ISSN 2007-9737

Autonomous Motion Planning for Avatar Limbs

Cristian E. Boyain y Goytia Luna[1], Andres Mendez Vazquez[1],
Marco Antonio Ramos Corchado[2]

[1] Instituto Politécnico Nacional, Centro de Investigación y Estudios Avanzados, Jalisco, Mexico

[2] Universidad Autónoma del Estado de México, Toluca, Estado de México, Mexico

{cboyain, amendez}@gdl.cinvestav.mx, marco.corchado@gmail.com

Abstract. In this work, a new algorithm for autonomous avatar motion is presented. The new algorithm is based in the Rapidly-exploring Random Tree (RRT) and an appropriate ontology. It uses a novel approach for calculating the motion sequence planning for the different avatar limbs: legs or arms. First, the algorithm uses the information stored in the ontology concerning the avatar structure and the Degrees Of Freedom (DOFs) to obtain the basic actions for motion planning. Second, this information is used to perform the growth process in the RRT algorithm. Then, all this information is used to produce planning. The plans are generated by a random search for possible motions that respect the structural restrictions of the avatar on kinesiology studies. To avoid a big configuration space search, exploration, exploitation, and hill climbing are used in order to obtain motion plans.

Keywords. Avatar, rapidly-exploring random tree, degree of freedom, ontology, kinesiology.

1 Introduction

As it is very well defined in [13], planning is searching for and designing of a sequence of actions that allow reaching an objective. The problem of planning in a situation where there is an initial or actual state and the objective or desire either to arrive at or obtain a final state has been considered by many researchers in the areas of robotics, control, and computing, and as a result there emerged several approaches for solving this problem.

In motion planning for autonomous animation, the objective is to compute a trajectory from an initial configuration to a goal configuration. This is applicable to avatar animation. In this area, an additional desirable objective would be that the motion plan algorithm could produce “realistic” animated movements and reduce human intervention in the animation. By “realistic” we mean motion plans that contain some degree of randomness. After all, even most incredible athletes do not repeat the same movement with millimetric precision.

This is motivated by the increasing use of computer animation in different areas such as films, video games, and disaster studies. Many new techniques have been developed to help in elaboration of these animations. Currently, a field of great interest is the humanoid avatar animation due to its use in a 3D world animation and robotics [9, 16, 29].

If an avatar has to move, it is expected and desired that the movements look “realistic”. This means that the movements produced by the avatar should respect the constraints imposed by the human body, and certain variability should be allowed. By variability we mean that if the same “realistic” movement is executed by all avatars with the same parameters as time goes, a certain odd perception will start to arise since even in the same human being performing the same movement certain variability exists. Therefore, it is highly desirable that, as in humans, the same movement should vary to a certain degree from motion plan to motion plan and from avatar to avatar.

Algorithm planning has been used for avatar animation in many scenarios [9, 16, 21, 23]. However, motion planning has not been used in general for avatar animation due to restrictions in the number of DOFs, joint lengths, and the unrealistic movements produced by the algorithms for autonomous animation.

Many approaches have explored avatar motion planning such as genetic algorithms [18], swarm intelligence [17], and sampling base motion planning (SBMP) [4, 8, 6, 22]. Some success has been achieved using these algorithms; however, many of them are off-line algorithms. Nevertheless, the SBMPs have showed some promises due to the way the Configuration Space (CS) [14, 25] is defined. For example, it allows obtaining better obstacle avoidance due to the implicit CS geometry [14]. Examples of these algorithms are the probabilistic road-map algorithm [12] and the Rapidly-exploring Random Tree (RRT) algorithm [13, 27].

In these kinds of algorithms [7, 13, 28], the incremental sampling and searching approach is used to assure excellent search coverage over the configuration space, which allows the avatar to reach a variety of movements. This is despite of the fact that usually just a certain range of the configuration space is required to accomplish a motion task.

In particular, motion planning using the RRT algorithm as in [13, 17, 28] implements an efficient sampling strategy used to build a tree search structure. In these methods, an initial state/configuration is considered as the tree root. Then, new nodes are randomly added as new branches are generated. Finally, when the objective is reached, the best trajectory in the tree is selected by means of backtracking.

Based in this observation, our algorithm uses the RRT search tree growing strategy to give a variability and CS coverage in avatar motion planning. In addition, it uses a version of the hill climbing method [19] which reduces the tree growth, and it guides the search in order to produce efficient movements. Therefore, the algorithm presented in this paper avoids the use of inverse kinematic for motion planning and search over all the CS. This reduces the computational complexity and allows producing an on-line algorithm.

