Mecánica Computacional Vol. XXII
M. B. Rosales, V. H. Cortínez y D. V. Bambill (Editores)
Bahía Blanca, Argentina, Noviembre 2003

SOFTWARE AND ALGORITHMS FOR MOTION PLANNING IN VIRTUAL ENVIRONMENTS

Omar A.A. Orqueda

Grupo de Robótica y Simulación - Departamento de Electrotecnia
Universidad Tecnológica Nacional, Facultad Regional Bahía Blanca
11 de abril 461, B8000LMI Bahía Blanca, Argentina
e-mail: orqueda@ieee.org, web page: http://www.frbb.utn.edu.ar

Key Words: Motion Planning, Virtual Environments, Nonholonomic Planning.

Abstract. *Motion planning in virtual environments is an exciting research field that has been studied during last three decades. It has evolved from the simplest problem of planning a 2D path of a mobile robot using geometric methods, to the very difficult problem of planning 3D motions of multiple robots with many degrees of freedom, or, more surprisingly, graphic animation, surgical planning, computational biology, or automatic sensing. This work presents a generic software package,* MPSLab, *that is aimed for motion planning and simulation of robots with multiple degrees of freedom (* dof*) in virtual environments. This package can represent and simulate multiple robots with several dofs moving among fixed polyhedral obstacles. The motion planning algorithms implemented in this packages are* probabilistic RRT*-based algorithms, some of them developed by the author of this work. With the motion planning algorithms, problems involving holonomic, non holonomic, and generic constraints can be solved. To implement these algorithms, the package possesses differential-equation solvers, optimization algorithms, collision detection/distance computation libraries, and fast computation of artificial potential fields.*

1 INTRODUCTION

The *Motion Planning Problem* (*MPP*)[1] have attracted the attention of many researchers during the last three decades. With the increase of computational power capability of today's computer, there exists the possibility of giving more autonomy to any system, if not full autonomy.

In its simpler form, the *MPP* consists in finding a collision-free path for a robot among rigid static obstacles connecting an initial state with a final desired one. The basic *MPP* has been traditionally solved using geometric approaches, but this problem is computationally hard for robots with several degrees of freedom, *i.e.*, more than five.

In 1969, Nilsson described a mobile robot system with motion planning capabilities using a visibility graph method combined with the A^* search algorithm to find the shortest collision-free path.[1] The robot was represented by a point amidst polygonal obstacles. Udupa solved collision avoidance by introducing the idea of shrinking a robot to a point.[2] In 1979, Lozano-Pérez and Wesley introduced the concept of *configuration space* and solved the *MPP* for polygonal or polyhedral robots translating among polygonal or polyhedral obstacles.[3,4] This year, Reif showed that path planning for a $3D$ linkage made of polyhedral links is *PSPACE*-hard and that there is strong evidence that any complete planner[2] requires time that grows exponentially with the number of *dof* of the robot to find a solution.[5–7]

The complexity of complete path planners and their lack of robustness have motivated the development of *heuristic planners*. Two of the most popular approaches are *approximate cell decomposition* and *artificial potential field*.[7–11] In the approximate cell decomposition, the free space is represented by a collection of simple cells, whereas artificial potential fields are used to move the robot under the local effects of repulsive fields associated to obstacles and the attractive field pulling toward the goal.

Both approaches are resolution-complete[3] and can solve complex path planning problems in $2D$ and $3D$ configuration spaces, but none of these approaches extends well to robots with more than 3 dofs, because the number of cells becomes too large, or the potential field can stuck the robot into local minima before reaching the goal position.

In 1991, a randomized planner which alternated *down motions* to track the negated gradient of a potential field and *random motions* to escape local minima was introduced.[12] This planner was able to solve complex path planning problems for many-dof robots. Because the problems of local minima caused by a deterministic potential field persist, another type of randomized planner was developed. This planner consists of sampling the configuration space at random and connecting the samples in free space by *local* paths,

[1] *Motion planning* is a general term that refers to either path planning or trajectory planning.

[2] A *complete*, or *exact*, path planner is one which returns a collision-free path whenever one exists, and indicates that no such path exists otherwise.

[3] A resolution-complete planner is one which finds a path if it exists and if the resolution parameter, the size of the smallest cells or the resolution of the grid, is set fine enough.

typically straight paths, thus creating a probabilistic roadmap (PRM)[4].[13–16] Experiments with PRM planners have been quite successful, showing that they are both fast and reliable even with robots with many dofs. Formal analysis supports this experimental observation by showing that PRM planning is complete in a probabilistic sense[5].[17] PRM planners are also robust to floating-point approximations and easy to implement. A number of variants applying different sampling strategies have been recently developed.[17–21]

Most of PRM-based algorithms are checked for collision using fast collision checkers,[22–34] because, despite of some attempts to compute an explicit representation of the free space,[35] it has been shown that this computation is computationally prohibitive.

