Using Particle Filters to Find Free Obstacle Trajectories for a Kinematic Chain

Alejandro REYES-AMARO[1], Alejandro MESEJO-CHIONG[1], Ramon MAS-SANSÓ[2], Antoni JAUME-I-CAPÓ[2]

[1] Facultad de Matemática y Computación, Universidad de la Habana, Cuba, aamaro@matcom.uh.cu, mesejo@matcom.uh.cu

[2] Departament de Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears, Spain, ramon.mas@uib.es, antoni.jaume@uib.es

Abstract: The problem of finding an appropriate path for a mechanical arm that tries to reach a target among obstacles is one of the most important in fields of automation and robotics. It is both a classic inverse kinematics and collision detection problem. This project aimed to construct a tool to plan a path for an articulated arm through a two-dimensional environment with obstacles. The inverse kinematics problem is addressed by heuristics Bayesian particles filter, and the collision detection problem is solved using computational geometry methods for calculating the free configurations space. The proposed tool has a graphical interface with which you can get information from the designed experiments. The feasibility of this approach and its advantages in complex two-dimensional environments is shown. We proved that good results can be obtained with an appropriate selection of the parameters.

Keywords: Inverse kinematics, particle filters, configuration-free space, path planning.

1. Introduction

Many automated processes today require to solve the problem of planning a set of movements for an articulated arm with the objective of reaching a specific goal and avoiding obstacles without using neither sensors nor other technologies [15], but knowing its spatial information. These articulated arms are formally designated as *kinematic chains*. A Kinematic chain is formed by a set of rigid bodies (*links or segments*) joined at the ends by articulations that allow for certain types of movement, mainly translation, rotation or revolution. One end of the chain remains fixed whilst the other end reaches the defined objective; the latter is called *end-effector*.

Using a geometrical approach we can easily compute the position and orientation of a given segment from the values of the preceding articulations (direct or forward kinematics). Let us say that in an articulated chain as the human arm we can easily compute the situation of a fingertip given the angles of all the articulations from the clavicle to the finger. However, the inversion of this model (inverse kinematics) becomes difficult due to the non-linearity of the governing equations. Let suppose that we want to infer the values of the angles *configuration space* that will locate our end-effector in a given position (situation space).

The problem of controlling a kinematic chain has been first posed in robotics [17] although it has also been widely used in computer graphics [2, 6]. Robotic structures are most of the times well-known mechanisms with a moderated number of degrees of freedom so it is relatively easy to derive an analytical solution. Nevertheless, more complex kinematic chains face the additional problem of being under constraint or redundant -i.e. they contain more *d.o.f.* than required for a class of tasks. In this case, instead of using a closed-form or an analytical solution we should use a more general approach for the positioning and manipulation of kinematic chains like resolved motion rate control. Resolved motion rate control is an Inverse Kinematics technique based on the inversion of the Jacobian matrix. This approach allows the manipulation of an articulated figure with a relatively low computational cost and using intuitive specifications, however it also has known drawbacks as singularities and local minima solutions.

The problem becomes even more complex when obstacles have to be avoided. Several solutions have been proposed in the literature using a broad variety of techniques like direct kinematics [14], neural networks [17], genetic algorithms [1] or heuristic solutions [8, 16] or solving optimization problems [13].

We work in rehabilitation environments where the user is required to reach a goal avoiding virtual obstacles. The arm of the user is modeled with a kinematic chain and we want to automatically compute the optimal path to perform such a task.

In such a context, singularities a local minima have to be prevented [4, 5]. We propose to solve inverse kinematics by means of extending heuristic solutions based on particle filter techniques combined with the computation of trajectories among obstacles. We want to prove its feasibility in complex two-dimensional situations in a controlled development environment.

The rest of this paper is organized as follows. The problem definition is stated in section 2. The section 3 shows how to use the particle filters to solve the inverse kinematic problem. Section 4 describes the stages that a kinematic chain goes through. The method to calculate the trajectory of the end-effector is presented in section 5. In section 6 experimental results are discussed and the section 7 is reserved to present the conclusions of the work.