This paper is organized as follows. Section 2 provides related work on motion planning. The motion planning problem and formalization are defined in Sections 3 and 4, respectively. Section 5 outlines motion planning with avatar ontology. The autonomous motion planning algorithm is described in Section 6, while Section 7 presents simulation and results. Finally, conclusions are given in Section 8.

2 Related Work

There exists vast literature [8, 16, 17, 30, 31] on motion planning using animated avatars or robots. In this section, some motion planning algorithms are presented according to whether the obtained plans are based on an articulated structure or not.

2.1 Motion Planning for Non-Articulated Structure

Various motion planning works have been done for a non-articulated structure implementation. To mention a few, the researcher in [10] uses Genetic Algorithms (GA) to generate paths that can avoid different obstacles of the terrain. The algorithms employ a fitness function to find the better plan paths. Also by implementing a sampling method, [8] uses a tree structure that can be pruned and grow according to possible obstacles in the environment, then generating plan paths.

2.2 Motion Planning for Articulated Structure

When a motion plan for an articulated structure is required, many algorithms exist to deal with this issue: for example, GA [18] and ant colony optimization [17] algorithms. These motion planning algorithms have to handle aspects like the CS and the DOF of the articulated structure. A drawback of these algorithms is that the combination of CS and DOF implies a humongous state space due to the possible positions that each joint in the articulated structure can achieve [3, 5, 26].

In the case of Inverse Kinematic (IK) [2, 20, 24], some drawbacks arise due to the complexity of the formulation. For example, if the articulated structure increases its number of joints, it requires modifications in the system formulation.

In addition, IK algorithms are more suitable for synthesizing posture than animation [16]. In order to resolve the explosion of state space, SBMP have been proposed not only because they are successful in handling a huge configuration space, but that they also provide an adequate framework for planning with many DOFs [22].

These are the reasons why we have made use of the SBMP approach to propose a motion planning algorithm able to work with a more flexible DOF representation. It accomplishes the planning and reduces the structure dependence in order to generate motion plans.

3 Problem Statement

The motion planning problem for an avatar in a 3D world will be represented by generation of plans that give us trajectories able to accomplish the avatar task. For this, we use the following model.

Let A be an avatar that has to accomplish a motion task, and $J = (j_1, \dots, j_n)$ are the joints that A has. Each joint in turn has $q_i = \{x | x \in Vector\ Space\}$, $i = 0, \dots, n$ constrained by DOF of that joint. The DOF configuration of A forms a system configuration $Q_J = (q_1, \dots, q_n)$.

The state space, where all $Q's$ of A are possible, is represented by the n-dimensional configuration space C_a. This state space has the following hierarchy: C_a represents all possible configurations of A due to the avatar structure, $C_{a-collide}$ represents the subset of states that are not allowed after collisions have been taken in account, and finally $C_{a-free} = C_a - C_{a-collide}$ which represents all possible useful configurations.

Then, the objective is to compute a sequence of actions, a plan P, which can take the J joints in the initial configuration Q_{init} to a goal configuration Q_{goal}. Our motion problem in CS is represented in Figure 1.

4 Planning Formalization

For the motion planning problem handled in this work, revolute joints are considered and the principal concepts of state and action are set as follows.

A state s_k is represented as the tuple $s_k = (k, q_k, \boldsymbol{v_k})$, which represents the configuration for the k-th joint, q_k is the quaternion representing the k joint rotation, and the translation vector $\boldsymbol{v_k}$ indicates the joint position.

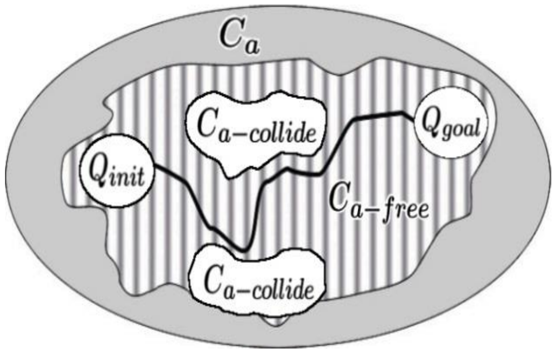

Fig. 1. Components of the plan of movement and its relation with the configuration space C_a

The joint rotation is represented by a quaternion $q_k = (\theta_k, \hat{u}_k)$ where the rotation angle is represented by θ_k and $\hat{u}_k$ is a unit vector representing the axis of rotation.

In order to represent an action $a_x = (x, q)$ in the plan, we set it as a motion for the x-th joint in q rotation angle.

The possible actions which could be generated are limited due to the avatar DOFs and its motion limits.

