Towards a Fast Solution Method for the General Robot Motion Planning Problem using a Manhattan-like Distance Function on a Non-Uniform Grid in Configuration Space. (Proposal for Ph.D.Thesis)

Carl Van Geem
RISC-Linz
Research Institute for Symbolic Computation
Joh. Kepler University
A-4040 Linz, Austria
Internet: Carl.Van.Geem@risc.uni-linz.ac.at

December 13, 1993

Abstract

The generalized Piano Mover's Problem can be extended in several ways, e.g. by taking in account kinematics and dynamic constraints, by allowing obstacles to move slightly, by asking for a path that can be seen as a good pay-off between safest and shortest path, by planning with uncertainty,... All these extensions are usually treated separately from each other in the literature. We will consider a combination of some of these extensions and we will investigate a direction of research that seems to be suitable for such combination. The approach plans in configuration space and is related to the potential field approach. The potential function however shall be replaced by an L^1-distance-like function on a non-homogeneous higher-dimensional grid. This distance function shall be computed with a wavefront expansion algorithm. The approach seems to suit in a natural way to massively parallel implementation.

1 Introduction

This text gives a short overview of the field of Robot Motion Planning in all its facets and indicates a direction for further research in this area. It is as such my research proposal to the faculty of RISC, since I intend to work on the here proposed line, also with the goal to write a Ph.D. Thesis about this topic.

In the next section the Motion Planning Problem is specified explicitely. A typical example, also of historical importance, is the well known "piano mover's problem", where a robot (piano) has to be moved from one position to another without colliding with any obstacle (furniture standing in the way, getting the piano through the door or through an open window). Of course, this is the way to formulate the robot motion planning problem as a 'pure geometric problem', for which one can find plenty of solutions and algorithms, e.g. in [Schwartz, Hopcroft & Sharir 87], which are still not always applicable to real world problems.

In Section 3 we sketch the most important solutions for the geometric problem and a classification of solution approaches. In Sections 4 and 5 we describe in more detail the tools that will be useful in Section 6, namely the concept of 'Configuration Space' and the 'Potential Field Approach'. In Section 6, I present my ideas and directions for future research.

2 Problem Specification

A formal specification of the three-dimensional generalized piano mover's problem $(O, P, p_{start}, p_{goal})$ can be given as follows:

- the obstacle set O consists of a finite set of (rational) polyhedra $O_1, ..., O_{n_1}$,
- the object P to be moved, consists of a finite set of (rational) polyhedra $P_1, ..., P_{n_2}$, which are freely linked at distinguished linkage vertices $v_1, ..., v_{n_3}$,
- p_{start} and p_{goal} are distinguished initial and final rational positions of P

(see [Schwartz, Hopcroft & Sharir 87], page 273).

Motion planning in its most basic form, can be defined as finding a collision-free path for an object, or a collection of objects, among stationary objects ([Latombe 91a]). The objective of Robot Motion Planning is to automatically guide around a robot in an environment filled with obstacles and to choose forces to be applied when assembling objects ([Canny 88]). This basic form of the problem can be extended.

One could take into account the physical limitations of the robot. These are the so called *kinematic constraints*. Two types of constraints can be distinguished. The *holonomic constraints* can be expressed as equations which can be used to eliminate some parameters and thus reduce the dimension of configuration space (Cspace). Examples of such constraints are the limitations in moves of links of robot arms (e.g. they cannot rotate around completely, some limitations are present on the joints) or the limitations of moves of car-like robots (e.g. they cannot move sidewards). The *non-holonomic constraints*, which are much harder to deal with, can be expressed as non-integrable equations. They involve velocity and do not reduce the dimension of the Cspace, but of its velocity space.

One could not only plan the position of the robot on every moment of time, but also the speed and/or acceleration in function of time (the *dynamics*).

One could allow the *obstacles* to *move*. Since we want to plan the motion of the robot, it is reasonable to assume that the obstacles move only slightly.

One could also try to plan with *uncertainty*, i.e. taking into account the fact that during execution of the path, the robot will not follow the planned path exactly. A robot arm usually oscillates around the planned route. Due to the weight of the robot (gravitation), the arm also tends to bow downwards, also depending on whether or not the robot (robot arm) is carrying an object.

One could ask for finding a path (an arbitrary path), the shortest path, the safest path or for a path that establishes a pay-off between the shortest and safest path. It would be nice in all cases to find a path whenever there exists one. However, sometimes one uses heuristic approaches that do not guarantee this, while one hopes such an approach will always produce a path in real applications and within a reasonably small execution time.