2. Problem Definition

In order to facilitate the understanding of the issue, some concepts must first be defined. Figure 1 shows a mechanical arm or bidimensional kinematic chain, consisting of three links. It is referred to as $\mathcal{R}$, and is formed by a base jointx_0, considered to be fixed, a set of intermediate joints x_1, x_2 and the end-effector x_e. The position of a joint within two-dimensional Euclidian space is denoted as x_i. It is therefore true that$\{x_0, x_1, x_2 \dots, x_e\} \in \mathbb{R}^2$. The joint$x_e$ shall reach region *G*, the objective. The links are denoted l_0, l_1 and l_2; since a link is between two joints, its number is one unit less than the corresponding joint number. The link l_i is attached to base jointx_i, around which it is free to rotate.

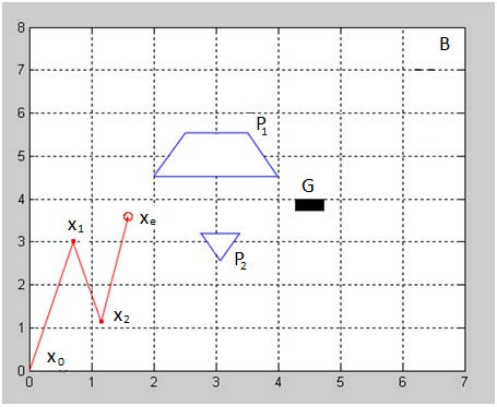


Figure 1. Workspace.

The region of interest for the mechanical arm is called environment, and it is denoted as $\mathcal{B}$. It shall be considered, without loss of generality, that $\mathcal{B} = [a_x, a_y] \times [b_x, b_y] \subset \mathbb{R}^2$, where $a_x < 0$ and $b_x < 0$. Hence $\mathcal{B}$ is a rectangular region in $\mathbb{R}^2$, parallel to the axis of coordinates and including the origin. The set of obstacles is called the obstacle system and it is denoted as *S*. The obstacle system is defined by a set of polygons$S = \{P_1, \dots, P_M\}$ verifying $P_i \subset \mathcal{B}$. The free configuration space is therefore the set $\mathcal{B} \setminus S = \mathcal{B} \setminus \cup_{i=1}^{n} P_i$.

Using the previous notations we can formally define the problem:

Problem 1: Given $\{\mathcal{B}, S, G, \mathcal{R}_0\}$, where $\mathcal{R}_0 = \{\Theta_0\}$ is an initial configuration, we want to find a sequence of configurations $T = \{\mathcal{R}_0, \mathcal{R}_1 \dots, \mathcal{R}_m\}$ for the kinematic chain verifying that $\mathcal{R}_i \cap S \neq \emptyset$ and $\mathcal{R}_m \ni x_{e_m} \cap G \neq \phi$.

The notation $\mathcal{R}_i \cap S \neq \phi$ denotes that for all $j = 0, \dots, n-1$ the link l_j from configuration $\mathcal{R}_i$ complies with $P_k \cap l_j = \emptyset$ for all $j = 1, \dots, M$.

2.1 The inverse kinematic problem

The mathematical model for the general inverse kinematics approach can also include additional optimization criteria using the components of the homogeneous solution:

$$\Delta\theta = J^{+}\Delta X + \alpha(I - J^{+}J)\Delta z$$

where

- $\Delta\theta$ is the state difference vector in the joint variation space, of dimension *n*.
- ΔXdescribes the so-called *main task* (or *behavior*) as a variation of one or more end effector(s) position and/or orientation in cartesian space. Its dimension is *m*.
- $J$is the Jacobian matrix of the linear transformation representing the first order approximation of the direct geometric model for the *main task.*
- $J^{+}$is the unique pseudo-inverse of J providing the minimum norm solution achieving the *main task.*
- $I$is the $n \times n$ identity matrix of the joint variation space.
- $(I - J^{+}J)$is a projection operator on the *null space* of the linear transformation J.
- Δzdescribes a *secondary task* (or *behavior*) in the joint variation space. Its projection on the null space constitutes the homogeneous solution that is mapped by *J*

into the null vector of the cartesian variation space, thus not affecting the realization of the main task.

- α is a constant gain.

The first term of this equation is usually known as the *pseudo-inverse solution* and the second term is called the *homogeneous solution*.

By definition the secondary task (or behavior) is partially performed by its projection on the null space. In this way, the projected component does not modify the achievement of the main behavior because it is mapped into the null vector of the cartesian variation space by the linear transformation *J*. The secondary task usually expresses the minimization of a cost function and it is important to evaluate the potential of this optimization to succeed.