This motion action representation will be expressed in STRIPS [19] form in order to give a better implementation understanding:

$$\begin{aligned} &a_k(k, q) \\ &= pre\colon currentstate(s_k) \wedge freestate(s'_k) \\ &\qquad pos\colon \neg currentstate\ (s_k) \wedge \\ &\quad currentstate(s'_k) \wedge \neg freestate(s'_k). \end{aligned} \tag{1}$$

An action will be considered to be usable in the plan as long as the new joint state is free of collisions.

For us, the plan P is terminated once the goal v_{goal} is reached, or once P has moved the avatar configuration to a state close enough to the goal $\boldsymbol{v}_{goal}$. This is different from IK where the final state needs to be reached all the time.

Although an avatar has J joints that could be used, not all of these joints are required to accomplish its motion task, i.e., depending of the task, some joints will never be moved. Therefore, taking into account all unnecessary joints will only increase the complexity of the motion planning, i.e., the search in the CS. This is due to a bigger joint configuration space.

The proposed algorithm handles the previous problems by generating motion plans using a cardinal variable task joint subset $L \subset J$. In other words, the proposed algorithm is not set with a predefined number of joints.

The use of L decreases the search space, but the number of possible configurations of the joint DOFs is still large. In this work C_a is decomposed by quantization of each angle of rotation θ_x for x joint DOFs of motion degrees:

$$D_x = \{n \in \mathbb{N} \mid n \bmod \Delta_p = 0 \wedge (\Theta_{min} \leq n \leq \Theta_{max})\}. \quad (2)$$

This allows reducing the state space of DOFs to j elements and keeping coverage of all state space. Here Θ_{max} indicates the maximum and Θ_{min} the minimum rotation angle for the DOF x.

Therefore, the set of all possible states that the avatar can reach in C_a is given by

$$S = \bigcup_{i=1}^{N} D_i, \quad (3)$$

where D_i is the state set for each DOF, and N is the number of DOFs used in the motion planning task.

In order to give a more understandable representation of an action and to define its rotation axis, $\hat{u}$ is specified according to the motion and orientation over all possible DOFs: yaw, pitch, and roll. They form a set of actions for each DOF at each joint. This action set will be defined over all possible motion degrees of each DOF. Using action definition, we can define the set M as all permissible actions:

$$M = \left\{a_k(k,q) \mid \Theta \in D_i \wedge \left(\Theta_{min} \leq (\Theta_k - \Delta_p) \wedge (\Theta_k + \Delta_p)\right) \leq \Theta_{max}\right\}, \quad (4)$$

where the Θ_k value represents the current k rotation angle in which the action a_k is going to be executed. Θ_{min} and Θ_{max} are taken from kinesiology studies [15], this ensures that our planning algorithm will not implement actions forbidden to the human physical structure.

5 Motion Planning with Avatar Ontology

Avatar independence is an important aspect of this work due to the fact that it allows our algorithm to be implemented by different avatars encouraging a more autonomous animation.

Knowledge Base (KB) Agent represents a feasible approach for us; here agents help an avatar to achieve independence. Works like [11] propose an ontology for KB agents, which complements our motion planning algorithm by providing the behavior and internal structure of an avatar that requires to achieve a motion task.

In this work a similar ontology is implemented, which provides the avatar internal structure knowledge (like joint DOFs and rotation limits) for the planning algorithm to generate a specific avatar motion plan.

6 Autonomous Animation Motion Planning

In order to generate motion plans independent of the avatar, the joint ontology information that the KB agent has is passed to the motion planner. The pseudo-code for this procedure is given in Algorithm 1.

This works in the following manner. First, the planning algorithm performs a search of possible configurations that let it move the J joints. This generates a tree structure, where each node represents all possible actions. Second, the motion planner is initialized with the ontology information and the initial system configuration according to the avatar that implements the plan. Now, for the planning algorithm, the joints are numbered from 1 to m, where m represents the final joint. We define the j_m joint as the end effector; it can be one of the following joints: the last joint of the hand, that of the foot, or the abdomen joint.

This end effector is used to determine the moment when a task has been finalized. For example, to reach a glass of water means to put the hand joint as near as possible to the glass of water. Also, to move to a door could mean to move the abdomen joint as near as possible to that door.

