8

Key Elements for Motion Planning Algorithms

Antonio Benitez, Ignacio Huitzil, Daniel Vallejo,
Jorge de la Calleja and Ma. Auxilio Medina
Universidad Politécnica de Puebla,
Universidad de las Américas – Puebla, México

1. Introduction

Planning a collision-free path for a rigid or articulated robot to move from an initial to a final configuration in a static environment is a central problem in robotics and has been extensively addressed over the last. The complexity of the problem is NP-hard (Latombe, 1991). There exist several family sets of variations of the basic problem, that consider flexible robots, and where robots can modify the environment. The problem is well known in other domains, such as planning for graphics and simulation (Koga et al., 1994), planning for virtual prototyping (Chang & Li, 1995), and planning for medical (Tombropoulos et al., 1999) and pharmaceutical (Finn & Kavraki, 1999) applications.

1.1 Definitions and Terminology

A robot is defined into the motion planning problems as an object, which is capable to move (rotating and translating) in an environment (the workspace) and it may take different forms, it can be: a rigid object, an articulated arm, or a more complex form like a car or an humanoid form.

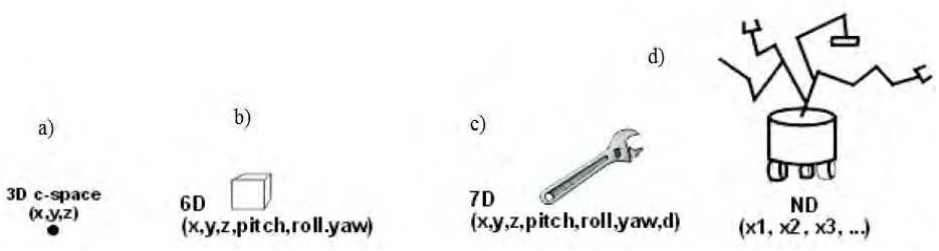


Fig. 1. Robot types.

Given different robot types, it is fully and succinctly useful to represent the position of every point of the robot in a given moment. As shown in Figure 1. (a), a robot can be represented as a point. When the robot is a point, as it is the case in theoretical discussion, it can be

completely described by its translational coordinates, (x, y, z). For a robot which is a rigid body freely moving in a 3D space (See Figure 1. (b)), the position of every point is represented by six parameters (x, y, z) for the position and (α, β, γ) for its rotation in every point on the space. Each parameter, or coordinate, necessary to give the full description of the robot, is called a degree of freedom (DOF). The seven DOF shown in Figure 1. (c) is a spanning wrench. It has the same six DOF as the cube robot plus a seventh DOF, the width of the tool jaw. The far right n DOF in Figure 1. (d) demonstrates that the number of DOF can become extremely large as robots become more and more complex.
Not only the number, but the interpretation of each DOF is necessary to fully understand the robot's position and orientation. Figure 2 (a) Shows six DOF that are necessary to describe a rigid object moving freely in 3D. Six parameters are also necessary to describe a planar robot with a fixed base and six serial links Figure 2. (b). Although the same size, the coordinate vectors of each robot are interpreted differently.

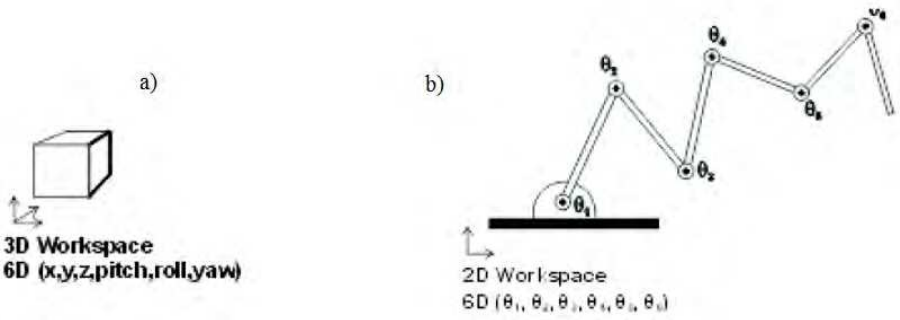


Fig. 2. Number of Degrees of freedom for robots.

1.2 Paths and Connectivity

Throughout the history of artificial intelligence research, the distinction between *problem solving and planning* has been rather elusive (Steven, 2004). For example, (Russell & Norvig, 2003) devotes a through analysis of "Problem-solving" and "Planning". The core of the motion planning problem is to determine whether "a point of the space" is reachable from an initial one by applying a sequence of actions (Russell & Norvig, 2003), p. 375.
Besides, it is difficult to apply results from computational geometry to robot motion planning, since practical aspects relevant to robotics are often ignored in computational geometry, e.g. only static environments are considered usually. Since testing for collisions between the robot and the environment is essential to motion planning, the representation of geometry is an important issue.
The motion planning problem can be defined (Steven, 2004) as a continuum of actions that can lead to a path in a state space. The path is obtained through the integration of a vector field computed by using the plan. Thus, the plane has an important role because it describes a set of states.

1.3 Workspace and Configuration Space

The robot moves in a workspace, consisting of several objects, guided by natural lows. For motion planning algorithms, the concept of workspace can be defined by considering two or three dimensions; of course it can be defined by N-dimensions. In this context, the workspace consists of rigid objects (obstacles) with six DOF. Initially, obstacles are placed in static configurations, so they can not move within the environment. The workspace representation is associated to a geometric model used to manipulate the objects. Both features must be considered to address the motion planning problem. However, in many instances, it may be possible to improve performance by carefully investigating the constraints that arise for particular problems once again. It may be possible to optimize performance of some of the sampling-based planners in particular contexts by carefully considering what information is available directly from the workspace constraints.

1.3.1 Configuration Space

If the robot has n degrees of freedom, this leads to a manifold of dimension n called the *configuration space* or *C-space*. It will be generally denoted by C. In the context of this document, the configuration space may be considered as a special form of state space. To solve a motion planning problem, algorithms must conduct a search in this space. The configuration space notion provides a powerful abstraction that converts the complicated models and transformations into the general problem of computing a path in a manifold (Steven, 2004).

1.3.2 The motion planning problem

The basic motion planning problem is defined as follows: Given a robot, which can be any moving object, a complete description of static workspace and star and goal configurations, the objective is to find a valid (i.e., collision free) path for the robot to move through the workspace from beginning to goal configurations. The robot must avoid collision with itself as well as obstacles in the workspace environment.

1.3.3 Probabilistic roadmap methods

A class of motion planning methods, known as probabilistic roadmap methods (*PRMs*), have been largely addressed (Ahuactzin, & Gupta, 1997), (Amato et al., 1998), (Boor et al., 1999). Briefly, *PRMs* use randomization to construct a graph (*a roadmap*) in configuration-space (C-space). *PRMs* provide heuristics for sampling C-space and C-obstacles without explicitly calculating either.

When PRM maps are built, roadmap nodes correspond to collision-free configurations of the robot, i.e. points in the free C-space (C-free). Two nodes are connected by an edge if a collision-free path between the two corresponding configurations can be found by a "local planning" method. Local planners take as input a pair of configurations and check the path (edge) between them for collision. As output they declare the path valid (collision-free) or invalid. Because of the large number of calls made to local planners, their design often sacrifices sophistication for speed. When local planners (Amato et al., 1998) are deterministic, the edges do not need to be saved, only the roadmap adjacency graph must saved. *PRM* methods may include one or more 'enhancement' steps in which some areas are

sampled more intensively either before or after the connection phase. The process is repeated as long as connections are found or a threshold is reached.
A particular characteristic of *PRM* is that queries are processed by connecting the initial and goal configurations to the roadmap, and then searching for a path in the roadmap between these two connection points.
The following pseudo-code summarizes the high-level algorithm steps for both roadmap construction and usage (query processing). Lower-level resources include distance metrics for evaluating the promise of various edges local planners for validating proposed edges, and collision detectors for distinguishing between valid (collision free) and invalid (in collision) robot configurations.