3. Particle Filters as a Solution to the Inverse Kinematics Problem

Particle filters methods constitute an efficient way to solve the inverse kinematics problem avoiding the calculation of the inverse of the Jacobean or its approximation. Particle filters [10] are methods used to estimate the state of a system at a given time. This technique is used to heuristically solve the inverse kinematics problem.

The main advantages of particle filters are that they don't require numerical inversion and they allow to easily incorporate any type of additional restriction, provided an evaluation function is supplied.

Particle filters can be defined, generally, as a function applied to a set of particles, with a defined objective. Hereafter, the definition of particle in the context of this work is proposed:

Definition 1. A particle associated to a certain kinematic chain $\mathcal{R}$, consisting of *n* links, is described by a set of points $\{x_1, x_2, \dots, x_n\}c \subset \mathbb{R}^2$ that can be interpreted as the set of positions of the joints of $\mathcal{R}$; or else by a vector $\Theta = \{\omega_0, \omega_1, \dots, \omega_{n-1}\} \in [0,2\pi[^n$that can be interpreted as the set of angles formed by the links of chain $\mathcal{R}$.

In the present paper work, both representations are used interchangeably. The first one is used in order to graphically represent the results, and the second one is used in the generation process of configurations, and they are modeled by quaternions, in order to make the implementation of the rotations more efficient.

Using an extension of the Euler's theorem, the following expression can be obtained:

$$e^{\theta\hat{n}} = (\cos\theta, \sin\theta\vec{n})$$

Being $\hat{n}$ a pure unit quaternion ($\hat{n}^2 = -1$) and $\vec{n}$ an axis. Then, we can deduce that any quaternion can also be represented in a polar form as $q = |q|e^{\theta\hat{n}}$.

In that way we can say that if $q = |q|e^{\theta\hat{n}}$ is a pure quaternion, then the transformation $T_q(\vec{x}) = q \times q^{-1}$ produces a rotation of the vector $\vec{x}$ around the axis $\vec{n}$ with an angle 2θ [11].

A transformation of this type is represented by:

$$R(\theta, \hat{n}) = \left(\cos\frac{\theta}{2} + \hat{\text{n}}\,\sin\frac{\theta}{2}\right)$$

It is used to describe a rotation with angle θ in the plane perpendicular to $\hat{n}$.

4. Operation of a Particle Filter

It is worth noting that a particle defines a randomly generated configuration of the mechanical arm. Its operation is based on generating a set of particles and interactively selecting the most adequate for a certain objective. In this case it is necessary to know which particle, or set of particles, approach the objective in a feasible way, so as to solve the problem of the inverse kinematics.

During the process of filtering [8], the system of particles undergoes five different stages:

1. *Initialization*: the particles are generated regardless of any prior information about the system. The weights of each particle, which measure their proximity with the final goal, are assigned with the same value, because at the beginning, all particles are equally relevant. Let *N* be the number of particles to generate and let denote by $\mathcal{R}_0^{(i)}, i = 1 \dots N$, the i^{th} particle of the initial generation and $\omega_0^{(i)}$ the weight of $\mathcal{R}_0^{(i)}$. The particles $\mathcal{R}_0^{(i)}$ are generated only with the information provided by the initial position of the kinematic chain. This means that the poses $\mathcal{R}_0^{(i)}$ are "similar" to the initial given $\mathcal{R}_0$.
2. *Exploration*: At each iteration *k* of a particle filter, each particle $\mathcal{R}_{k-1}^{(i)}$ is updated in response to some criterion and generates the $\mathcal{R}_k^{(i)}$. For that we need to know certain

properties of particles already generated: the particle $\mathcal{R}_k^{(i)}$ is generated taking into account theproximity to the target and the feasibility of the particle $\mathcal{R}_{k-1}^{(i)}$.

3. *Weight calculation*: The weights are updated for the new generated particles. The first factor to be considered is the proximity of the end-effector pose $\mathcal{R}_k^{(i)}$ to the target. Another key that should be respected is the inclusion of the particle in the free configuration space. This criteria will be discussed in detail later.
4. *Position estimation*: The lighter particles and the not-fully ones included in the free configuration space are removed and replaced cloning the others.
5. *Mutation/Selection*: The objective of this phase is to remove weakly standardized weights and to increase the number of particles associated with high weights. This can only be done when the number of the significant particles is small, so this step is not always necessary in the iterative process.