Algorithm 1 Autonomous Animation Motion Planning

Requires: avatar ontology instance, motion plan goal, $\boldsymbol{v}_{goal}$, and the L joint subset involved

Ensures: motion plan that allows reaching the plan goal starting from the initial avatar configuration

```
1:   goalReached ← false
2:   Plan ← motionPlanner( AvatarOntology, $\boldsymbol{v}_{goal}$, $L$)
3:   repeat
4:     if distance( $j_m$ ) ≤ threshold then
5:       goalReached ← true
6:     else
7:       Plan ← growthPlan(lastActionExecuted)
8:       lastActionExecuted ← execute(Plan)
9:     end if
10:  until goalReached = true
```

Algorithm 2 growsPlan

Requires: latest action plan that has been executed
Ensures: grows the plan, once a new action is added from the last action executed

```
1:   while $i \leq x$ do
2:     $a_{rand}$ ← randomAction($L$)
3:     if isNull($a_{rand}$) = false then
4:       addAction($a_{rand}$, lastActionExecuted)
5:     end if
6:     $i \leftarrow i + 1$
7:   end while
```

Now, with the tree structure initialized, the distance j_m between the end effector and $\boldsymbol{v}_{goal}$ is used to check if the plan should terminate. For this, the algorithm uses a threshold set by the user. If the goal has not been reached, the action search continues. Then, after x added actions, the tree formed with the found actions is checked in order to find the best possible plan so far that the avatar is going to execute. This process is repeated until the j_m joint achieves the goal threshold.

The growth process of the tree search is given in Algorithm 2.

As the algorithms show, actions in C_a will be obtained through a random process. Each action is checked to find if the random process was unable to generate a movement for the chosen joint/node to grow the tree. This only happens when a generated action violates the motion restrictions set in the ontology, i.e., an action that leads to an unreal configuration. If a valid action is obtained, it is added to the tree structure.

The growth process presented in Algorithm 2 shows an important difference between our implementation and the RRT. This is due to the fact that our method does not require having the whole action plan to start executing movements that bring us closer to the goal. Even when new obstacles are discarded or added to the environment, the proposed planning algorithm can build an alternative plan starting from the last executed action.

By using just the $L \subset J$ joints, a_{rand} is obtained. Now, we have the action and we know the last executed action, so we can add the action to the search tree.

Given a_{rand} and the last executed action *lastActionExecuted*, Algorithm 3 looks for an action in the search tree that is closer to the random one, because in that place the new action will be added to the search tree. In order to do this, Algorithm 3 compares the resulting distance value: if the a_{better} was executed, it assigns x_1 if we are close enough according to a threshold, otherwise it assigns x_2 as the rotation angle Θ, where $x_1, x_2 = \{\mathbb{R} \mid (x_1 < x_2)\}$; in our work, values of 5 and 10 were used respectively.

The new Θ value replaces the previous value from the a_{rand} action, and it is added to the tree.

Something that influences the design of the proposed algorithm is based in the following observation of human motion: most of the time, a person moves a joint until the joint motion does not give any perceptible benefit. Using this observation, Algorithm 3 implements the exploitation action decision dilemma used in some RL problems where an action is executed several times until the goal is reached or until the action fails to reduce the goal distance.

The new action that has been added is exploited as presented in Algorithm 4.

As in Algorithm 3, the rotation angle value is set by using the distance. This new action is verified if it gets the avatar closer to the goal and if the joint is kept inside its motion limits in order to add the action into the search tree. The exploitation process of the new added action ends if the actions obtained move the j_m joint away from the goal.

Algorithm 3 addAction

```
Requires: the $a$ action that will be added and the last executed action
Ensures: action $a$ added
1:  $a_{better} \leftarrow$ nearesAction($a_{rand}$, lastActionExecuted)
2:  if distance($a_{better}$) < threshold2 then
3:  $\Theta \leftarrow x_1$
4:  else
5:  $\Theta \leftarrow x_2$
6:  end if
7:  $a_{rand} \leftarrow \Theta$
8:  addAc($a_{better}$, $a_{rand}$)
9:  exploitAction($a_{rand}$)
```

Algorithm 4 exploitAction

```
Requires: action $a_{current}$ to exploit
Ensures: action $a_{new}$
1:  if distance($a_{current}$) < threshold2 then
2:    $\Theta \leftarrow x_1$
3:  else
4:    $\Theta \leftarrow x_2$
5:  end if
6:  $a_{new} = a_{current}$
7:  $a_{new} \leftarrow \Theta$
8:  if distance($a_{new}$) < distance($a_{current}$) then
9:    if (withinRangeMotion($a_{new}$) = true) then
10:     addAc($a_{current}$, $a_{new}$)
11:     exploitAction($a_{new}$)
12:   end if
13: end if
```

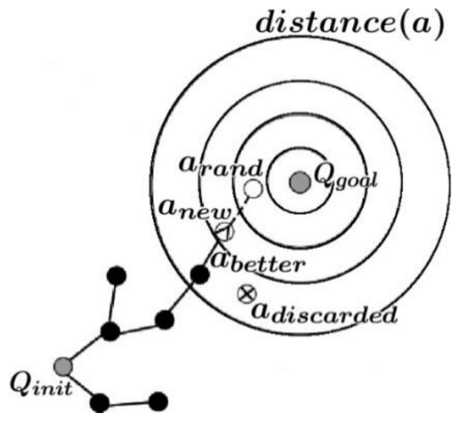

Fig. 2. The proposed planning algorithm

Using these concepts of exploitation, valid actions are obtained in a faster way.

