions which may nevertheless be mutually similar. The global distance metric (GDC) relies on finding an average sequence, from which the distance of individual sequences can be assessed following the model defined by the local distance metric. The average sequence is created by finding, for each consecutive position in the sequence, an action which appears most frequently on this position, considering all the action sequences from the input set. The optimal sequence is thus defined as:

$$\mathcal{S} = \{\underset{i}{argmax}(h_{a_1}(S_i)), \cdots, \underset{i}{argmax}(h_{a_M}(S_i))\} \tag{5}$$

where the function $h_{a_j}(s_i)$ indicates the a_j bin (and also the j^{th} action $\in \mathcal{A}$ available set) of the histogram of occurrence over all actions in sequence S_i. The optimal sequence is built by using the $argmax$ - over all available sequences ($\forall S_i$). Each $argmax$ operator returns the action a in the set $\mathcal{A}$ of available actions, yielding the maximum among all histograms. If a histogram has more maxima, then one of such actions is chosen at random.

5.3 Action similarity metric

This metric is similar to the local distance metric in that it considers pairs of action sequences. The number of occurrences of each composite action within both sequences are counted and compared between the two sequences, the resulting differences in occurrences of each composite action are added up. The result is a scalar, which is higher if the sequences consist of a larger number of different composite actions. Thus the similarity metric for two action sequences S_a and S_b is defined as:

$$d_{\mathcal{AB}} = d_{\mathcal{BA}} = \sum_{j=1}^{M} |h_{a_j}(S_a) - h_{a_j}(S_b)| \tag{6}$$

where L is the number of all different low-level actions required to train the avatar.

6 Results

Based on two result sets generated by the learning algorithm the Local Distance Metric and the Global Distance Metric have been calculated. The first result set (Set 1) contained 651 sequences each consisting of 14 actions obtained from experiment 2.2, the second set contained 72 sequences (Set 2) with 11 actions each, generated by experiment 2.3. LDC applied to Set 1 generated 120 sequences with the maximum distance of 12. LDC applied to Set 2 generated 72 pairs of sequences with the maximum distance of 8. GDC applied to Set 1 generated 40 sequences most distant from the averaged sequence, the maximum distance was 8. GDC applied to Set 2 generated 4 sequences most distant from the averaged sequence, the maximum distance was 6. The four resulting sequences were:

2 4 1 1 4 2 2 0 4 6 0 (seq 25, see Fig. 4)
2 4 1 4 1 2 2 0 4 6 0 (seq 29)
4 4 1 1 2 2 2 0 4 6 0 (seq 60)
4 4 1 2 1 2 2 0 4 6 0 (seq 65)

Although these sequences are the most distant from the average, they are relatively similar and therefore an extension of this approach has been proposed. The resulting set proposed for addition to the plan library should include sequences for which the GDC generates the highest and the smallest values thus guaranteeing to include both most average and most different sequences. Rerun of the GDC algorithm with a distance of 0 resulted in further 4 sequences being proposed for the action library:

1 1 4 4 4 2 2 2 0 6 0 (seq 8, see Fig. 4)
4 1 1 4 4 2 2 2 0 6 0 (seq 41)
4 1 4 1 4 2 2 2 0 6 0 (seq 46)
4 1 4 4 1 2 2 2 0 6 0 (seq 51)

This extension of the algorithm applied to Set 1 generated additional 10 sequences.

The results indicate that for large sets of sequences the sets of sequence pairs generated by the LCD are too large to be analyzed

individually. The GDC metric reduces the amount of resulting sequences, however the most distant solutions appear to be very similar. The proposed extension of generating both most distant and most average solutions and selecting a few examples from each set appears to overcome this problem.

7 Final remarks

A filled Q-table contains optimum transitions from any state to the goal state. Therefore the results of the learning tasks demonstrated in this paper can be used to create animations from any starting position of the agent thus adding more than one specific action to the plan library at a time. It may even be possible to store the learnt Q-tables rather than single action sequences and use them whenever the agent needs to perform an action starting from a state present in the Q-table and ending in a goal state defined for the specific learning task for which the Q-table was generated.

Similarly the Q-learning technique may be used to learn the same action for a different configuration of objects (for example the size of the table or teapot may be different) and encode this different configuration as a set of varying preconditions for the same action. Equally the size of the avatar itself can change and a new solution for a different character may be generated (see [6] for a different solution to this problem).

Another benefit of using the learning technique for action acquisition is that the resulting sequence comes in a form of a script, which can then easily be manipulated and parameterised and also incorporated into other animation tools. This is in strong opposition to the key-framing and motion capture based approaches which although easily portable and capable of delivering realistic motion, are nevertheless difficult to modify using automatic methods. This advantage is also utilized when calculating the metrics presented in the previous sections, as it is difficult to calculate metrics for motion sequences without underlying semantic representation.