This process is repeated until at least one of the generated particles in the set, makes contact with the final goal; and it guarantees that, once satisfactorily finished, the end-effector will be in contact with the objective, hence solving the inverse kinematics problem.

4.1 Generation of feasible particles

As explained previously, the weight of a particle depends on how close the end-effector is in relation to the pose the particles generate with the objective. There is, however, a physical restriction to avoid contact of the pose with the obstacles presents in the work environment.

A feasible particle is defined as any particle that generates a pose of the chain that does not collide with the system of obstacles. During the evolution of the particle filter and, specifically, during the weight calculation phase, it becomes necessary to verify if a particle is feasible or not. To do this, a *Minkowskisum* [9] is applied, as it is a simple technique to calculate the intersection of the chain with the obstacles.

4.1.1 Minkowski sum

A *Minkowskisum* [9] is a set operation used to calculate the region of the plane in which two polygons meet with a non-empty intersection. Given that the objective is to decide if a certain configuration for the mechanical arm is valid or not, that is, if it overlaps with any of the obstacles, the *Minkowskisum* can be used. To this end, once the non-empty obstacle intersection area is calculated, it is asked whether any part of the mechanical arm is within that area.

To do this, the arm is divided according to its links and the non-empty intersection space for each one is calculated. The free configuration space will be the complement of the union of the resulting spaces.

To apply the *Minkowskisum* to the link, we assign a representing point and we compute the region where that point is not included in any obstacle. This way, the problem of calculating a segment-obstacle collision is reduced to a problem of point-polygon inclusion.

In the weight calculation phase we compute $fac\left(x_k^{(i)}\right)$ so that:

$$fac\left(x_k^{(i)}\right) = \begin{cases} 0 & if\ \exists\, P_i \in S: x_k^{(i)} \cap P_i \neq \emptyset \\ 1 & otherwise \end{cases}$$

And then $w_k^{(i)} = fac\left(x_k^{(i)}\right) w_k^{(i)}$. During the mutation/selection phase, the generation algorithm discards any zero weighted particles, and generates a replacing one that will be subsequently verified.

5. Trajectory Calculation for the End-effector

It is convenient to previously construct a valid trajectory for the end-effector to follow, one that guarantees an obstacle free path. At the same time, it is necessary to verify that all the links in the chain are located in the free configuration space.

5.1 Trapezoidal map of a system of obstacles

We have previously presented an algorithm that guarantees that the end-effector of the mechanical arm reaches its objective, by defining partial objectives and attaining them in a successive manner.

The free configuration space, denoted as $C_{free}(R,S)$, is defined as

$$C_{free}(R,S) = B - \bigcup_{i=1}^{t} S_i$$

If we continuously check that the kinematic chain is within this area, then it is guaranteed that there will be no collision with the system of obstacles. We need an algorithm to calculate this space. We use a set of trapezoids that determine the space free of collision in the system and allow the construction of a trajectory map amongst the obstacles. This set, consisting of the union of the trapezoids, is called the trapezoidal map of the system of obstacles.

5.2 Trajectory calculation for the end-effector

As an initial approximation, the end-effector is placed on the first trapezoid of the free configuration that area of the map. This is the starting node for the graph of plausible paths of the end-effector.

If the point objective is within the initial trapezoid, the path required is trivial: a straight line that joins the end-effector with the objective. When this is not the case, a graph node must be created in the centre of each trapezoid; then a graph node must also be created in the centre of each vertical line of the trapezoid. Thereafter, an arch between two nodes is created, when one of the nodes is in the centre of the trapezoid and the other on the frontier line. Finally, an arch is created from the centre point of the trapezoid, containing the point objective, to the point objective itself.

The end-effector has now a path to follow, which guarantees that it will not collide with any obstacle. However, it cannot be assured that the mechanical arm as a whole will not do so. Each movement must therefore verify if the chain is within a free configuration area.

5.3 Optimal trajectory calculation for the end-effector

The path generated by the trajectory construction algorithm may lead the intermediate nodes to positions that are unreachable by the end-effector as the arm would collide with an obstacle or the intermediate objective would be positioned too far.