Pre-processing: Roadmap Construction
1. Node Generation (find collision-free configurations)
2. Connection (connect nodes to form roadmap)

(Repeat the node generation as desired)

On Line: Query Processing

1. Connect start/goal to roadmap
2. Find roadmap path between connection nodes

(repeat for all start/goal pairs of interest)

1.4 The narrow corridor problem

A narrow passage occurs when in order to connect two configurations a point from a very small set must be generated. This problem occurs independently of the combinatorial complexity of the problem instance. There have been several variants proposed to the basic *PRM* method which do address the "narrow passage problem". Workspaces are difficult to handle when they are "cluttered". In general, the clutter is made up of closely positioned workspace obstacles. Identifying "difficult" regions is a topic of debate. Nevertheless when such regions are identified the roadmap can be enhanced by applying additional sampling. Naturally, many of the nodes generated in a difficult area will be in collision with whatever obstacles are making that area difficult. Some researchers, not wishing to waste computation invested in generating nodes discovered to be in-collision, are considering how to "salvage" such nodes by transforming them in various ways until they become collision-free.

2. Probabilistic Roadmap Methods

This section presents fundamental concepts about PRM, including a complete description of PRM, OBPRM, Visibility Roadmap, RRT and Elastic Band algorithms. These methods are important because they are the underlying layer of this topic. The parametric concept of configuration space and its importance to build the roadmap is also presented.

The world (Workspace or W) generally contains two kinds of entities:

1. *Obstacles*: Portions of the world that are "permanently" occupied, for example, as in the walls of a building.

2. *Robots*: Geometric bodies that are controllable via a motion plan.

The dimension of the workspace determines the particular characteristic of W. Formulating and solving motion planning problems requires defining and manipulating complicated geometric models of a system of bodies in space. Because physical objects define spatial distributions in 3D-space, geometric representations and computations play an important role in robotics.

There are four major representation schemata for modelling solids in the physical space (Christoph, 1997). They are the follows. In constructive solid geometry (CSG) the objects are represented by unions, intersections, or differences of primitive solids. The boundary representation (BRep) defines objects by quilts of vertices, edges, and faces. If the object is decomposed into a set of nonintersecting primitives we speak of spatial subdivision. Finally, the medial surface transformation is a closure of the locus of the centres of maximal inscribed spheres, and a function giving the minimal distance to the solid boundary. We describe the boundary representation because this is related to our work.

2.1 Geometric Modelling

There exists a wide variety of approaches and techniques for geometric modelling. Most solid models use BRep and there are many methods for converting other schemata into BRep (Christoph, 1997). Research has focused on algorithms for computing convex hulls, intersecting convex polygons and polyhedron, intersecting half-spaces, decomposing polygons, and the closest-point problem. And the particular choice usually depends on the application and the difficulty of the problem. In most cases, such models can be classified as: 1) a boundary representation, and 2) a solid representation.

Fig. 3. Triangle strips and triangle fans can reduce the number of redundant points.

Suppose $W = \mathbb{R}^3$. One of the most convenient models to express the elements in W is a set of triangles, each of which is specified by three points, (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3). It is assumed that the interior of the triangle is part of the model. Thus, two triangles are considered as *colliding* if one pokes into the interior of another. This model is flexible because there are no constraints on the way in which triangles must be expressed. However, there exists redundancy to specify the points. Representations that remove this redundancy

are triangle strips. *Triangle strips* is a sequence of triangles such that each adjacent pair shares. *Triangle fan* is triangle strip in which all triangles share a common vertex, as shown in Figure 3.

2.2 Rigid body transformations

Once we have defined the geometric model used to represent the objects in the workspace, it is necessary to know how these objects are going to be manipulated (the robot as a particular case) through the workspace. Let *O* refer to the obstacle region, which is a subset of *W*. Let *A* refer to the robot, which is a subset of $\mathbb{R}^2$ [2] or $\mathbb{R}^3$, matching the dimension of *W*. Although O remains fixed in the world, W, motion planning problems will require "moving" the robot, *A*.

Let *A* be a rigid body which we want to translate by some $x_t, y_t, z_t \in \mathbb{R}$ by mapping every $(x, y, z) \in A$ to $(x + x_t, y + y_t, z + z_t)$. Primitives of the form $H_i = (x, y, z) \in W \mid f_i(x, y, z) \leq 0$, are transformed to $(x, y, z) \in W \mid f_i(x - x_t, y - y_t, z - z_t) \leq 0$. The translated robot is denoted as $A(x_t, y_t, z_t)$. Note that a 3D body can be independently rotated around three orthogonal axes, as shown in Figure 4.

1. A *yaw* is a counter clockwise rotation of α about the Z-axis. The rotation is given by the following matrix.

$$R_Z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

Note that the upper left entries of $R_Z(\alpha)$ form a 2D rotation applied to the *XY* coordinates, while the *Z* coordinate remains constant.

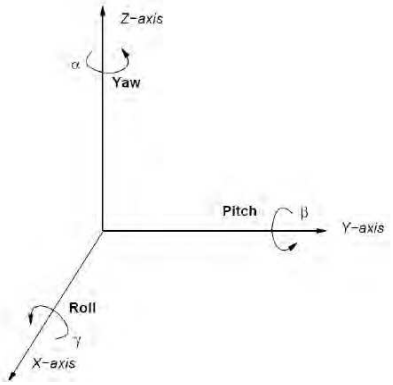


Fig. 4. Any rotation in 3D can be described as a sequence of yaw, pitch, and roll rotations.

2. A *pitch* is a counter clockwise rotation of β about the Y-axis. The rotation is given by the following matrix.

$$\begin{pmatrix} \cos\beta & 0 & \sin\beta \end{pmatrix} \quad (2)$$

$$R_Y(\beta)= \begin{matrix} 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{matrix}$$

3. A *roll* is a counter clockwise rotation of γ about the X-axis. The rotation is given by the following matrix.

$$R_x(\gamma)= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} \quad (3)$$

As in the 2D case, a homogeneous transformation matrix can be defined. For the 3D case, a 4 X 4 matrix is obtained that performs the rotation given by R(α, β, γ), followed by a translation given by x_t, y_t, z_t. The result is defined as:

$$T= \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\cos\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & x_t \\ \cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & y_t \\ \beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4)$$

The triangle meshes associated to the robot (using a triangle as primitive) can be transformed, to yield A(xt, yt, zt, α, β, γ).

2.3 Configuration Space

Configuration-space (C-space) is an abstraction of the motion planning problem. Briefly, the motion planning problem is expressed in a *n-dimensional* space, where *n* represents the number of DOF of the robot, and a robot configuration (all the necessary DOF's for fully describing the robot's position and pose) is a point. The C-space consist ALL possible points, those of free collision configuration corresponding to the robot free space and collision configuration corresponding to the robot in collision with or more obstacles or itself. In a 3D workspace, the path between any starting and configuration is a swept volume. In a given C-space, the same path is a one-dimensional curve traced by the C-space point representing the robot moving from a start to configuration. Only when the robot is *really* a point robot (as in theoretical discussion) the workspace and the robot's C-space are the same.
If the robot has *n* degrees of freedom, this leads to a manifold of dimension the *configuration space* or *C-space*. It will be generally denoted by C. Configurations space has different properties, next paragraphs describes them.