In [Canny 88], the author states that the most difficult planning problems arise in assembly, where clearances between parts are typically smaller than the robots' positioning and sensing accuracy. A distinction is made between *gross and fine motion planning*. Fine motion planning is concerned with the planning for contact between the robot and an obstacle, e.g. when a gripper has to grap an object, while gross motion planning deals with moves that may be executed in a 'less careful' way, e.g. the robot might move slightly away from the planned path without colliding with an obstacle. Sensors of different types are sometimes attached to the robot in order to detect changes in the environment and as a safety measure.

Path planning can also be classified into two other categories: *on-line* path planning and *off-line* path planning. In the off-line case, more complex and time consuming algorithms can be used. Here planning for applications that are scheduled in the future can be done using simulation tools, while the robot in the meanwhile executes some other operation, which is currently needed ([Conte, Longhi & Zulli 93]). When planning on-line however, fast algorithms are needed, whenever the environment of the manipulator changes unforeseenly, for on-line updating of actual planning data in order to avoid wait states of the manipulator ([Fink & Wend 91]).

3 Solving the Problem ...

The geometric problem, in the previous section also refered to as the problem in its most basic form, is solved adequately by Schwartz and Sharir ([Schwartz & Sharir 83]), with a method of time complexity which is double-exponential in the size of the problem and by Canny ([Canny 88]), with a method of time complexity which is only exponential in the size of the problem.

An interesting introductory article on Motion Planning, giving a nice overview on recent developments in this field, is [Latombe 91a]. The author states that in essence there are two computational approaches to solve this problem: the Connectivity Representation approach and the Potential Field approach.

The *Connectivity Representation* approach consists of precomputing the connectivity of the space of all positions that are safe for the robot in the form of a non-directed graph, the so called connectivity graph, and then searching in this graph for a path. The methods of Schwartz and Sharir and of Canny are examples of this approach.

Schwartz and Sharir use the exact cell decomposition method (described in [Schwartz & Sharir 83]). Here the space of free positions in the configuration space is divided into a finite collection of semi-algebraic cells. These cells are computed by first finding the projection of the cells in a lower dimensional space and then lifting these projections again to the higher dimension. This can be done recursively, i.e. first going down from dimension m (usually bigger than 3) to dimension 1 and then lifting again. The method uses the well-known result in [Collins 75], for deciding the satisfiability of Tarski sentences.

In his Ph. D. Thesis ([Canny 88]), Canny introduced the so called roadmap method. The roadmap algorithm he presented, finds all components of a high-dimensional set S by computing its 1-dimensional skeleton (or roadmap) $R(S)$. The roadmap has the property being connected within each component of the original set S. To find a path between two arbitrary points p_1 and p_2 in S, one only has to find a path from each of these points to two points r_1 and r_2 nearby and on the roadmap and find a path along the skeleton. It can already be decided whether a path exists or not, when the two points r_1 and r_2 are found. If they do not lie in the same connected component, there will not be a path between p_1 and p_2 either. The roadmap is constructed by reducing the dimension of the problem by 1, taking slices of S by intersecting S with a 'moving' hyperplane. Extremal points in the slices trace out so called silhouette curves while the hyperplane moves. The silhouette curves form the essential part of the roadmap. On some critical moments however, one has to make sure that the roadmap is connected in a 1-1 correspondence with the connectivity of the original set S by adding some appropriate links. Doing so, the problem is reduced to several subproblems of searching paths in the slices of S, which have dimension 1 less than S.

Approaches based on Voronoi diagrams are also in this category.

The *Potential Field* approach makes the configuration space (i.e. the set of all possible configurations, whereas a configuration is a specification of the position of the robot relative to some fixed coordinate system) discrete, by turning it into a fine regular grid, and searches this grid for a free path. The search is then guided by a heuristic that always can be seen as a potential field.

The book of the same author ([Latombe 91b]) gives a thorough overview of the 'state of the art' of robot motion planning.

4 Configuration Space

Let R denote a set (the robot) and A_1,...,A_k a number of obstacles, pairwise disjoint. Let us introduce a coordinate system CS_W for the workspace W of R, with origin o a specific (arbitrary) point of R. A configuration of R is the image of R by a rigid body transformation t (also called positive isometry, i.e. preserving distance and orientation) from the workspace to some other space, which can be described as (o, q), where $q = t(o)$. The set of all such transformations is called the configuration space (also written as 'Cspace'). The dimension of the Cspace is equal to the number of degrees of freedom (DOF) of R in the workspace. A position p (or configuration) of R now corresponds to a point in Cspace. Indeed, a point of Cspace gives us the transformation of the reference point of R in the origin of CS_W to the position of the reference point of R when R is in position p. We will write R_q for the image of R under transformation t where $t(o) = q$. When one consideres a coordinate system CS_R moving with R and identical to CS_W when the reference point of R is at the origin of CS_W, the Cspace can also be seen as the set of all specifications of the position and orientation of CS_R with respect to CS_W.