To alleviate this effect, the calculation of an optimal trajectory is proposed. This trajectory shall follow the end-effector in its path to the objective. For the optimal trajectory, each intermediate node must be located above the vertical extension of each trapezoid on the trapezoidal map, or on the same x coordinate of that chosen to situate it when the node is within a trapezoid. The aim is to find an adequate translation on the y axis for each intermediate node. This way, it is assured that the end-effector will not collide with any obstacle in the system.

The following optimization problem arises:

$$min: \sqrt{(\Delta y_{0,1})^2 + (\Delta x_{0,1})^2} + \sqrt{(\Delta y_{1,2})^2 + (\Delta x_{1,2})^2} + \cdots + \sqrt{(\Delta y_{n-1,n})^2 + (\Delta x_{n-1,n})^2}$$

$$s.t.: y_{i_{min}} < y_i < y_{i_{max}} \quad (1)$$

$$where: \quad \Delta_{y_{i,i+1}} = y_i - y_{i+1}, \quad \Delta_{x_{i,i+1}} = x_i - x_{i+1}$$

This problem is solved for each generated path, this depending solely on the amount of obstacles in the system. Efficient tools already generated and tested with good results are used [3, 7, 12].

5.3 Partitioning the path

Once the paths are calculated for the end-effector, either optimal or alternative paths, it is necessary to partition them as finely as possible. This way the particle generation algorithm is able to work efficiently.

Once a trajectory map is obtained we create a series of nodes, separated by a set distance and controlled from the application, between each pair of nodes.

6. General Outline of the Solution

In this section we propose a general outline of the algorithm, but first we should define some used functions and operators.

In the section 2 was defined the quatrain $\{\mathcal{B}, S, G, \mathcal{R}_0\}$. It is the input of the algorithm. The following functions are defined as:

$Trap(S)$: Returns the trapezoidal map of the obstacle's system*S*.

$Traj(T)$: Returns a graph that represents the set of possible trajectories to be followed by the end-effector of the kinematic chain, starting by a given trapezoidal map.

$Opt(t, T)$: Calculates the minimum length trajectory starting in *t* and taking into account the *T*trapezoidal map, using (1).

$add(D, \Theta, t)$: Adds to a *D* dictionary the Θ pose as one of poses that should be adoptedin the path through the *t* trajectory. In our project,

besides to do this, we represent the configuration graphically.

$ee(\Theta)$: Returns the end-effector of the Θ configuration.

$mark(\mathrm{t}, \mathrm{true})$: Labels the t trajectory as the one possible to be followed.

$IK(\Theta, nod, S)$: It is the most important function in this algorithm. It is in charge to obtain the configuration, starting from the initial Θ, in which one the end-effector gets in touch with the intermediate node *nod*, and it cannot be in touch with any obstacle in the *S* system.

The algorithm:

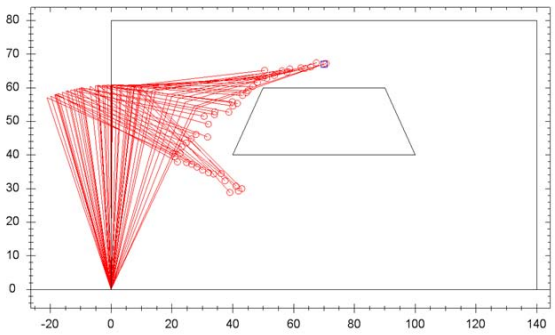
```
Input:{B,S,G,R0}
Output:
Dictionary<Trajectory,List<Configuration>> D
Set of configurations for each trajectory
through the final goal.
1-Ts = Trap(S);
2-TM = Traj(Ts);
3-foreach Trajectory t in TMdo
4-   topt = Opt(t,Ts);
5-   □ = R0;
6-   add(D,□,t);
7-   foreach Node nod in toptdo
8-      □ = IK(□,nod,S);
9-   mark(t,|ee(θ) − nod| < ε);
```

7. Results

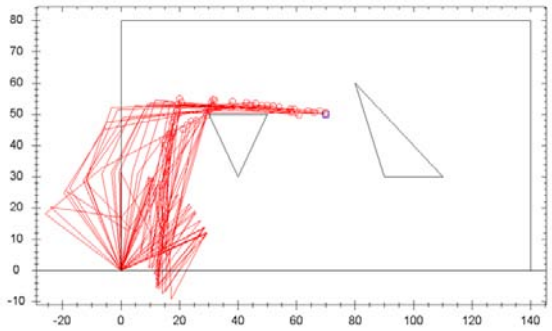
We have designed an application to plan the movement of a kinematic chain towards a target in a 2D environment with obstacles. It includes a tool that shows, before running the program for a certain configuration (position and characteristics of the kinematic chain, the obstacles and the target), information concerning the pre-processing: the graph of the initial path which is calculated from the trapezoidal map of the system of obstacles, the graph of the computed optimal path followed by the end-effector on its way to the target and the trapezoidal map of the system of obstacles.