Finally the technique can be applied to other research domains such as robotics, provided the robot can already perform basic actions such as walking.

The biggest limitation of this approach is currently fast increase in the learning time with the size of the state space. Future research will attempt to tackle this problem.

Acknowledgments

Research presented in this paper was partially funded by the British Council/Polish Committee for Scientific Research Young Scientists Grant project no. WAR/342/12. The authors also acknowledge the grant from the Spanish Ministry of Science and Technology Code: TIC2003-8496-C04-01.

REFERENCES

[1] L. C. Baird and A. W. Moore, 'Gradient descent for general reinforcement learning', in *Advances in Neural Information Processing Systems 11*, eds., M. S. Kearns, S. A. Solla, and D. A. Cohn, Cambridge, MA, (1999). MIT Press.

[2] D.P. Bertsekas and J.N. Tsitsiklis, *Neuro Dynamic Programming*, Athena Scientific, 1996.

[3] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley Interscience, 1991.

[4] J.J. Craig, *Introduction to Robotics: Mechanics and Control*, Addison Wesley, 1989.

[5] J. Funge, *Making Them Behave: Cognitive Models for Computer Animation*, Ph.D. dissertation, University of Toronto, 1998.

[6] J. K. Hodgins and N. S. Pollard, 'Adapting simulated behaviors for new characters', in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 153–162. ACM Press/Addison-Wesley Publishing Co., (1997).

[7] D. Isla, R. Burke, M. Downie, and B. Blumberg, 'A layered brain architecture for synthetic creatures', in *Proceedings of Seventeenth Joint Conference on Artificial Conference IJCAI-01*, pp. 1051–1058, Seattle, USA, (2001).

[8] J.-S. Monzani, *An architecture for the Behavioural Animation of Virtual Humans*, Ph.D. dissertation, Ecole Polytechnique Fdrale de Lausanne, 2002.

[9] K. B. Russell and B. Blumberg, 'Behavior-friendly graphics', in *Computer Graphics International*, pp. 44–, (1999).

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

[11] A. Szarowicz and P. Forte, 'Combining intelligent agents and animation', in *AIxIA 2003 - Eighth National Congress on AI, Lecture Notes in Artificial Ingelligence, vol 2829*, Pisa, Italy, (2003). Springer-Verlag.

[12] B. Tomlinson, B. Blumberg, and D. Nain, 'Expressive autonomous cinematography for interactive virtual environments', in *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, Spain, (2000).

[13] B. Tomlinson, M. Downie, M. Berlin, J. Gray, D. Lyons, J. Cochran, and B. Blumberg, 'Leashing the alphawolves: mixing user direction with autonomous emotion in a pack of semi-autonomous virtual characters', in *Proceedings of the ACM SIGGRAPH symposium on Computer Animation*, San Antonio, Texas, (2002).

[14] C. J. C. H. Watkins, *Learning from Delayed Rewards*, Ph.D. dissertation, Cambridge, Psychology Department, 1989.

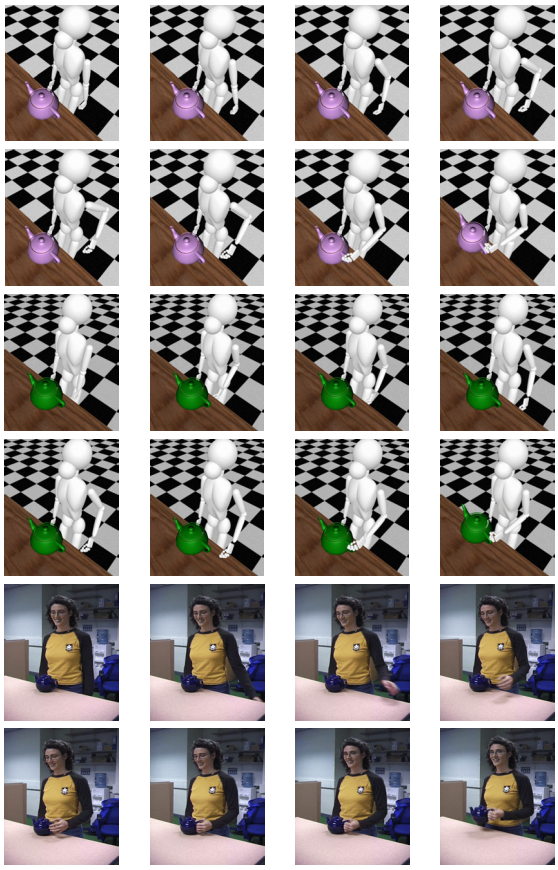

Figure 4. Top 8 frames refer to sequence 25, middle 8 frames refer to sequence 8 and last 8 frames a qualitative comparison with same action performed by a person