With the introduction of differential constraints, a challenging problem emerges that involves both nonlinear control and traditional path planning issues. This problem is often referred to as *nonholonomic planning*,[36–39] or kinodynamic planning.[40–42] The design of a roadmap-based algorithm is more challenging because of the increased difficulty of connecting pairs of states in the presence of the constraints, also referred to as the *steering problem*.[36] Several randomized approaches to kinodynamic planning appeared based on *Rapidly-exploring Random Trees* for static and time-varying environments, nonholonomic and dynamic constraints, optimization criteria, moving obstacles, and/or flexible robots.[43–46]

In this work, it is presented a novel software package for motion planning and simulation of robots with multiple degrees of freedom (dof) in virtual environments, *MPSLab*. This package can represent and simulate multiple robots with several dofs moving between fixed polyhedral obstacles. The motion planning algorithms implemented in this packages are *probabilistic* RRT-based algorithms, some of them developed by the author of this work.[11,43,44,46,47] This motion planning algorithms can solve *MPPs* involving holonomic, nonholonomic, and generic constraints. It possess several differential-equation solvers and optimization algorithms, collision detection/distance computation libraries, and fast computation of artificial potential fields.

The work is organized a follows: In Section 2, some mathematical expressions and notions on constraints are resumed. In Section 3, the Motion Planning Problem is formalized. In Section 4, the main characteristics of the software package *MPSLab* is introduced. In Section 5, new fields of application are briefly discussed. Conclusions, future work, and references close the work.

2 MATHEMATICAL PRELIMINARIES

Let $\mathcal{W}$ denote the *world space*, or the physical space in which robots and obstacles exist. Using the configuration space concept, a robot, $\mathcal{A}$, is represented as a point, q, called a *configuration*, in a parameter space encoding the robot's dofs, the configuration space,

[4] A *roadmap* is a network of collision-free paths that captures the configuration-space topology, and is generated by preprocessing the configuration space independently of any initial-goal query.

[5] Under reasonable geometric assumptions on the free space, the probability that a PRM planner fails to find a path while one exists decreases exponentially toward 0 with the number of samples.

$\mathcal{C}$. The obstacles in the workspace, $\mathcal{B}_i$, $i = 1, 2, \ldots$, map as forbidden regions into the configuration space, $\mathcal{CB}_i = \{q \in \mathcal{C} \mid \mathcal{A}(q) \bigcap \mathcal{B}_i \neq \emptyset\}$, or $\mathcal{C}$-obstacle. The union of all the $\mathcal{C}$-obstacles in $\mathcal{C}$ is called the *C-obstacle region*. The complement of the $\mathcal{C}-$obstacle region is the free space, $\mathcal{C}_{free}$. Path planning for a dimensioned robot is thus *reduced* to the problem of planning a path for a point in a space that has as many dimensions as the robot has dofs.[7]

Let $\mathcal{C}_{valid} \in \mathcal{C}$ denote the subset of valid postures of the robot $\mathcal{A}$. Let $\tau : \mathcal{I} \longmapsto \mathcal{C}$ denote a *motion trajectory* or *path* for $\mathcal{A}$ expressed as a function of time, where $\mathcal{I}$ is an interval $[t_0, t_1]$. $\tau(t)$ represents the configuration q of $\mathcal{A}$ at time t, with $t \in \mathcal{I}$. A trajectory τ is said to be *collision-free* if $\tau(t) \in \mathcal{C}_{free}$ for all $t \in \mathcal{I}$.

The differential equations describing a nonlinear system affine in control that represents the motion of $\mathcal{A}$ can be described in state space form by[6]:

$$\dot{x} = f_0(x) + \sum_{i=1}^{m} f_i(x)\, u_i = f_0(x) + f(x)\, u \tag{1}$$

In the general case, the *state variable* x evolves over a real-analytic, connected manifold $\mathcal{M} \in \mathbb{R}^n$, with $q \in \mathcal{C} \subseteq \mathcal{M}$, the controls u_i are assumed to lie in the Sobolev space $H = H^k[0, T]$, while the f_i are smooth $\mathcal{C}^\infty(\mathcal{M})$vector fields.