A configuration q can be represented bijectively, depending on the choice of CS_R and CS_W, as a pair (t, r), where t determines the position of the origin of CS_R with respect to CS_W and where r determines the orientation of the axes of CS_R with respect to CS_W. Now t can be considered as the N-dimensional vector being the coordinates of the origin of CS_R with respect to CS_W, where N is the dimension of W, and r as the $N \times N$ matrix whose columns are the components of the unit vectors along the axes of CS_R with respect to CS_W. If we denote $SO(N)$ the special orthogonal group of the $N \times N$ matrices, then we have $r \in SO(N)$. In this way, the Cspace can be embedded in $\mathbb{R}^M$, with $M = N+N.N$. When we denote M_C for the image of $Cspace_R$ to the embedding in $\mathbb{R}^M$, we have:

$$M_C = \mathbb{R}^N \times SO(N) \subset \mathbb{R}^M.$$

4.1 The Cspace is a Manifold

It is not the aim to explain here in detail what a manifold is, nor to prove that the Cspace fits nicely in the theory of algebraic topology. In what follows, the reader should recognize the construction of a manifold structure in the special case of the Cspace. More details can be found in [Latombe 91b], Chapter 2.

The configuration space can be given a manifold structure now in the following way, using the embedding mentioned above. Let the topology of $Cspace_R$ be the subset topology induced by the Euclidean metric on $\mathbb{R}^M$, i.e. $U \subseteq M_C$ represents an open set of $Cspace_R$ iff there exists a $Y \subseteq \mathbb{R}^M$ such that $U = Y \cap M_C$. This topology on $Cspace_R$ can also be induced by other metrics (see Section 4.4). The properties of $SO(N)$ can be expressed by $\frac{1}{2}N(N+1)$ polynomial equations, such that locally M_C looks like $\mathbb{R}^m$, $m < M$. More precise, let $m = M - \frac{1}{2}N(N+1)$. Then every point $u \in M_C$ admits an open neighborhood U diffeomorphic to an open set X of $\mathbb{R}^m$, by the Implicit Function Theorem. The map

$$\phi : U \subset M_C \to \mathbb{R}^m : (u_1, ..., u_M) \mapsto (u_1, ..., u_m)$$

is a smooth one-to-one map of U onto an open subset X in $\mathbb{R}^m$, whereby the numbering of the coordinates of $u \in U$ is chosen appropriately according to the Implicit Function Theorem. If we now consider $f : Cspace_R \to M_C$ the bijective mapping sending each configuration q to its representation u in M_C, then considering an open subset U of $Cspace_R$ and a diffeomorphism ϕ of the image $f(U)$ in M_C onto an open subset X of $\mathbb{R}^m$, then the pair (U, ϕ) is a chart on U. This makes $Cspace_R$ into a manifold. The charts also have the nice 'glue-property', or more precisely, charts are C^∞-related.

4.2 Obstacles in Cspace

The $Cspace_R$-obstacle of A_i, the expanded obstacle K_i, is defined by:

$$K_i = CO_R(A_i) = \{x \in Cspace_R | R_x \cap A \neq \emptyset\}.$$

Obviously, x represents a safe position (a so called free position) for R if and only if x is not in $CO_R(A_i)$ for all obstacles A_i. Denote

$$C_{free} = Cspace_R \setminus \bigcup_j CO_R(A_i)$$

for the set of all free positions of R.

4.3 Planning in Cspace

The explicit use of configuration space for planning collision-free motions was first proposed by Lozano-Pérez (cf. [Canny 88] for a reference). The start and goal position of the robot in the workspace can now be described as the start configuration q_{start} and goal configuration q_{goal} (points) in Cspace (these points have to be in C_{free}, of course). And there exists a path between the start and goal position in the workspace if and only if q_{start} and q_{goal} lie in the same connected component of C_{free}. Planning a movement of the robot is thus reduced to finding a continuous path (curve) in C_{free}. A path of the robot R in $Cspace_R$ from q_{start} to q_{goal} can be defined as a continuous map $\tau : [0,1] \to Cspace_R$, with $\tau(0) = q_{start}$ and $\tau(1) = q_{goal}$.

The notion of direction in the manifold $Cspace_R$ can now be introduced by means of the *tangent space* in each configuration in the usual way. Since the tangent vector γ' to a differential curve $\gamma : [0,T] \to Cspace_R : t \mapsto \gamma(t)$ (where t is to be read as time and γ as a representation of a motion of R) represents the velocity at each configuration, the tangent space is also called *velocity space*.