2.3.1 Paths

Let X be a topological space, which is a manifold. A *path*, τ , in X is a continuous function, τ : [0, 1] $\rightarrow X$. Other intervals of $\mathbb{R}$ may alternatively be used for the domain of τ . Note that a path is a function, not a set of points. Each point along the path is given by τ (s) for some $s \in$

[0, 1]. Recall in that case, a countable set of stages was denned, and the states visited could be represented as x_1, x_2,.... In the current setting τ (s) is used, in which s replaces the stage index. To make connection clearer, we could use x instead of τ, to obtain $x(s)$ for each $s \in$ [0, 1].

3. Traditional Roadmap Methods

A class of motion planning methods, known as probabilistic roadmap methods (*PRMs*), have made large recent gains in popularity. In general roadmap methods solve the motion planning problem in two phases: roadmap construction and roadmap querying (runtime). It is important to understand that the objective of the pre-processing phase is to build an adequate roadmap for any particular problem. Such result is used to quickly satisfy (typically within a few seconds) a large number of different queries to the same roadmap at execution time. It is not assumed that any crucial start and goal configurations are known a *priori* but rather that the entire workspace must be explored and mapped before the robot can efficiently operate there. The process is analogous to that of cartographers making a map of a country's road system. When they finish, many drivers can all obtain the same map and use it for their individual navigation purposes. There is no single start to goal; rather all possible routes must be explored and charted.

3.1 General PRM

Briefly, PRMs use randomization to construct a graph (a roadmap) in a configuration space (C-space) taking into account that there exist forbidden and feasible spaces. Therefore the algorithm computes configurations in both regions and tests for collision to determine which configurations are going to be retained. Once the configuration space has been sampled using a tool called "Local Planner". This local planner verifies if there exists a collision free path between two corresponding configurations. The path exists if two nodes are directly or transitively connected by an edge. As output the local planner declares a valid path (collision-free) or invalid.

Finally, queries are processed by connecting the initial and goal configurations to the roadmap, and then searching for a path in the roadmap between these two connection points, see Figure 5. As described (Kavraki& Latombe, 1994), (Kavraki et al., 1996), (Overmars & Svestka, 1994) the basic *PRM* uses uniform sampling to generate, uniformly and randomly configurations (nodes) of the robot in the workspace. The local planner that is likely to return failure by submitting only pairs of configurations whose relative distance (according to the distance function D) is smaller than some constant threshold MAXDIST. Thus, N_q define:

$$N_q = \{ q' \in N \mid D(q, q') \leq \text{MAXDIST} \} \tag{5}$$

Additionally, according to the algorithm, the authors try to connect q with all nodes in N_q in order to increase the distance from q and another configuration. They skip those nodes which are in the same connected component as q. By considering elements of N_q in this order they expect to maximize the chances of quickly connecting q to other configurations and, consequently, reduce the number of calls to the local planner, (Nice every successful connection results in merging two connected components into one). In (Kavraki et al., 1996), is found it a useful to bound the size of the set N_q by the constant K.

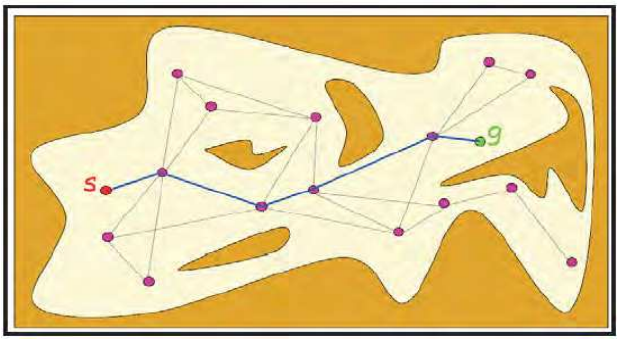


Fig. 5. The PRM is searched for a path from s to g through a graph which represent la connectivity of configuration space.

The distance function. The function D is used both to construct and sort the set N_q of candidate neighbours of each new node q. It should be defined so that, for any pair (q, q') of configurations, $D(q, q')$ reflects the possibility that the local planner will fail to compute a feasible path between these configurations. One possibility is to define $D(q, q')$ as a measure (area/volume) of the workspace region swept by the robot when it moves along the path computed by the local planner between q and q' in the absence of obstacles. Thus, each local planner would automatically induce its own specific distance function. Since exact computation of swept areas/volumes tends to be rather time-consuming, a rough but inexpensive measure of the swept-region gives better practical results. Very simple distance measure also seems to give good results. For example, when the general local planner described above is used to connect q and q', $D(q, q')$ may be defined as follows:

$$D(q, q') = max_{x \in robot} \| x(q) - x(q') \|, \tag{6}$$

Where x denotes a point on the robot, $x(q)$ is the position of x in the workspace when the robot is at configuration q, and $\| x(q) - x(q') \|$ is the Euclidean distance between $x(q)$ and $x(q')$.

3.2 The Obstacle Based PRM

In (Amato & Wu, 1996) presents a randomized roadmap method for motion planning for several DOF robots. The general approach follows traditional roadmap methods: during pre-processing a roadmap is built in C-space; planning consists of connecting the initial and goal configuration to the roadmap, and then finding a path in the roadmap between these two connection points. The main novelty in their approach is a method for generating roadmap candidate points. In particular, they attempt to generate candidate points uniformly distributed on the surface of each C-obstacle.

3.2.1 Sampling and connection strategies

This planner samples the obstacle surfaces without explicitly calculating those surfaces. The *OBPRM* generation strategy provides information which make possible to find a connection strategy where every node generated can be considered for connection with its k closest

neighbours on each obstacle in the environment. *OBPRM* shows marked success in discovering and navigating narrow passages in C-space. They evaluate various sampling and connection strategies within the context of *OBPRM* in (Amato et al., 1998). A multi-strategy for connecting roadmap nodes where different local planners are used at different stages is shown to enhance the connectivity of the resulting roadmap significantly. The most common local planner used by *PRM* methods is a straight line in C-space. They evaluate distance metrics and local planner methods in (Amato et al., 1998) where they propose the *rotate − at − s* local planner. *Rotate − at − s* divides the usual straight line path into three straight line segments to be tested for collision. The first and final segments consist of pure translation while the intervening segment is pure rotation. The choice of strategy depends on the value of s ∈ [0, 1] which is in general provided by the user. Briefly, for each obstacle *X*:

PROTOTYOE NODE GENERATION

1. C_{in}:= collinding robot cfg with C-obtacle X
2. D:= m random directions emanating out from C_{in}
3. for each $d \in D$
4. C_{out}:- free cfg in direction d (if exists)
5. find contact cfg on (C_{in}, C_{out}) by binary search
6. end for

This example strategy was sufficient to establish the potential of the method. However, it is clear that more sophisticated node generation strategies are needed for more complex objects to produce a 'good' distribution of nodes in all the 'different' regions of C-free. Outline below are some of the methods we've implemented and tested. Keeping in the spirit of OBPRM, all these method employ information regarding the environment to guide node generation. Briefly, the methods are designed to generate three types of nodes: (i) *contact* configuration, (ii) *free* configuration (near contact surfaces), and (iii) *sets* of configuration (*shells*) *surrounding* C-obstacles.

Generating contact configurations. The node generation strategy used in the prototype version of OBPRM is attractive due to its simplicity and its efficiency (node generation typically accounted for 1-2% of pre-processing time). However, the distribution of the generated nodes is clear very sensitive to both the shape of the C-obstacle and to the *seed* (origin C_{in} for the binary search). That is, no single seed will yield a good distribution of configuration on the surface of the C-obstacle if its shape is not roughly spherical, and even if the C-obstacle is spherically shape, a seed configurations on the region of its surface closest to the seed.