System motion can be subjected to constraints that may arise from the structure of the mechanism, or from the way in which it is actuated and controlled. Constraints may be expressed as equalities or inequalities, *bilateral* or *unilateral* constraints, respectively, and they may explicitly depend on time or not, *rheonomic* or *scleronomic* constraints.

Motion restrictions that may be put in the form

$$h_i(q) = 0,\ i = 1, \ldots, k < n,\ h_i : \mathcal{Q} \mapsto \mathbb{R}, \text{ smooth} \tag{2}$$

are called *holonomic constraints*. Its effect is to confine the attainable system configurations to an $(n-k)$-dimensional smooth submanifold of $\mathcal{Q}$. The problem is solved by defining $n-k$ new coordinates on the restricted submanifold that characterize the actual *degrees of freedom* of the system. For simulation purposes, a *Differential-Algebraic Equation (DAE)* system solver is used.

System constraints whose expression involves generalized coordinates and velocities in the form

$$a_i(q, \dot{q}) = 0,\ i = 1, \ldots, k < n,\ a_i : \mathcal{Q} \mapsto \mathbb{R}^n, \text{ smooth}$$

are referred to as *kinematic constraints*. These will limit the admissible motions of the system by restricting the set of *generalized velocities* that can be attained at a given configuration. In mechanics, such constraints are usually encountered in the Pfaffian form

$$a_i^T(q)\, \dot{q} = 0,\ i = 1, \ldots, k < n, \text{ or } \mathbf{A}^T(q)\, \dot{q} = 0, \tag{3}$$

[6]Not every robot has an associated differential model. This cases are addressed by equating the differentials to zero, and computing the next state by interpolation.

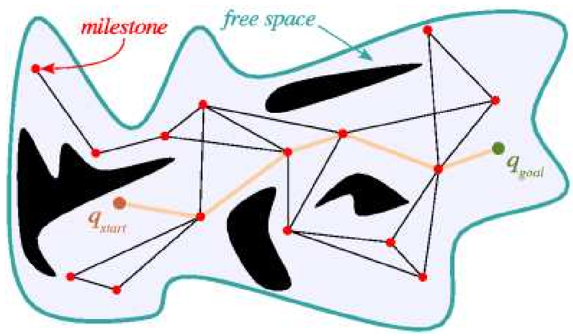

Figure 1: *Basic-PRM*.

It may happen that the kinematic constraints (3) are not integrable, *i.e.*, cannot be put in the form (2). In this case, the constraints and the mechanical system itself are called *nonholonomic*.

3 MOTION PLANNING PROBLEM

Given $q_{init} \in \mathcal{C}_{valid}$ and $q_{goal} \in \mathcal{C}_{valid}$, the main goal of the planning algorithm is to compute a continuous motion planning trajectory τ, or equivalently, a control law $u(t)$, such that $\forall t \in [t_0, t_1]$, $\tau(t) \in \mathcal{C}_{valid}$, and $\tau(t_0) = q_{init}$ and $\tau(t_1) = q_{goal}$.

The idea behind the basic *Probabilistic Roadmap Method* (*PRM*) is to represent and to capture the connectivity of $\mathcal{C}_{free}$ by a random network, a *roadmap*, whose nodes and edges correspond to randomly selected configurations and path segments, respectively.[48,49] In a preprocessing step, or a *learning phase*, a large number of points are distributed uniformly at random in $\mathcal{C}$, and those found to be in $\mathcal{C}_{free}$ are retained as nodes in the roadmap. A local planner is then used to find paths between each pair of nodes that are sufficiently close together. If the planner succeeds in finding a path between two nodes, they are connected by an edge in the roadmap. In the *query phase*, the user specified start and goal configurations are connected to the roadmap by the local planner. Then the roadmap is searched for a shortest path between the given points, Fig. 1.

A strategy for building a *Basic-PRM* can be summarized as follows, Alg. 1: Let G be a graph composed by a set E of edges that connect two configurations by a free path. The path represents an edge, the set V contains all the configuration nodes of the graph. A configuration q_{rand} is randomly chosen each iteration with uniform distribution. If this configuration belongs to $\mathcal{C}_{free}$ then it is added to V. Then it is tested the connection with its neighbors. The neighbors of this configuration are defined as the nodes whose distance to the configuration q_{rand} is less than d_{max}.

The *Basic-PRM* in its original form, is a purely geometric planner. There exist several modification to take into account system constraints and/or dynamics,[36,50] but none of them are generic solution for rapid development: a careful study of the resulting trajectories of a particular system must be obtained.