The obtained optimal path does not ensure that the arm can adopt a configuration in which the end-effector gets in contact with each intermediate target. Our algorithm follows three steps:

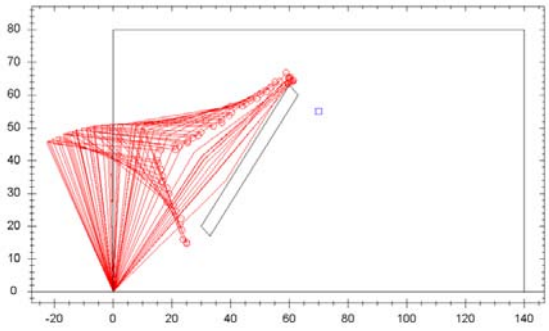
- First, we use the particle filter to reach intermediate target *K*
- In case the final effector is located close enough to the target (regulated by an epsilon parameter) or a maximum of iterations is performed, then this stage is over and a new one is started for the next intermediate target $K+1$
- In a final step, where the final target must be reached, the application forces the filter to apply a maximum of 50 iterations and tries to approach the target to a distance closer than a given epsilon.



(a) Example 1 - Reachable



(b) Example 2 - Reachable



(c) Example 3 - Unreachable

Figure 2. Poses sequence of a kinematic chain trying to reach a goal among obstacles.

The experiments we have performed proof that is feasible to solve the problem of finding a path for a kinematic chain in a 2D environment with obstacles using particle filters (see Figure 2). We have prepared seven test examples, four of them with unreachable goals, and three with the goals in the scope of the arm. We thought it might be interesting, as they can show the behaviour of an arm trying to reach a goal and not achieving it because of its physical

limitations. They evidence in a practical way that the proposed algorithm ensures that the arm does not collide with any obstacle, even if some of them prevent it from reaching its target, as is the case of Figure 2(c).

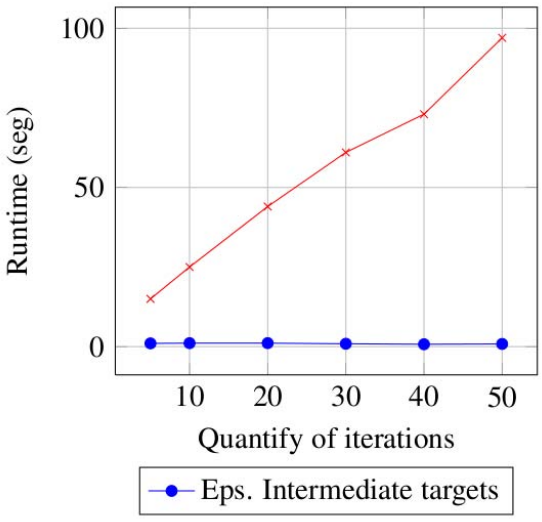
For each of the experiments we performed more than 30 runs of the software combining several characteristics:

- Between 5 and 30 particles for the filter
- Approach tolerance to intermediate targets between 10^{-1} and 10^{-6}
- Between 5 and 50 iterations for each filtering stage.
- Sigma parameter (σ) used for the generation of particles generated according to a normal probability distribution, between 0.01 and 0.035.