GENERRATE CONTACT CONFIGURATION

1. p_{rob}:= point associated with root
2. p_{obe}:= point associated with obstacle of interest
3. C_{in}:= translate robot so p_{rob} and p_{obe} coincide an rotate robot randomly until collision
4. d:= random direction emanating out from C_{in}
5. C_{out}:= free cfg in direction d (if exists)
6. Find contact on (C_{in}, C_{out}) by binary search

3.3 Visibility roadmap

In the visibility roadmap method proposed in (Laumond & Siméon, 2000), roadmap nodes are randomly generated just as in a *Basic − PRM* but connection is done as they are generated in the following way. For each generated node, if it can be connected to more than one currently existing connected component or to no currently existing connected component, it is retained and the connecting edges (if any) are added to the roadmap. If the node can be connected to only one of the existing connected component, it is discarded. Multiple connectable nodes are assumed to yield important information about the connectivity of free space. Isolated nodes are assumed to point to unexplored areas of free space. Singly connectable nodes are assumed to represent another sample in an already explored region and are discarded rather than allowing them to increase the size of the roadmap. Because nodes are generated randomly in the basic *PRM* manner, this method is not better than the basic *PRM* at handling narrow passages.

This method proposes a variant of the Probabilistic Roadmap (PRM) algorithm introduced in (Kavraki & Latombe, 1994) (and independently in (Overmars & Svestka, 1995) as the Probabilistic Path Planner). These algorithms generate collision-free configurations randomly and try to link them with a simple local path planning method. A roadmap is then generated, tending to capture the connectivity of the collision-free configuration space CS_{free}. This variant of these approaches takes advantage of the visibility notion. While usually each collision-free configuration generated by the PRM algorithm is integrated to the roadmap, our algorithm keeps only configurations which either connect two connected components of the roadmap, or are not "visible" by some so-called guard configurations. This approach computes roadmaps with small number of nodes. It integrates a termination condition related to the volume of the free space covered by the roadmap. Experimental comparison shows good performances in terms of computation time, especially when applied to configuration spaces with narrow passages.

Roadmaps A *roadmap* is a graph whose nodes are collision-free configurations.

Two nodes q and q' are adjacent if the path $\mathcal{L}(q,q')$ computed by the local method lies in CS_{free}. Roadmaps are used to solve motion planning problems by the so called *query* procedure: given two configurations q_{init} and $q_{goal,}$ the procedure first connects q_{init} (resp.q_{goal}) to the roadmap **R** if there exists q^*_{init} (resp. q^*_{goal}) goal such that $\mathcal{L}(q_{init}, q^*_{init}) \subset CS_{free}$ and $\mathcal{L}(q_{goal,} q^*_{goal}) \subset CS_{free}$. Then the procedure searches for a path in the extended roadmap. If such a path exists, the solution of the motion planning problem appears as a path constituted by a finite connected sequence of subpaths computed by $\mathcal{L}$.

Visibility domains For a given local method $\mathcal{L}$, the visibility domain of a configuration q is defined as the domain:

$$V_{is}\mathcal{L}(q) = \{ q' \in CS_{free} \text{ such that } \mathcal{L}(q, q') \subset CS_{free} \} \quad (7)$$

Configuration q is said to be the *guard* of $V_{is}\mathcal{L}(q)$.

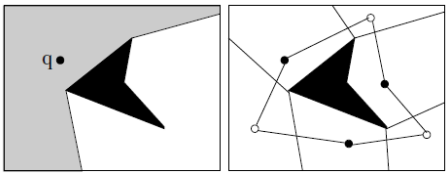


Fig. 6. Visibility domain of a configuration and the visibility roadmap defined by three guards nodes (black) and three connection nodes (white). Here paths $\llcorner(q, q')$ are the straight-line segments $[q, q']$.

- **Free-space Coverage** A set of guards constitutes a *coverage* of CS_{free} if the union of their visibility domains covers the free space CS_{free}. Note that the existence of finite coverage both depends on the shape of CS_{free} and on the local method $\llcorner$. Such finite coverage may not always exist. This issue is related to the notion of ϵ-*goodness* introduced in (Overmars & Svestka, 1995).

Visibility Roadmaps Consider now s visibility domains $V_{is}\llcorner(q_i)$ such that the *s* guards do not "see" mutually through the local method, i.e., $\llcorner(q_i, q'_j) \subset CS_{free}$ for any pair of guards (q_i, q'_j). Then we build the following graph **R**. Guards $\{ q_i \}_{i=1,s}$ are nodes of the graph. For any two intersecting visibility domains $V_{is}\llcorner(q_i)$ and $V_{is}\llcorner(q_j)$, we add a node *q*, called a *connection node*, and two edges (q, q'_i) and (q, q'_j). (see Figure 6) The graph **R** is said to be a *visibility roadmap*. **R** clearly verifies the following property:

Property: Let us assume a visibility roadmap R whose set of guards covers CS_{free}. Let us consider any two configurations q_{init} and q_{goal} such as there exists a connected sequence of collision-free paths of type $\llcorner$ between them. Then there is a guard node q_1 and a guard node q_2 in R such as: $q_{init} \in V_{is}\llcorner(q_1)$, $q_{goal} \in V_{is}\llcorner(q_2)$ with q_1 and q_2 lying in a same connected component of ***R****.*

The notion of visibility roadmap raises several comments:

- Since the definition of the visibility domains is related to a local method, it would have been better to use the term of "reachable domain". Both notions are identical when the local method simply computes straight line segments.
  We keep the word "visibility" because it is more intuitive.
- We consider implicitly that **R** is an undirected graph: that means that $\llcorner$ is assumed to be symmetric.
- Finally the number of guards is not required to be optimal. Optimality refers to the well known and challenging art gallery problem (Goodman & O'Rourke, 1997)

Description The algorithm, called Visib-PRM iteratively processes two sets of nodes: *Guard* and *Connection*. The nodes of *Guard* belonging to a same connected component (i.e. connected by nodes of *Connection*) are gathered in subsets G_i.

Algorithm Visib-PRM

Guard ← Ø; *Connection* ← Ø; *ntry* ← *0*
While (*ntry* < *M*)
Select a random free configuration *q*
g_{vis} ← Ø;G_{vis} ← Ø
For all components G_i of Guard **do**
found ← *FALSE*
For all nodes *g* of G_i **do**
If (*q* belongs to $V_{is}(g)$) **then**
found ← *TRUE*
If (g_{vis} = Ø) then g_{vis} ← *g*; G_{vis} ← G_i
Else /* *q is a connection node* */
Add *q* to Connection
Create edges (*q*, *g*) and (*q*, g_{vis})
Merge components G_{vis} and G_i
until *found* =*TRUE*
If (g_{vis} = Ø)**then** /* *q is a guard node* */
Add{*q*} to Guard; *ntry* ← 0
Else *ntry* ← *ntry* +1
End

3.4 Rapidly Random Tee (RRT)

The idea behind this method is to incrementally construct a search tree that gradually improves the resolution but does not need to explicitly set any resolution parameters. A dense sequence of samples is used as a guide in the incremental construction of the tree. If this sequence is random, the resulting tree is called a rapidly exploring random tree (RRT). In general, this family of trees, whether the sequence is random or deterministic, will be referred to as rapidly exploring dense trees (RDTs) to indicate that a dense covering of the space is obtained.

This method uses the term *state space* to indicate a greater generality than is usually considered in path planning. For a standard problem, *X*= *C*, wich is the configuration space of a rigid body or system of bodies in a 2D or 3D world (Latombe, 1991).