Algorithm 1 *PRM-Basic*

PRM_BASIC (q_{init})

$V \rightarrow \emptyset$; $E \rightarrow \emptyset$; $n_{nodes} \rightarrow 0$;
while $n_{nodes} < N_{\max}$ **do**
    $q_{rand} \leftarrow$RANDOM_CONFIG();
    if $q_{rand} \in \mathcal{C}_{free}$ **then**
        $V \rightarrow V \bigcup \{q_{rand}\}$;
        $n_{nodes} \leftarrow n_{nodes} + 1$;
        $V_c \leftarrow$ set of neightbours of q_{rand} (including q_{rand})
        for all $v \in V_c$ in increasing order of distance **do**
            if $\neg$connected (q_{rand}, v) &$\mathcal{L}(q_{rand}, v) \in \mathcal{C}_{free}$ **then**
                $E \leftarrow E \bigcup \{(q_{rand}, v)\}$;
            end if
        end for
    end if
end while

The *Rapidly-Exploring Random Tree* (*RRT*)[51] is an exploration algorithm for quickly searching high-dimensional spaces that have both *global constraints,* arising from workspace obstacles and velocity bounds, and *differential constraints*, arising from system kinematics and dynamics. The key idea is to bias the exploration toward unexplored portions of the space by randomly sampling points in the state space and incrementally *pulling* the search tree toward them.

In order to build a *Basic-RRT*, Alg. 2, a simple iteration is performed in which each step attempts to extend the *RRT* by adding a new vertex that is biased by a randomly-selected configuration. The EXTEND function selects the nearest vertex already in the *RRT* to the given sample configuration, $\mathbf{q}$. The nearest vertex is chosen according to a metric ρ. The function NEW_STATE makes a motion toward q by applying an input u for some time increment Δt, with some fixed incremental distance ϵ, and test for collision. This input can be chosen at random, or selected by trying *all possible inputs* and choosing the one that yields a new state as close as possible to the sample, q. NEW_STATE implicitly uses the collision detection function to determine whether the new state and all intermediate states satisfy the global constraints. Three situations can occur: REACHED, in which q is directly added to the *RRT* because it already contains a vertex within ϵ of q; ADVANCED, in which a new vertex $q_{new} \neq q$ is added to the *RRT*; TRAPPED, in which the proposed new vertex is rejected because it does not lie in $\mathcal{C}_{free}$.

Notwithstanding the basic-*RRT* can be used in isolation as a path planner, the problem is that, without any bias toward the goal, convergence might be very slow. Several improved planning algorithm have been proposed based on the basic-*RRT*, *p.e.*, *RRT-GoalBias*, *RRT-Connect*, and *MPC-RRT*.

In *RRT-GoalBias,* RANDOM_STATE is replaced by a *biased coin* that if it returns

Algorithm 2 *RRT-Basic*

BUILD_RRT (q_{init})

$\mathcal{T}.init\,(q_{init})$
for $k = 1$ to K **do**
    $q_{rand} \leftarrow$ RANDOM_CONFIG $(\mathcal{T}, q_{rand})$
end for
$Return\,\mathcal{T}$

EXTEND $(\mathcal{T}, q)$

$q_{near} \leftarrow$ NEAREST_NEIGHBOR $(\mathcal{T}, q)$
if NEW_CONFIG (q, q_{near}, q_{new}) **then**
    $\mathcal{T}$.add_vertex (q_{new})
    $\mathcal{T}$.add_edge $(q_{near}, q_{new}, u_{new})$
    if $q_{new} = q$ **then**
        Return REACHED
    else
        Return ADVANCED
    end if
    Return TRAPPED
end if

heads, then q_{goal} is returned; otherwise, a random state is returned. Even with a small probability of returning heads, such as 0.05, *RRT-GoalBias* converges to the goal much faster than the basic *RRT*.

RRT-Connect is a probabilistically complete planner designed specifically for path planning problems that involve no differential constraints, or that can be integrated in time reversed. The method is based on two ideas: CONNECT heuristic that attempts to move over a longer distance, and the growth of *RRT*s from both q_{init} and q_{goal}. The CONNECT heuristic is a greedy function that can be considered as an alternative to the EXTEND function of the basic-*RRT*, Alg. 3. Instead of attempting to extend an *RRT* by a single ϵ step, the CONNECT heuristic iterates the EXTEND step until q, an obstacle, or an infeasible configuration is reached. This operation has a similar function than the artificial potential function in a randomized potential field approach.[43] With the CONNECT heuristic, the basin of attraction continues to move around as the *RRT* grows, as opposed to an artificial potential field method, in which the basin of attraction remains fixed at the goal. Two trees $\mathcal{T}_a$ and $\mathcal{T}_b$ are maintained at all times until they become connected and a solution is found. In each iteration, one tree is extended, and an attempt is made to connect the nearest vertex of the other tree to the new vertex. Then the roles are reversed by swapping the two trees. This causes both trees to explore $\mathcal{C}_{free}$, while trying to establish a connection between them.