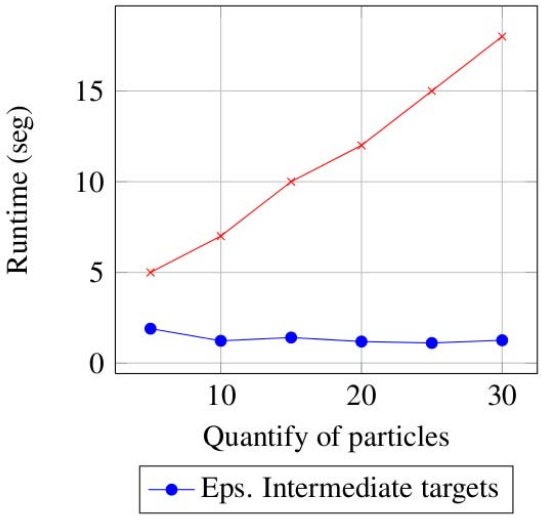
Figures 3(a), 3(b), 3(c) and 3(d) show the results of the runs performed using the configuration presented on Figure 2(a). This example was chosen because it was designed so that its end-effector could reach each intermediate target. The Figure 3(a) shows, that the higher the amount of iteration of the particles in the filter, the bigger the execution time; while the end-effector's proximity remains almost invariant. For that reason the following experiments were performed using as a maximum five iterations. We obtained both good results and low execution times.

Figure 3(b) shows that the number of particles used in the filter also played an important role in the runtime. Although the proximity of the end-effector to the calculated optimum trajectory remained invariant we used 25 particles in the filter to keep the probability of generating the particles in a collision space low.

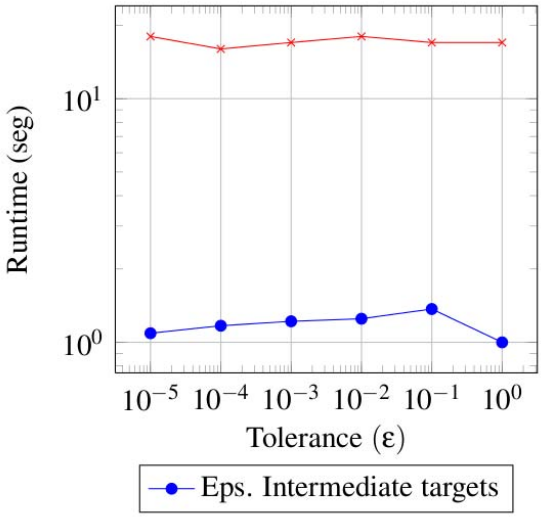
In the performed experiments with the end-effector tolerance to follow the calculated optimum trajectory (Figure 3(c)), an expected behaviour could be observed: run time did not vary. This occurred because the Euclidian distance between the end-effectors of each generated particle is in the range of 1px to 10px. For that reason, demanding the end-effector to get closer to the trajectory would imply to force the filter to perform too many iterations. If the mechanical arm follows the calculated trajectory in an acceptable way, then it is not necessary to constrain that parameter.



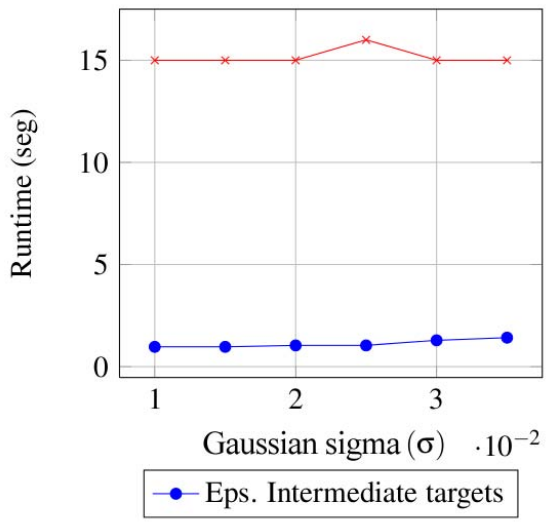
(a) Quantify of particles used in the filter



(b) Quantify of particles used in the filter



(c) Tolerance to follow the calculated optimum trajectory



(d) Gaussian mean (σ)

Figure 3. The graphs show how intervenes each parameter in the runtime.

Figure 3(d) shows the performed experiments with the sigma parameter (Gaussian distribution variance). Sigma controls the similarity of the generated particles. The results prove that the chosen sigma parameter makes the execution time invariant if, and only if, it remains within the range indicated in the graphic. Performed runs with parameters far from this range, showed non-expected results.

8. Conclusions

The results show that the particle filter technique is effective for solving the inverse kinematics problem. It can be also be noted that the method applied to track a valid trajectory for the end-effector, guarantees that the mechanical arm will not collide with any obstacles and that the end-effector reaches its goal when possible.