In this manner, the motion planning algorithm is designed to generate motion plans for different avatars, requiring only the information of the joints that it has to move by taking advantage of the concept of exploitation, the efficient RRT search strategy, and the CS decomposition. The entire process of the proposed algorithm is presented in Figure 2.

7 Simulation and Results

In this section the animation of a human avatar's left arm is generated by using ontology implementation [11] and the proposed motion planning algorithm. For this simulation, the goal is to reach the object by using the avatar left arm. Motion planning and the avatar structural parameters are presented first. Second, we performed another case study where the left arm structure is modified and the number of DOFs is increased.

Finally, the images of a motion plan simulation are presented.

7.1 Planning Parameters

The proposed algorithm was applied to the avatar left arm shown in Figure 3. The arm has eight DOFs distributed as shown in this figure.

The configuration space corresponds only to the left arm scope, C_{L-arm}, where no obstacle is present. In addition, $C_{L-arm} = C_{free}$. The system configuration due to the involved joints $L = (j_1, \dots, j_4)$ is $Q = (q_1, \dots, q_8)$.

The DOF distribution that composes the configure system over the joints involved is presented in Table 1.

Table 1. System configuration

Joint	Degree of freedom		
	Pitch	Roll	Yaw
Shoulder	X	X	X
Elbow	X		
Wrist	X	X	X
Hand	X		

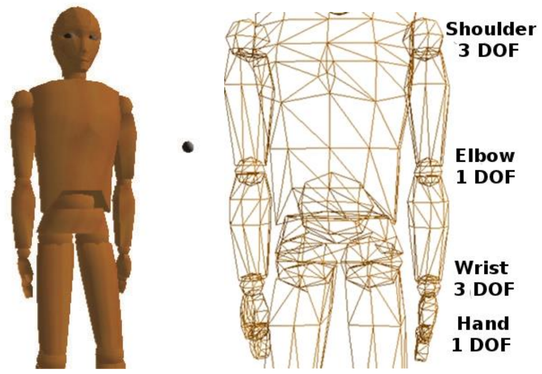

Fig. 3. Avatar used in case studies

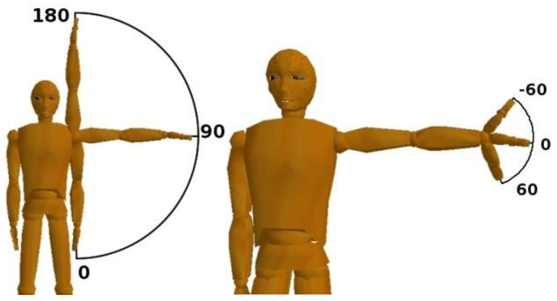

Fig. 4. Joint motion limits

It is important to set the distance function to know if we get closer to the solution. In this simulation, we use the Euclidean distance between the end effector and the goal position to guide the movements.

7.2 Movement Limits

A standing person has a natural position as shown in Figure 3, and in our work this position is the initial system configuration Q_{init} from where it has to reach a goal configuration Q_{goal}.

Although only the left arm is used, the CS is wide, but the decomposition of the motion degree helps to cover all the search space without specifying all possible configurations. After some experiments, $\Delta_p = 10$ gives good space search coverage for the simulation.

By matching the kinesiology information [15] to the different DOFs in the avatar left arm, we can obtain the motion limits, Θ_{min} and Θ_{max} of each DOF to be saved in the ontology. This ontological information is presented in Figure 4 and Table 2.

Table 2. Ontological information

Joint	DOF limit		
	Pitch	Roll	Yaw
Shoulder	90°, -90°	180°, -50°	180°, 0°
Elbow	140°, -10°		
Wrist	60°, -60°	90°, -90°	30°, -20°
Hand	90°, 0°		

7.3 Many DOFs Planning

In order to illustrate that the planning algorithm can generate plans independent of a specific avatar structure and a predetermined number of DOFs, the avatar left arm is modified by adding the elbow joint two times. This modification increases the arm wide range and the number of DOFs. In addition, they have the same DOF number and distribution, i.e., the number of extra joints equals the number of elbow joints. Therefore, the system configuration of this set of joints $L = (j_1, \dots, j_6)$ is $Q = (q_1, \dots, q_{10})$.