MPC-RRT was proposed by the author of this paper.[11,46] It consists in changing the

Algorithm 3 *RRT-Connect*

CONNECT $(\mathcal{T}, q)$

repeat
  $S \leftarrow$ EXTEND $(\mathcal{T}, q)$
until not $(S = Advanced)$
Return S

RRT_BIDIRECTIONAL (q_{init}, q_{goal})

$\mathcal{T}_a$.init (q_{init}) , $\mathcal{T}_b$.init (q_{goal})
for $k = 1$ to K **do**
  $q_{rand} \leftarrow$ RANDOM_CONFIG()
  if not (**EXTEND** $(\mathcal{T}_a, q_{rand}) = Trapped$) **then**
    if (**CONNECT** $(\mathcal{T}_a, q_{new}) = Reached$) **then**
      Return PATH $(\mathcal{T}_a, \mathcal{T}_b)$
    end if
    SWAP $(\mathcal{T}_a, \mathcal{T}_b)$
  end if
  Return $Failure$
end for

CONNECT heuristic by a *path-space iteration method*.[37–39, 52] This method iterate on the control along a trajectory until a feasible trajectory is found, incorporating the inequality constraints by using *exterior penalty functions*. At each iteration a performance index that reflects the feasibility of the path itself is minimized, *i.e.*, the error between the end-point map of the system and the desired final configuration is minimized. The convergence of the algorithm requires that the brackets of the *Control Lie Algebra* (*CLA*) satisfy a linear growth condition. For the exterior penalty functions a *generalized potential function*[10, 11] is computed. This functions assume that the *generalized potential* of a differential mass dm is given by dm/r^m. Then it can be shown that the *generalized potential field* of a polyhedral body of mass M and volume V can be expressed as:

$$U(q) \equiv \iiint_M \frac{dm}{r^m} = k \iiint_V \frac{dV}{r^m} = \sum_{e \in \mathcal{E}} U_e(q), \tag{4}$$

where $U_e(q)$ can be computed in closed form if $m = 1$ or $m > 3$.[10, 11]

4 MPSLAB: MOTION PLANNING AND SIMULATION IN 3D ENVIRONMENTS

The previous sections have introduced some of the tools needed for motion planning in virtual environments. In order to try different planners a flexible platform is needed: *MPSLab* is a *C++* generic software package intended for motion planning and simulation in virtual environments. It can represent multiple robots with high degrees of freedom

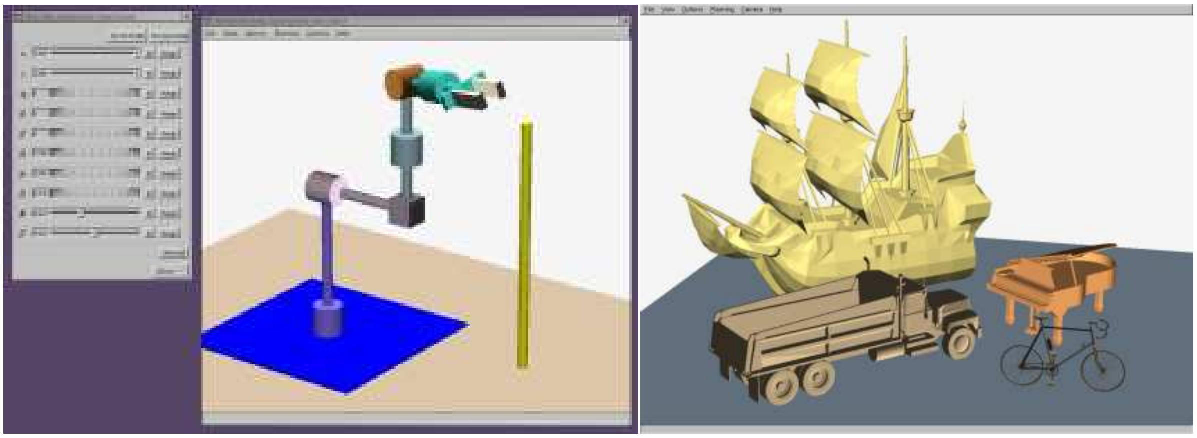

Figure 2: MPSLab screen shots

using polyhedra, Fig. 2. Generality and easy implementation are the purposes of this software. To this end, it is used a language similar to *VRML* as language programming, and were added three collision packages/distance computation libraries, three differential equation solvers, and an optimization package. Figure 3 shows a typical simulation using *MPSLab*.