For a given initial state, $x_{init,}$ an RRT, *T*, with *K* vertices is constructed as shown below:

1. GENERATE_RRT (x_{init}, *K*, Δt)
2. *T*.init(x_{init});
3. **For** *k*=1 **to** *K* **do**
4. x_{rand} ← RANDOM_STATE();
5. x_{near} ← NEAREST_NEIGHBOR (x_{rand}, T);
6. *u* ←SELECT_INPUT ($x_{rand,}$ x_{near});

7. $x_{new} \leftarrow$ NEW_STATE ($x_{near,\ u}$, Δ_t)
8. T.add_vertex(x_{new});
9. T.add_edge($x_{nea,}\ x_{new\ ,\ u}$);
10. Return T

3.4.1 Nice properties of RRTs

The key advantages of RRTs are: 1) the expansion of a RRT is heavily biased toward unexplored portion of the states space; 2) the distribution, leading to consistent behaviour; 3) an RRT probabilistically complete under very general condition; 4) the RRT algorithm is relatively simple, which facilitates performance analysis (this is also a preferred feature of probabilistic roadmaps): 5) an RRT always remains connected, even though the number of edges is minimal; 6) an RRT can be considered as a path planning module, which can be adapted and incorporated into a wide variety of planning system; 7) entire path planning algorithms can be constructed without requiring the ability to steer the system between two prescribed states, which greatly broadens the applicability of RRTs.

3.5 PRM based on Obstacles Geometry

The method is based on obstacles geometric for addressing narrow corridors problems (Antonio & Vallejo, 2004). The method takes advantage of geometric properties for computing free configurations in both, first approximation and improving phase.

A geometric representation of the workspace (the obstacles and the robot) is given through triangle meshes, including their positions and orientations. Each object in the environment is associated whit its *straightness* and *volume*. The following defines volume and straightness feature for each object in the environment.

"Straightness" indicates the direction of an object with respect its longest side. We represents this property using a vector denoted as v_i and its represents the direction of the straightness of each body. During the construction phase, we use this vector as the most convenient direction of the rotation axis of an object B_i. The direction of the rotation axis is very important, because, when the algorithm attempts to rotate the robot, it assumes that, the best selection to rotate the body is around v_i.

The "**volume"** of the body gives an approximation of the size of an object respect to the volume of its surrounding sphere. This feature is given as parameter for each object within the environment. The value is proportional to the surrounding sphere.

It is important to take into account that our algorithm takes advantage of the form of the bodies. Provided that an obstacle is built by smaller ones, "*straightness*" and "*volume*" are defined for each element part.

3.5.1 First approximation of the configuration space

The objective of the construction step is to obtain a reasonably connected graph and to make sure that most "*difficult*" regions in this space contain at least a few nodes. In particular, the objective of the first approximation of the roadmap is to obtain an initial sampling using an economic and fast process.

The nodes of R should constitute a uniform sampling of C_{free}. Every such configuration is obtained by drawing each of its coordinates from the interval of allowed values of the corresponding DOF using a uniform probability distribution over this interval. This sampling is computed using spheres which surround the objects. Figure 7 illustrate these sampling surrounding spheres of objects. During this stage, the center of gravity and the radius are used to compute the size of the surrounding sphere of each body in the environment. Two advantages can be seen: the collision detection algorithm, (which is a deterministic used to decide if the position and orientation where the robot is placed does not collide with any object into the environment), is reduced to verify intersection between spheres. And the fact that these free configurations is involved into a sphere, having the possibility to rotate to any direction.

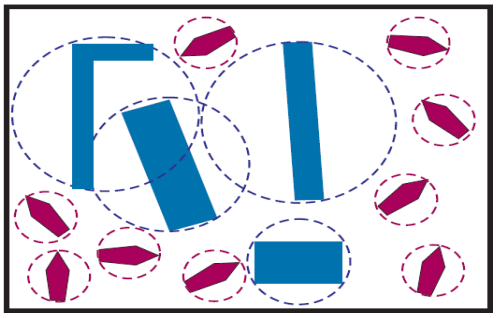


Fig. 7. The sampling of the roadmap is computed using a uniform distribution on the C-space and keeping the free configurations.

Collision detection is implemented using spheres, which means that we surround the robot and the obstacles within a sphere and the collision verification is reduced to compute the intersection among spheres. The computed configuration is labeled as *far configurations*). This property is used when the algorithm will tries to connect this configuration with its k-nearest neighbors.

3.5.2 Expanding the roadmap

If the number of nodes computed during the first approximation of the roadmap is large enough, the set N gives a fairly uniform covering of C_{free}. In easy scenes R is well connected. But in more constrained ones where C_{free} is actually connected, R often consists of a few large components and several small ones. It therefore does not effectively capture the connectivity of C_{free}. The purpose of the expansion is to add more nodes for facilitating the construction of the large components comprising as many nodes as possible for covering the most difficult narrow parts of C_{free}.

3.5.3 Elastic band algorithm

The "elastic band" algorithm attempts to find a free configuration from a collision one. To reach such goal, the algorithm moves the configuration using a small step for each iteration. This process is the result of applying the combination of both features (straightness and volume). First, the algorithm calculates the distance vector d_i between the obstacle position

and the configuration $c(B_i)$ position. Next, a value between the *MIN* and *MAX* parameters associated to the volume is computed and used as scalar quantity to increase or decrease the vector d_i. The following algorithm describes the operations used to compute the distance vector. Figure 8 presents a graphic illustration.
The main idea behind this scalar operation is to approach and move the robot away from the obstacle. To compute this operation, the process scales the d_i vector using the values computed with respect the *"volume"* feature to calculate the next position where the $c(B_i)$ will be placed. The following algorithm describes this process.

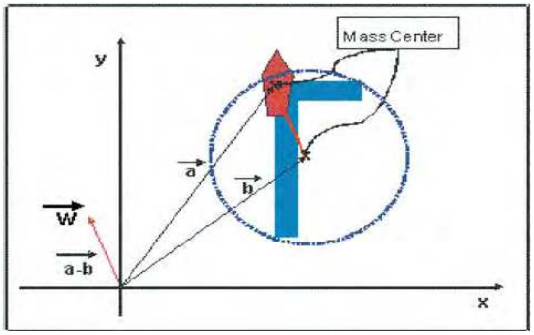


Fig. 8. Graphical representation of distance vector between the mass center of the Object and the mass center of the robot.

The elastic band process works with parallel and perpendicular configurations. Both types of configurations are computed around the obstacle. Configurations calculated in this phase are called *near configurations*. While the distance vector is computing the next configuration to be tested, the robot is rotated around its rotation axis, searching to find a free configuration and taking advantage of the "*straightness*", sweeping the minor volume as result of this rotation.

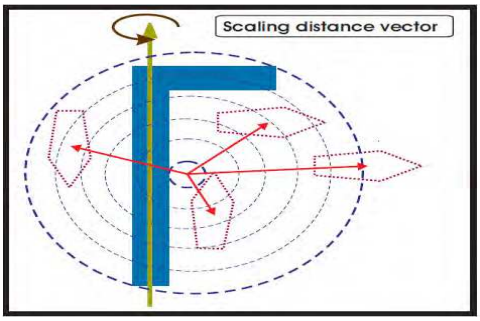


Fig. 9. Elastic Band approach and move the robot away from the obstacle to compute free configurations.

Figures 8 and Figure 9, shows how the parallel and perpendicular configurations are computed around the obstacle and how the scalar vector is changing, approaching and

moving away the robot from the obstacle. The configurations marked with dots are calculated during the elastic band process.