7.4 Results

Figure 5 presents animation stills of an avatar reaching a sphere in front of it. The motion plans generated in different experiments shown in Figure 5 were limited to a specific number of motion movements to reach the goal. In addition, the design work was limited to the definition of the goal within the configuration space C_{L-arm}. The goal motion limitations were selected to avoid unnecessary actions by the motion planner.

In addition, the threshold was selected such that the proposed algorithm could avoid a race condition at the goal state.

Motion plans generated in the experiments implement natural motions, i.e., motions that a common person could make. Also plans starting from the same initial configuration and having the same goal configuration have differences between them, this enhances the motion realism.

In Figure 6 animation stills of the avatar arm with many DOFs are presented. This animation

ISSN 2007-9737

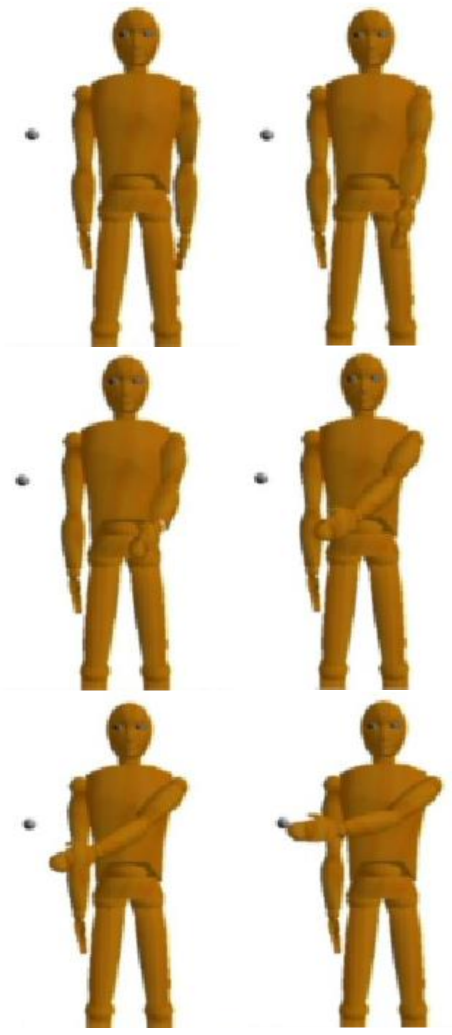

Fig. 5. Simulation results where the goal is in front of the left arm

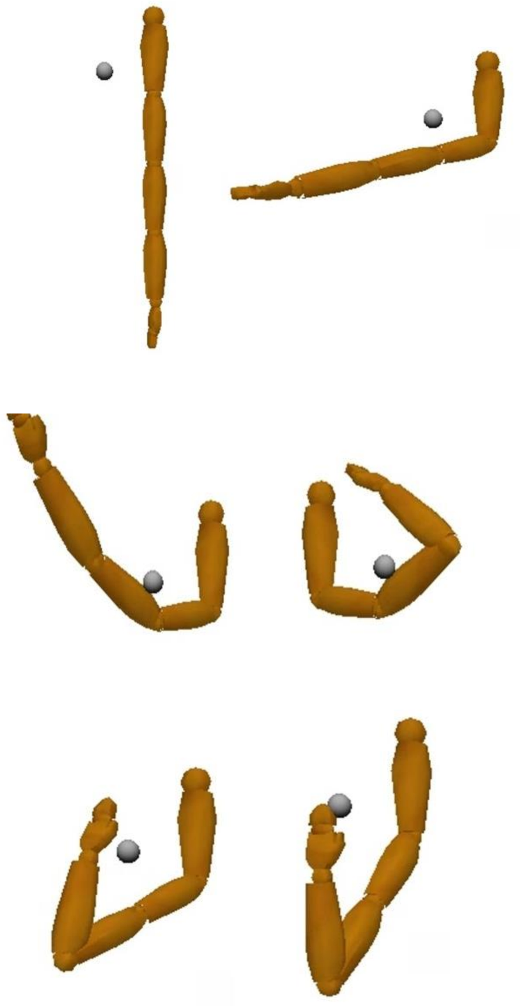

Fig. 6. Arm with many DOFs

shows that it was able to generate motion plans with different number of DOFs assigned to the motion task.

8 Conclusions

This work presents a motion planning algorithm for avatars, which contributes to the generation of a motion sequence for simple animations in a more autonomous and realistic way. More specifically, the main contributions of our work are the following:

- The proposed algorithm was able to work with a big number of DOFs that are needed to accomplish the task, and for this the DOF configuration space is discretized by constraining the possible rotation angles for each DOF.
- As long as the ontology provides the avatar internal structure, without the exact joint dimensionality and position, a motion plan is generated for the avatar.