In the following paragraphs the main components of *MPSLab* are resumed.

Collision Detection/distance computation Constructing an explicit representation of the obstacle region, $\mathcal{CB}$, is not an easy task, and is nor always required. It is often preferable to simply build a logical predicate that serves as a probe that tests whether a configuration lies in $\mathcal{CB}$. This is referred to as *collision detection*. *Distance computation* refers to the computation of the minimum distance of any object to any body of the robot.

The packages used by *MPSLab* for these purposes are: *PQP*, a library for collision detection, distance computation, and tolerance verification on a pair of geometric models.[29] *V-Collide*, a package for collision detection between arbitrary polygonal objects in large environments[30]. *SWIFT++*, a library for intersection detection, tolerance verification, approximate or exact distance computation of polyhedral models.[26]

Differential Equation Solvers *MPSLab* supports resolution of ordinary differential equation (*ODE*) using *RKSUITE*[53] and a library written by the author of this paper, and resolution of differential algebraic equation (*DAE*) using *DASPK*.[54]

Optimization *Nonlinearly constrained large-scale optimization* can be solved with the interface with *Huge Quadratic Programming* (*HQP*).[55]

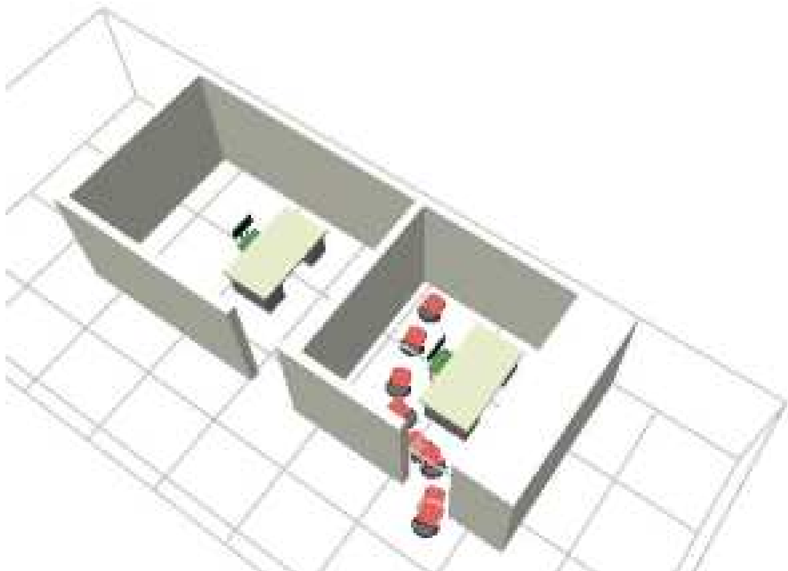

Figure 3: *MPSLab* simulation.

5 DISCUSSION

Motion planning algorithms born with the main idea of solving the problem of finding a collision-free path for a robot in a given environment. However, several exciting application fields have been appeared in last few years and new ones present challenges for researches. For instance:

Medical Surgery Imaging techniques are widely available to produce detailed and precise computer representations of $3D$ tissue structures. It is desirable to precompute minimally invasive paths of surgical tools among soft tissue structures having different elastic properties. Motions must be computed for objects sliding in contact with one another, which requires dealing with friction models.

Planning for sensing Sensors can be used to acquire information about an environment, like building a $3D$ model or localizing objects of interest. *Active sensing* deals with the problem of finding the next placement (and the corresponding motion) where the acquisition of *new* information is a maximum to avoid redundancy and data explosion.

Graphic Animation During the last few years, animation have incorporated dynamic modeling to create physically realistic animation, vision sensing to automatically acquire $3D$ models, haptic interaction to *feel* virtual objects, and motion planning to create collision-free motions. Fast planner can help to generate realistic-looking motions.

6 CONCLUSIONS AND FUTURE WORK

In this work a novel software package combining *randomized motion planning algorithms*, *collision detection/distance computation*, *optimization*, and *differential equation resolution* have been presented.