Elastic Band Heuristic

```
1.    B_i ← obstacle[i]
2.    q ← robot_configuration
3.    robot.get_parameters_obstacle ( MIN, MAX )
4.    c_i_init ← position_of(B_i)
5.     k ←− 0
6.    scalar ←− MIN
7.    d_i ←− distancebetween(c_i init, q)
8.    do
9.          scale(d_i,, scalar)
10.         q ←− get configuration on(d_i)
11.         q ←− rotate robot(q)
12.          k ←− k + 1
13.   while (q is in collision and k ≤CTE)
14.   if(q is free )
15.   N ←− N ∪ q
```

4. Collision Detection Algorithms

Collision detection is a fundamental problem in robotics, computer animation, physically-based modeling, molecular modeling and computer-simulated environments. In these applications, an object's motion is constrained by collisions with other objects and by other dynamic constraints. The problem has been well studied in the literature. A realistic simulation system, which couples geometric modeling and physical prototyping, can provide a useful toolset for applications in robotics, CAD/CAM design, molecular modeling, manufacturing design simulations, etc. In Figure 10. two sceneries are presented as a sample of environments that use collision detection. Such systems create electronic representations of mechanical parts, tools, and machines, which need to be tested for interconnectivity, functionality, and reliability. A fundamental component of such a system is to model object interactions precisely.

Fig. 10. 3-D Environments uses collision detection algorithms to restrict the movement ranks of several elements.

The interactions may involve objects in the simulation environment pushing, striking, or smashing other objects. Detecting collisions and determining contact points is a crucial step in portraying these interactions accurately. The most challenging problem in a simulation, namely the collision phase, can be separated into three parts: collision detection, contact area determination, and collision response.

4.1 Rapid version 2.01

RAPID is a robust and accurate polygon interference detection library for large environments composed of unstructured models (http://www.cs.unc.edu/~geom/OBB/OBBT.html).

- It is applicable to polygon soups - models which contain no adjacency information, and obey no topological constraints. The models may contain cracks, holes, self-intersections, and nongeneric (e.g. coplanar and collinear) configurations.
- It is numericaly robust - the algorithm is not subject to conditioning problems, and requires no special handling of nongeneric cases (such as parallel faces).

The RAPID library is free for non-commercial use. Please use this request form to download the latest version. It has a very simple user interface: the user need noncommercial use. Be familiar with only about five function calls. A C++ sample client program illustrates its use.
The fundamental data structure underlying RAPID is the OBBTree, which is a hierarchy of oriented bounding boxes (a 3D analog to the "strip trees" of Ballard). (Gottschalk et al., 1996).

5. GEMPA: Graphic Environment for Motion Planning Algorithms

Computer graphics has grown phenomenally in recent decades, progressing from simple 2-D graphics to complex, high-quality, three-dimensional environments. In entertainment, computer graphics is used extensively in movies and computer games. Animated movies are increasingly being made entirely with computers. Even no animated movies depend heavily on computer graphics to develop special effects. The capabilities of computer graphics in personal computers and home game consoles have now improved to the extent that low-cost systems are able to display millions of polygons per second.
The representation of different environments in such a system is used for a widely researched area, where many different types of problems are addressed, related to animation, interaction, and motion planning algorithms to name a few research topics. Although there are a variety of systems available with many different features, we are still a long way from a completely integrated system that is adaptable for many types of applications. This motivates us to create and build a visualization tool for planners capable of using physics-based models to generate realistic-looking motions. The main objective is to have a solid platform to create and develop algorithms for motion planning methods that can be launched into a digital environment. The developed of these tools allows to modify or to adapt the visualization tool for different kind of problems (Benitez & Mugarte, 2009).

5.1 GEMPA Architecture

GEMPA architecture is supported by necessary elements to represent objects, geometric transformation tools and visualization controls. These elements are integrated to reach initial goals of visualization and animation applied to motion planning problems.

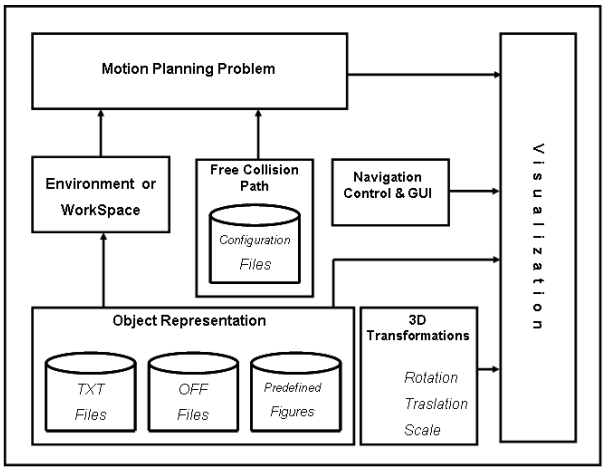


Fig. 11. Several modules are coupled to integrate the initial GEMPA architecture which offer interesting functionalities; visualization 3-D environments as well as animation of motion planning algorithms.

5.2 Recovering Objects Representation

People focus to solve problems using computer graphics, virtual reality and simulation of motion planning techniques used to recover information related to objects inside the environment through files which can storage information about triangle meshes. Hence, several objects can be placed on different positions and orientations to simulate a three-dimensional environment. There exist different formats to represent objects in three-dimensional spaces (3-D), however, two conventions used for many tools to represent triangle meshes are the most popular; objects based on *off - files* and objects based on *txt - files*. In motion planning community there exist benchmarks represented through this kind of files. GEMPA is able to load the triangle meshes used to represent objects from *txt* or *off – files.* On the other hand, GEMPA allows the user to built news environments using predefined figures as spheres, cones, cubes, etc. These figures are chosen from a option menu and the user build environments using translation, rotation and scale transformations. Each module on GEMPA architecture is presented in Figure 11 There, we can see that initially, the main goal is the visualization of 3-D environments and the animation of motion planning algorithms. In the case of visualization of 3-D environments, information is recovered form files and the user can navigate through the environment using mouse and keyboard controls. In the second case, the animation of motion planning algorithms, GEMPA needs information about the problem. This problem is described by two elements; the first one is called workspace, where obstacles (objects), robot representation and configuration (position and orientation) is recovered from files; the second, a set of free

collision configuration conform a path, this will be used to animate the robot movement from initial to goal configuration. An example of 2D environment can be seen in Figure 12.

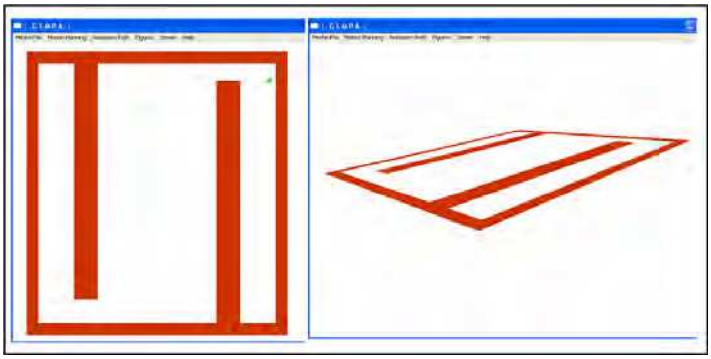


Fig. 12. Two different views of two-dimensional environment since the X-Y plane are painted.

5.3 GUI and Navigation Tools

GEMPA has incorporated two modes to paint an object; wire mode and solid mode. Next, Lambert illumination is implemented to produce more realism, and finally transparency effects are used to visualize the objects. Along the GUI, camera movements are added to facilitate the navigation inside the environment to display views from different locations. In Figure 13. Two illumination techniques are presented when GEMPA recover information since off-files to represent a human face.