- Realistic but maybe not optimal plans are generated by constraining the joint motion range using kinesiology information.

The work presented here shows an improvement in generation of autonomous animations. However, our algorithm generates plans only for a small motion task. A more complex motion planning algorithm is required to perform more complex movements like carrying objects or running. Also, our proposal is sensitive to the threshold and Δ_p values. Unsuitable values may result in motion plans with configurations caught in suboptimal positions or unable to reach $\boldsymbol{v}_{goal}$.

Although the algorithm handles collisions with obstacles in the environment and is able to modify the trajectories, the implementation of a process for collisions with itself is necessary. This means that such DOF configurations are allowed that may overlap the avatar with itself.

Finally, autonomous animation generated by this motion planning algorithm can be improved if a visual sensor is added to it to give feedback.

Acknowledgements

This work has been funded by CONACYT scholarship 227266/212740.

References

1. **Garro, B. A., Sossa, H., & Vazquez, R. A. (2007).** Evolving ant colony system for optimizing path planning in mobile robots. *IEEE Conference on Electronics, Robotics and Automotive Mechanics,* pp. 444–449, doi: 10.1109/ICSPC.2007.4728416.
2. **Baerlocher, P. & Boulic, R. (1998).** Task-priority formulations for the kinematic control of highly redundant articulated structures. *IEEE International Conference on Intelligent Robots and Systems,* pp. 323–329.
3. **Brock, O. & Khatib, O. (2000).** Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. *International Conference on Robotics and Automation*, pp. 550–555, doi: 10.1109/ROBOT.2000.844111.
4. **Ma, C., Li, W., Yang, Y., & Chang, L. (1995).** Robot motion planning with many degrees of freedom. *IEEE International Conference on System, Man and Cybernetics,* Vol. 1, pp. 892–897, doi: 10.1109/ICSMC.1995.537880.
5. **Choi, J. & Amir, E. (2007).** Factor-guided motion planning for a robot arm. *IEEE International Conference on Intelligent Robots and Systems,* pp. 27–32, doi: 10.1109/IROS.2007.4399555.
6. **Ferguson, D., Kalra, N., & Stentz, A. (2006).** Replanning with RRTs. *IEEE International Conference on Robotics and Automation,* pp. 1243–1248, doi: 10.1109/ROBOT.2006.1641879.
7. **Plaku, E., Kavraki, L.E., & Vardi, M.Y. (2010).** Real-time inverse kinematics of the human arm. *IEEE Transactions on Robotics,* Vol. 26, No. 3, pp. 469–482.
8. **Ferguson, D. & Stentz, A. (2007).** Anytime, dynamic planning in high-dimensional search spaces. *IEEE International Conference on Robotics and Automation,* pp. 1310–1315, doi: 10.1109/ROBOT.2007.363166.
9. **Arechavaleta, G., Esteves, C., & Laumond, J.P. (2004).** Planning fine motions for a digital factotum. *IEEE International Conference on Intelligent Robotics and Systems,* Vol. 1, pp. 822–827.
10. **Gerke, M. (1999).** Genetic path planning for mobile robots. *American Control Conference,* Vol. 4, pp. 2424–2429, doi: 10.1109/ACC.1999.786483.
11. **Orozco, H.R., Ramos, F., Zaragoza, J., & Thalmann, D. (2007).** Avatars animation using reinforcement learning in 3d distributed dynamic virtual environments. *International Conference on Logic Applied to Technology (LAPTEC),* pp. 67–84 doi: 10.3233/978-1-58603-936-3-67.
12. **Kavraki, L.E., Svestka, P., Latombe, J.-C., & Overmars, M.H. (1996).** Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation,* Vol. 12, No. 4, pp. 566–580, doi: 10.1109/70.508439.
13. **LaValle, S.M. (1998).** *Rapidly-exploring random trees: A new tool for path planning.* Department of Computer Science, Iowa State University.
14. **LaValle, S.M. (2006).** *Planning Algorithms.* Cambridge University Press.
15. **Luttgens, K. & Hamilton, N. (2002).** *Kinesiology - Scientific Basis of Human Motion.* Mc Graw Hill.
16. **Kallmann, M., Aubel, A., Abaci, T., & Thalmann, D. (2003).** Planning collision-free reaching motions for interactive object manipulation and grasping. *Eurographics,* Vol. 22, No. 3, pp. 313–322, doi: 10.1111/1467-8659.00678.
17. **Mohamad, M.M., Taylor, N.K., & Dunningan, M.W. (2006).** Articulated robot motion planning using ant colony optimization. *IEEE Conference on*

ISSN 2007-9737

International Intelligent Systems, pp. 690–695, doi: 10.1109/IS.2006.348503.