There exist a lot of applications where this software can be used, and more research is being done on the integration of this simulation platform with real systems.[46] Moreover, the author is adding collision response capability to finish the simulation power of the software. There is also work on programming more complex examples, *i.e.*, robots with several degrees of freedom (more than 70) in order to test the real capacity of the planner.

REFERENCES

[1] N.J. Nilsson. A mobile automation: An application of artificial intelligence techniques. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 509–520, Washington D.C., (1969).

[2] S. Udupa. *Collision Detection and Avoidance in Computer Controlled Manipulators.* PhD thesis, Department of Electrical Engineering, California Institute of Technology, Pasadena, CA., (1977).

[3] T. Lozano-Pérez and M.A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of ACM*, **22**(10), 560–570 (1979).

[4] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, **C-32**(2), 108–120 (1983).

[5] J.H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings of FOCS*, pages 421–427, (1979).

[6] Y.K. Hwang and N. Ahuja. Gross motion planning - A survey. *ACM Computing Surveys*, **24**(3), 219–291 (1992).

[7] J.C. Latombe. *Robot Motion Planning.* Kluwer Academic Pub., Boston, (1991).

[8] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, **5**(1), 90–98 (1986).

[9] M. Khatib. *Sensor-based control for mobile robots.* PhD thesis, Laboratoire d'Automatique et d'Analyse des Systemes LAAS-CNRS, 7, Avenue du Colonel Roche 31077 Toulouse Cedex 4, France, (December 1997).

[10] O.A.A. Orqueda and O.E. Agamennoni. Motion planning and control of autonomous vehicles - I: Generalized potential field functions. In *X Reunión de Trabajo en Procesamiento de la Información y Control (RPIC'03)*, San Nicolás de los Arroyos, Argentina, (October 2003).

[11] O.A.A. Orqueda, J.L. Figueroa, and O.E. Agamennoni. Motion planning and control: From virtual environments to the real world. In *Proceedings of the IFAC International Symposium on Intelligent Components and Instruments for Control Applications (SICICA)*, Aveiro, Portugal, (July 2003).

[12] J. Barraquand and J.C. Latombe. Robot motion planning: A distributed represen-

tation approach. *International Journal of Robotics Research*, **10**(6), 628–649 (1991).

[13] L.E. Kavraki, J-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proceedings of the 27th annual ACM symposium on Theory of computing*, pages 353–362, Las Vegas, NV, USA, (May 1995).

[14] J. Barraquand, L.E. Kavraki, J.C. Latombe, T.Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. *International Journal of Robotics Research*, **16**(6), 759–774 (1997).

[15] P. Bessière, J-M. Ahuactzin, El-Ghazali Talbi, and E. Mazer. The Ariadne's clew algorithm: Global planning with local methods. In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, (1993).

[16] E. Mazer, J.M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. *Journal of Artificial Intelligence Research (JAIR)*, **9**, 295–316 (1998).

[17] D. Hsu, J.C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2719–2726, Albuquerque, NM, (1997).

[18] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *Proceedings of the IEEE International Conference on Robotics and Automation*, (1998).

[19] D. Hsu, J.C. Latombe, and S. Sorkin. Placing a robot manipulator amid obstacles for optimized execution. In *Proceedings of the IEEE International Symposium on Assembly and Task Planning (ISATP'99)*, pages 280–285, Porto, Portugal, (1999).

[20] V. Boor, M.H. Overmars, and A.F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1018–1023, Detroit, MI, (1999).

[21] C. Nissoux, T. Siméon, and J.-P. Laumond. Visibility-based probabilistic roadmaps. In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1316–1321, Kyongju, Korea, (October 1999).

[22] S. Quinlan. Efficient computation between non-convex objects. In *Proceedings of the IEEE International Conference on Robotics and Automation(ICRA '94)*, pages 144–154, (1994).

[23] J. Cohen, M. Lin, D. Manocha, and K. Ponamgi. Interactive and exact collision detection for large-scale environments. Technical Report TR94-005, Department of Computer Science, University of N. Carolina, Chapel Hill, (1994).

[24] J. Cohen, M. Lin, D. Manocha, and K. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scaled environments. In *Proceedings of ACM Interactive 3D Graphics Conference*, pages 189–196, (1995).

[25] S. A. Ehmann and M. C. Lin. Accelerated proximity queries between convex polyhedra by multi-level Voronoi marching. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, (2000).

[26] S.A. Ehmann and M.C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Proceedings of EUROGRAPHICS 2001*,

number 3 in Vol. 20, (2001).

[27] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, **4**(2) (April 1988).

[28] S. Gottschalk, M. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Computer Graphics*, **30**(Annual Conference Series), 171–180 (1996). also.