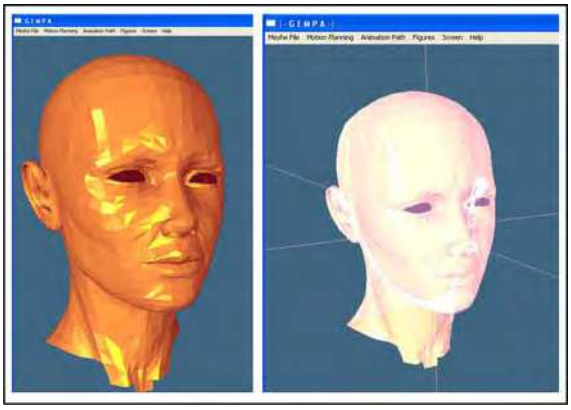


Fig. 13. Light transparency. In the left side, an object is painted using Lambert illumination, in the right side, transparency effect is applied on the object. Both features are used to give more realism the environment.

5.4 Simulation of Motion Planning Algorithms

Initially, only PRM for free flying objects are considered as an initial application of GEMPA. Taking into account this assumption, the workspace is conformed by a set of obstacles (objects) distributed on the environment, these objects has movement restrictions that mean that, the obstacles can not change their position inside the environment. In addition, an object that can move through the workspace is added to the environment and is called robot. The robot can move through the workspace using the free collision path to move from the initial configuration to the goal configuration. For PRM for free flying objects, only a robot can be defined and the workspace can include any obstacles as the problem need.

GEMPA also includes the capability to recover from an *environment – file* information about the position and orientation for each object inside a workspace including the robot configuration. Hence, GEMPA can draw each element to simulate the workspace associated. Therefore, initially GEMPA can recover information about the workspace, an example of this file can be see in Figure 14, where the environment file (left side), include initial and goal configuration for the robot, beside includes x,y,z parameter for position and (α, β, γ) parameters for orientation for every objects inside the workspace. Along with this *environment file*, a *configuration - file* can also be loaded to generate the corresponding animation of the free collision path. This *configuration - file* has the form presented in Figure 14 (right side). This file is conformed by n six-tuples (x, y, z,α, β, γ) to represent each configuration included in the free collision path.

Environment File	Configuration File					
	163					
Robot	5.0492	-0.7257	-3.2830	2.2233	6.1148	4.0886
robot_angular.txt	4.3394	-0.7601	-3.3466	2.1348	6.0969	4.1308
0.0 0.0 0.0 0.0 0.0 0.0	3.6292	-0.7945	-3.4102	2.0463	6.0791	4.1730
1.0 20.0 0.0 0.7 1.2 0.8	2.9189	-0.8289	-3.4737	1.9578	6.0613	4.2153
	2.2086	-0.8633	-3.5373	1.8692	6.0434	4.2575
Obstacle #1	1.4983	-0.8977	-3.6008	1.7807	6.0256	4.2997
bench_lamina_angosta_grande.txt	0.7881	-0.9322	-3.6644	1.6922	6.0078	4.3420
3.0 10 8.0 0.0 0.0 0.0	0.0778	-0.9666	-3.7280	1.6037	5.9900	4.3842
	-0.6324	-1.0010	-3.7915	1.5152	5.9721	4.4264
Obstacle #2	-1.3427	-1.0354	-3.8551	1.4267	5.9543	4.4687
bench_lamina_angosta_grande.txt	-2.0529	-1.0698	-3.9186	1.3382	5.9365	4.5109
3.0 10 -8.0 0.0 0.0 0.0		.				
End		.				
		.				
		.				

Fig. 14. On the left side, an example of environment - file (robot and obstacles representations) is presented, and on the right side a configuration file (free collision path) is shown.

Once GEMPA has recovered information about workspace and collision free path, the tool allows the user to display the animation on three different modes.

Mode 1: Animation painting all configurations.
Mode 2: Animation painting configurations using a step control.
Mode 3: Animation using automatic step.

From Figures 15 to Figure 18, we can see four different samples of motion planning problems which are considered as important cases. For each one, different views are

presented to show GEMPA's functionalities. Besides, we have presented motion planning problems with different levels of complexity.
In Figure 15. (Sample 1) The collision free path is painted as complete option and as animation option. In this sample a tetrahedron is considered as the robot.
Next, Figure 16: (Sample 2). A cube is presented as the robot for this motion planning problem. Here, GEMPA presents the flat and wire modes to paint the objects.
In Figure 17: (Sample 3). Presents a robot which has a more complex for and the problem becomes difficult to solve because the motion planning method needs to compute free configuration in the narrow corridor.
Finally in Figure 18: (Sample 4). Animation painting all configurations (left side), and animation using automatic step (right side) are displayed. Although the robot has not a more complex form, there are various narrow corridors inside the environment.

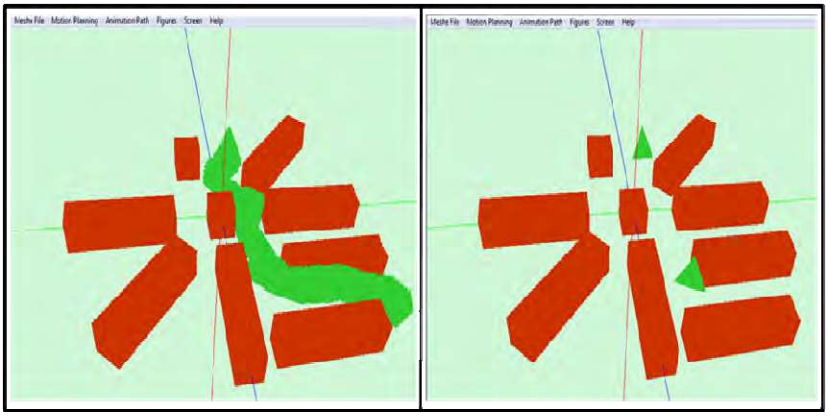


Fig. 15. Sample 1. The robot is presented as a tetrahedron.

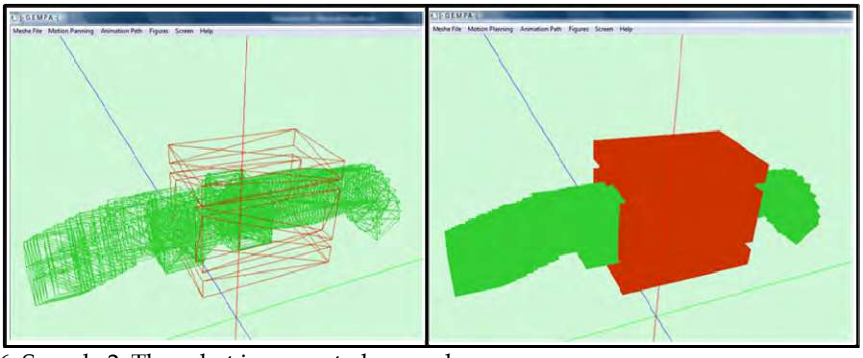


Fig. 16. Sample 2. The robot is presented as a cube.

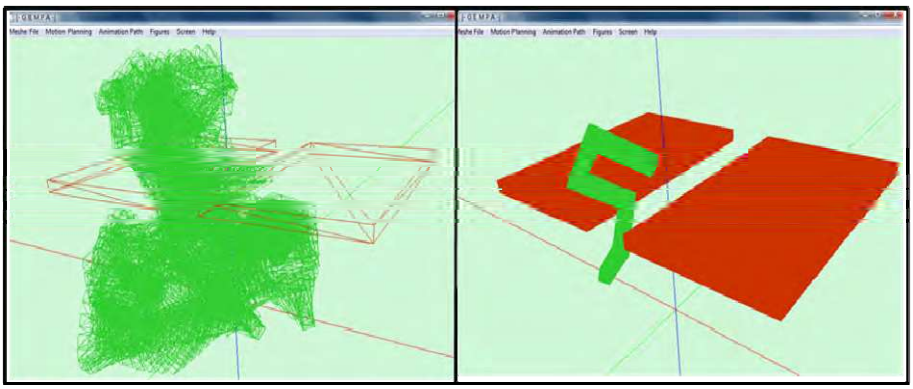


Fig. 17. Sample 3. The robot's form is more complex.