18. **Uc, M., Rodriguez, A., & Ramos, F. (2007).** Reinforcement learning and dynamic planning applied to virtual humans animation. *4th International Conference on Electrical and Electronics Engineering*, pp. 169–172, doi: 10.1109/IS.2006.348503.
19. **Russell, S. & Norvig, P. (2002).** *Artificial Intelligence: A Modern Approach.* Prentice Hall.
20. **Tolani, D. & Badler, N.I. (1996).** Real-time inverse kinematics of the human arm. *Presence*, Vol. 5, No. 4, pp. 393–401.
21. **Koga, Y., Kondo, K., Kuffner, J., & Latmobe, J.C. (1994).** Planning motion with intentions. *International Conference on Computer Graphics and Interactive Techniques*, doi: 10.1145/192161.192266.
22. **Yoshida, E. (2005).** Humanoid motion planning using multi-level DOF exploitation based on randomized method. *IEEE International Conference on Intelligent Robots and Systems,* pp. 3378–3383, doi: 10.1109/IROS.2005.1544954.
23. **Arenas-Mena, J.C., Hayet, J.B., & Esteves, C. (2012).** A motion capture based Planner for virtual characters navigating in 3D environments. *Computación y Sistemas,* Vol. 16, No. 4.
24. **Choi, J. & Amir, E. (2009).** Combining Planning and Motion Planning. *IEEE International Conference on Robotics and Automation,* Kobe, Japan, doi: 10.1109/ROBOT.2009.5152872.
25. **Plaku, E. & Hager, G.D. (2010).** Sampling-based Motion and Symbolic Action Planning with Geometric and Differential Constraints. *IEEE International Conference on Robotics and Automation,* Alaska, USA, doi: 10.1109/ROBOT.2010.5509563.
26. **Ding, X. & Fang, C. (2013).** A Novel Method of Motion Planning for an Anthropomorphic Arm Based on Movement Primitives. *IEEE/ASME Transactions on Mechatronics,* Vol 18, No. 2, pp. 624–636, doi: 10.1109/TMECH.2012.2197405.
27. **Akgun, B. & Stilman, M. (2011).** Sampling Heuristics for Optimal Motion Planning in High Dimensions. *IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 2640–2645, California, USA, doi: 10.1109/IROS.2011.6095077.
28. **Islam, F., Nasir, J., Malik, U., Ayaz, Y., & Hasan, O. (2012).** RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution. *IEEE International Conference on Mechatronics and Automation,* Chengdu, China, doi: 10.1109/ICMA.2012.6284384.
29. **Xie, B., Zhao, J., & Liu, Y. (2011).** Human-like Motion Planning for Robotics Arm System. *International Conference on Advanced Robotics,* pp. 88–93, Tallinn, Estonia, doi: 10.1109/ICAR.2011.6088543.
30. **Zong, D., Li, C., Xia, S., & Wang, Z. (2012).** Planning interactive task for intelligent characters. *Computer Animation and Virtual Worlds,* Vol. 23, No. 6, pp. 547–55, doi: 10.1002/cav.1470.
31. **Zhang, L., Pan, J., & Manocha, D. (2009).** Motion Planning of Human-like Robots using Constrained Coordination. *IEEE International Conference on Humanoid Robots,* pp.188–195, Paris, France, doi: 10.1109/ICHR.2009.5379545.

Cristian E. Boyain y Goytia Luna received his M.Sc. from the Center for Research and Advanced Studies of the IPN (CINVESTAV), Guadalajara campus, in 2009. His research interests are motion planning, reinforcement learning, and computer animation.

Andres Mendez Vazquez received his Ph.D. from the University of Florida, Gainesville, USA in 2008. He is a research professor at the Center for Research and Advanced Studies of the IPN (CINVESTAV), Guadalajara campus. His research interests include machine learning and data mining, artificial intelligence, computer vision, analysis of algorithms, and numerical optimization.

Marco Antonio Ramos Corchado received his Ph.D. from the University of Toulouse, France. He is a research professor of Artificial Intelligence and Virtual Reality at the Autonomous University of Mexico State. His research interests include animation techniques, artificial life, artificial intelligence, distributed systems, and intelligent agents.

Article received on 05/12/2014; accepted on 09/04/2015.
Corresponding author is Cristian E. Boyain y Goytia Luna.