[29] S. Gottschalk, M. Lin, D. Manocha, and E. Larsen. PQP - The proximity query package. http://www.cs.unc.edu/geom/SSV/, (1999).

[30] T. Hudson, M. Lin, J. Cohen, S. Gottschalk, and D. Manocha. V-COLLIDE: Accelerated collision detection for VRML. In R. Carey and P. Strauss, editors, *Proceedings of 2nd Symposium on the Virtual Reality Modeling Language (VRML'97)*, New York, NY, (February 1997).

[31] J.T. Klosowski. *Efficient collision detection for interactive 3D graphics and virtual environments.* PhD thesis, Applied Mathematics and Statistics, State University of New York, (May 1998).

[32] M. Lin, D. Manocha, J. Cohen, and S. Gottschalk. Collision detection: Algorithms and applications. In Jean-Paul Laumond and A.K. Peters M. Overmars, editors, *Proceedings of Algorithms for Robotics Motion and Manipulation*, pages 129–142, (1998). invited submission.

[33] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR-97-05, MERL - Mitsubishi Electric Research Laboratory, 201 Broadway, Cambridge, Massachusetts 02139, (June 1997).

[34] C Van Geem and T. Siméon. Kcd: a collision detector for path planning in factory models. Technical Report Rapport LAAS N°01073, LAAS-CNRS, 7 Av. Colonel Roche 31077, Toulouse Cedex 4, France, (March 2001).

[35] L. Kavraki. Computation of configuration-space obstacles using the Fast Fourier Transform. *IEEE Transactions on Automatic Control*, **11**(3), 408–413 (1995).

[36] J.-P. Laumond. *Robot Motion Planning and Control*, volume 229 of *Lecture Notes in Control and Information Sciences.* Springer Verlag, (January 1998).

[37] H.J. Sussmann. A continuation method for nonholonomic path-finding problems. In *Proceedings of the 32nd IEEE Conference on Decision and Control*, pages 2718–2723, San Antonio, TX, (December 1992). IEEE publications.

[38] A.W. Divelbiss and J.T. Wen. A path space approach to nonholonomic motion planning in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, **13**(3), 443–451 (June 1997).

[39] D-O. Popa. *Path-Planning and Feedback Stabilization of Nonholonomic Control Systems.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY 12180, (April 1998).

[40] B.R. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM*, **40**(5), 1048–1066 (November 1993).

[41] C.I. Connolly, R.A. Grupen, and K.X. Souccar. A Hamiltonian approach to kinody-

namic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, page 940, (1995).

[42] B.R. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, **14**(6), 443–479 (1995).

[43] S.M. LaValle and J.J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Proceedings of the 2000 Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, (March 2000).

[44] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, **20**(5), 378–400 (2001).

[45] R. Kindel, D. Hsu, J.C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, (April 2000).

[46] O.A.A. Orqueda. *Motion Planning and Control of Autonomous Robots*. Doctoral thesis, Graduate College of the National University of the South, Av. Alem 1253, Bahía Blanca, Argentina, (September 2003). Under revision.

[47] J.J. Kuffner and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 95–101, San Francisco, CA, (April 2000).

[48] L.E. Kavraki. *Random Networks in Configuration Space for Fast Path Planning*. PhD thesis, Stanford University, Stanford University, (December 1994).

[49] L.E. Kavraki, P. Švestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, **12**(4), 566–580 (1996).

[50] A. De Luca and G. Oriolo. Modeling and control of nonholonomic mechanical systems. In *Proceedings of CISM95*, number 7, (1995).

[51] S.M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Department of Computer Science, Iowa State University, Ames, Iowa, (October 1998).

[52] E.D. Sontag. Control of systems without drift via generic loops. *IEEE Transactions on Automatic Control*, **40**, 413–440 (1995).

[53] R.W. Brankin, I. Gladwell, and L.F. Shampine. RKSUITE: a suite of Runge-Kutta codes for the initial value problem for ODEs. Softreport 92-S1, Department of Mathematics, Southern Methodist University, Dallas, Texas, U.S.A., (1992).

[54] P.N. Brown, A.C. Hindmarsh, and L.R. Petzold. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM Journal on Scientific Computing*, **15**, 1467–1488 (1994).

[55] R. Franke. *Omuses: A tool for the optimization of multistage systems and HQP: A solver for sparse nonlinear optimization*. Dept. of Automation and Systems Engineering, Technical University of Ilmenau, Germany, 1.5 edition, (September 1998).