The performed experiments proved that it is possible to obtain good results with an appropriate selection of the parameters, and to improve the runtime. It was demonstrated that the choice of the parameters quantify of particles and sigma found in the literature, is the most stable in order to obtain coherent results. The others parameters were adjusted taking into account the runtime, because it was demonstrated that if they are selected from the inside of the proposed intervals, the mechanical arm will follow the optimal calculated trajectory.

There still remain some questions to be addressed. Our goal is to apply this method in a three dimensional space to reconstruct human movement using inverse kinematics and computer vision.

Acknowledgements

This work is partially supported by the projects MAEC-AECID A/030033/10 and MAEC-AECID A2/037538/11 of the Spanish Government.

REFERENCES

1. ABO-HAMMOUR Z. S., A. G. ASASFEH, A. M. AL-SMADI, O. M. K. ALSMADI, **A Novel Continuous Genetic Algorithm for the Solution of Optimal Control Problems.** Optim Control Applied Methods, vol. 32, 2011, pp. 414-432.
2. BAERLOCHER, P., **An Inverse Kinematic Architecture Enforcing an Arbitrary Number of Strict Priority Levels**. Visual Computation, vol. 20(6), 2004, pp. 402-417.
3. BIRD, R. H., P. LU, J. NOCEDAL, C. ZHU, **A Limited Memory Algorithm for Bound Constrained Optimization**. Technical Report NAM-08 1994.
4. BUSS, S. R.: **Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods**. Department of Mathematics, University of California San Diego USA, 2004.
5. BUSS, S. R., J. S. KIM, **Selectively Damped Least Squares for Inverse Kinematics**. Department of Mathematics Department of Computer Science, University of California San Diego, USA, 2004.
6. CARY, P. B., J. ZHAO, N. I. BADLER, **Interactive Real-time Articulated Figure Manipulation using Multiple Kinematic Constraints**. SIGGRAPH Comput Graph, vol. 24(2), 1990, pp. 245-250.
7. CORDERO, Y., **OCSolv: un sistema con estrategia adaptativa para Problemas de Control Optimal**. Facultad de Matemática y Computación, Universidad de La Habana La Habana, Cuba, 2010.
8. COURTY, N., E. ARNAUD, **Sequential Monte Carlo Inverse Kinematics**. INRIA Rapport de recherche 2007, p. 6426.

9. DE-BERS, M., M. VAN-KREVEL, M. OVERMARS, O. SCHUARZKOPF, **Computational Geometry Algorithms and Application**, 2nd edition; 2000.

10. DOUCET, A., A. M. JOHANSEN, **Tutorial on Particle Filtering and Smoothing: Fifteen Years Later.** Institute of statistical mathematics, Tokyo, Japan Department of Statistic, University of Warwick, UK, 2008.

11. JOHNSON, M. P., **Exploiting Quaternions to Support Expressive Isteractive Character Motion.** Massachusetts Institute of Technology USA PhD Thesis, 2003.

12. LIU, D. C., J. NOCEDAL, **On the Limited Memory BFGS Method for Large Scale Optimization**. Mathematical Programming, vol. 45, 1989, pp. 503-528.

13. PORTILLO-VELEZ, R. D. J., C. A. CRUZ-VILLAR, A. RODRIGUEZ-ANGELES, **On-line Master/Slave Robot System Synchronization with Obstacle Avoidance**. Studies in Informatics and Control, ISSN 1220-1766, vol. 21(1), 2012, pp. 17-26.

14. SCIAVICCO, L., B. SICILIANO, **A Solution Algorithm to the Inverse Kinematic Problem for Redundant Manipulators**. Robotics and Automation, vol. 4(4), 1988, pp. 403-410.

15. SUSNEA, I., VASILIU, G., **On Using Passive RFID Tags to Control Robots for Path Following**. Studies in Informatics and Control, ISSN 1220-1766, vol. 20(2), 2011, pp. 157-162.

16. SUSNEA, I., G. VASILIU, A. FILIPESCU, A. RADASCHIN, **Virtual Pheromones for Real-Time Control of Autonomous Mobile Robots**. Studies in Informatics and Control, ISSN 1220-1766, vol. 18(3), 2009, pp. 233-240.

17. ZHANG, Y., J. WANG, **Obstacle Avoidance for Kinematically Redundant Manipulators using a Dual Neural Network**. Systems, Man, and Cybernetics, Part B: Cybernetics, 34(1), 2004, pp. 752- 759.