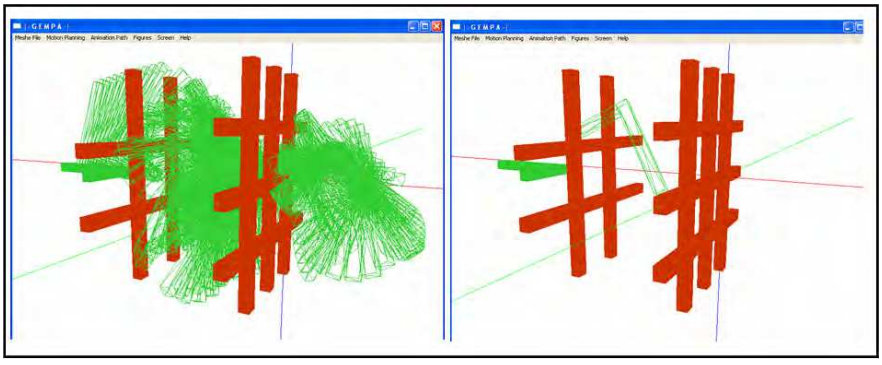


Fig. 18. Sample 4. More complex environment where various narrow corridors are presented.

6. References

Amato, N.; Bayazit, B. ; Dale, L.; Jones, C. &. Vallejo, D. (1998). Choosing good distance metrics and local planer for probabilistic roadmap methods. In in Procc.IEEE Int. Conf. Robot. Autom. (ICRA), pages 630–637.

Amato, N.; Bayazit, B. ; Dale, L.; Jones, C. &. Vallejo, D. (1998). Obprm: An obstaclebased prm for 3d workspaces. In in Procc. Int. Workshop on Algorithmic Fundation of Robotics (WAFR), pages 155–168.

Amato, N. M. & Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. In In IEEE Int. Conf. Robot. and Autom., pages 113–120.

Amato, N. Motion ning puzzels benchmarks.

Benitez, A. & Mugarte, A. (2009). *GEMPA:Graphic Environment for Motion Planning Algorithm.* In Research in Computer Science, Advances in Computer Science and Engineering. Volumen 42.

Benitez, A. & Vallejo, D. (2004). *New Technique to Improve Probabilistic Roadmap Methods.* In proceedings of Mexican International Conference on Artificial Intelligence. (IBERAMIA) Puebla City, November 22-26, pag. 514-526.

Benitez, A.; Vallejo, D. & Medina, M.A. (2004). Prms based on obstacle's geometry.In In Proc. IEEE The 8th Conference on Intelligent Autonomous Systems, pages 592–599.

Boor, V.; Overmars, N. H. & van der Stappen, A. F. (1999). The gaussian sampling strategy for probabilistic roadmap planners. In In IEEE Int. Conf. Robot. And Autom., pages 1018–1023.

Chang, H. & Li, T. Y. (1995). Assembly maintainability study with motion planning. In In Proc. IEEE Int. Conf. on Rob. and Autom., pages 1012–1019.

Christoph, M. Hoffmann. Solid modeling. (1997). In Handbook of Discrete and ComputationalGeometry, pages 863,880. In Jacob E. Goodman and Joseph ORourke, editors Press, Boca Raton New York.

Goodman, J. & O'Rourke, J. (1997). Handbook of Discrete and Computational Geometry. CRC Press.In Computer Graphics (SIGGRAPH'94)., pages 395–408.

Kavraki, L. & Latombe, J.C. (1994). Randomized preprocessing of configuration space for path planning. In IEEE Int. Conf. Robot. and Autom, pages 2138–2145.

Kavraki, L. & Latombe, J.C. (1994). Randomized preprocessing of configuration space for fast path planning. In IEEE International Conference on Robotics and Automation, San Diego (USA), pp. 2138-2245.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C. & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In IEEE Trans. Robot. & Autom, pages 566–580 .

Kavraki, L.; Kolountzakis, L. & Latombe, JC. (1996). Analysis of probabilistic roadmaps for path planning. In IEEE International Conference on Robotics and Automation, Minneapolis (USA), pp. 3020-3025.

Kavraki, L.E, J.C. Latombe, R. Motwani, & P. Raghavan. (1995). Randomized preprocessing of configuration space for path planning. In Proc. ACM Symp. on Theory of Computing., pages 353–362.

Koga, Y.; Kondo, K.; Kuffner, J. & Latombe, J.C. (1994). Planning motions with intentions.

Latombe, J.C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA.

Laumond, J. P. & Siméon, T. (2000). Notes on visibility roadmaps and path planning. In In Proc. Int. Workshop on Algorithmic Foundation of Robotics (WAFR), pages67–77.

M. LaValle and J. J. Kuffner. (1999). Randomized kinodynamic planning. In IEEE Int. Conf. Robot. and Autom. (ICRA), pages 473–479.

M. LaValle, J.H. Jakey, and L.E. Kavraki. (1999). A probabilistic roadmap approach for systems with closed kinematic chains. In IEEE Int. Conf. Robot. and Autom.

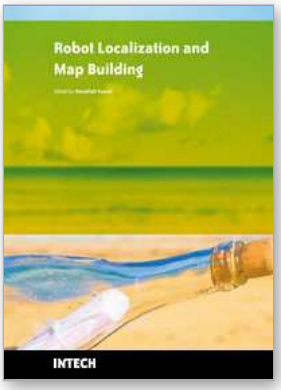
Overmars, M. & Svestka, P. (1995). A Probabilistic learning approach to motion Planning. In Algorithmic Foundations of Robotics of (WAFR94), K. Goldberg et al (Eds), pp. 19-37, AK Peters.

Overmars, M. & Svestka, P. (1994). A probabilistic learning approach to motion planning. In Proc. Workshop on Algorithmic Foundations of Robotics., pages 19–37.

Russell, S. & Norvig, P. (2003). Articial Intelligence: A Modern Approach. Pearson Education, Inc., Upper Saddle River, NJ.

Steven, M. LaValle. (2004). *Planning Algorithms*.

Tombropoulos, R.Z.; Adler, J.R. & Latombe, J.C. Carabeamer. (1999). A treatment planner for a robotic radiosurgical system with general kinematics. In Medical Image Analysis, pages 237–264.



Robot Localization and Map Building

Edited by Hanafiah Yussof

ISBN 978-953-7619-83-1
Hard cover, 578 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

Localization and mapping are the essence of successful navigation in mobile platform technology. Localization is a fundamental task in order to achieve high levels of autonomy in robot navigation and robustness in vehicle positioning. Robot localization and mapping is commonly related to cartography, combining science, technique and computation to build a trajectory map that reality can be modelled in ways that communicate spatial information effectively. This book describes comprehensive introduction, theories and applications related to localization, positioning and map building in mobile robot and autonomous vehicle platforms. It is organized in twenty seven chapters. Each chapter is rich with different degrees of details and approaches, supported by unique and actual resources that make it possible for readers to explore and learn the up to date knowledge in robot navigation technology. Understanding the theory and principles described in this book requires a multidisciplinary background of robotics, nonlinear system, sensor network, network engineering, computer science, physics, etc.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821