4.4 Metrics

Several metrics (wherever they are) induce the same topology (which has been shown in the literature) on $Cspace_R$ as the one used above. Some planning methods use following distances as heuristic functions in order to guide the search process.

Since M_C is a subset of $\mathbb{R}^M$, the Cspace could also inherit the Euclidean distance from $\mathbb{R}^M$. For $q_1, q_2 \in Cspace_R$, mapping to $u_1, u_2 \in M_C$ respectively, by the bijective mapping f, the distance on $Cspace_R$ can be defined:

$$d(q_1, q_2) = \|u_1 - u_2\|.$$

Another way is using the Euclidean distance on the workspace W. Each configuration q corresponds to a position of R in W. The robot then fills up an area in W, a set $R_q \subset W$. For each point $a \in R$, the point with the same coordinates relative to CS_R is then notated $a(q) \in R_q$. A distance on C can thus be defined by

$$d(q_1, q_2) = \max_{a \in R} \|a(q_1) - a(q_2)\|,$$

if R is compact. Another candidate is:

$$d(q_1, q_2) = \sum_{k=1,..,N} \|a_k(q_1) - a_k(q_2)\|,$$

where the a_k's are N distinct linearly independent points in W (recall, $N = \dim(W)$). When R is compact, also the Hausdorff distance can be used:

$$d(q_1, q_2) = \max\{d_h(R_{q_1}, R_{q_2}), d_h(R_{q_2}, R_{q_1})\},$$

with

$$d_h(U, V) = \max_{u \in U} \min_{v \in V} \|u - v\|.$$

A third way makes $Cspace_R$ into a Riemannian manifold by defining an inner product on every tangent space $T_q(Cspace_R)$ in q. Denote $\|\xi\|$ for the norm of vector $\xi \in T_q(Cspace_R)$ associated with the inner product on the tangent space. Then the length of a smooth curve $\gamma[0,1] \rightarrow Cspace_R : s \mapsto \gamma(s)$ is well-defined by:

$$L(\gamma) = \int_0^1 ||\frac{d\vec{\gamma(s)}}{ds}||ds,$$

where $\frac{d\vec{\gamma(s)}}{ds}$ denotes the tangent vector of γ at s. The metric on $Cspace_R$ then becomes:

$$d(q_1, q_2) = \inf_{\gamma}\{L(\gamma)\},$$

over all paths γ from q_1 to q_2, where $Cspace_R$ has to be connected (say, only look at the component in which q_1 and q_2 both lie).

5 Potential Field Approach

In the potential field approach, in the configuration space the robot, represented as a point, is attracted from start position to the goal position by an attractive potential field, whereas the obstacles are modelled as repulsive potentials. A path is found by computing the gradients of the total potential and following the negative trend from the well towards the minimum point. Local minima are causing some inefficiency or failure of this approach. One can try to define a function without any local minima (or with very few) or one can find ways to escape out of the local minima when the robot gets trapped in them. When this approach is used as a heuristic, usually one cannot guarantee that a path will be found always if there exists one (since such a path might have to move in a direction opposite to the one 'downwards').

The potential field approach leads in a natural way to massively parallel implementation, using some kind of a grid to model the Cspace.

5.1 The Casual Potential Field

In general, the potential field approach treats the robot, represented as a point in Cspace, as a particle under the influence of a potential field U whose local variations are expected to represent the structure of the free-space. The force $\vec{F}(q) = -\vec{\nabla}U(q)$ induced by the potential field is considered as the most promising direction of motion at every place at any time. The potential field U is considered as the sum of an attractive potential, pulling the robot towards the goal position, and a repulsive potential, pushing the robot away from the obstacles.

In the following part of this section, a summary of the overview as it is given in [Latombe 91b] will be presented.

5.1.1 Settings

In the following we will use the notation C for the Cspace of the robot.

Given a differential potential function $U : C_{free} \to \mathbb{R}$, the artificial forces

$$\vec{F}(q) = -\vec{\nabla}U(q)$$

in C are induced by U, where $\vec{\nabla}U(q)$ stands for the gradient vector of U at q.

The potential U is constructed as the sum of two potentials U_{att} and U_{rep}:

$$U = U_{att} + U_{rep},$$

where U_{att} is the attraction potential associated with q_{goal} and independent from the C-obstacles, and U_{rep} the repulsive potential associated with the C-obstacle region.

Let us assume for simplicity that the Cspace is the whole space $\mathbb{R}^N$, for some N.

5.1.2 Attractive Potential

The attractive potential can be defined as a parabolic well:

$$U_{att}(q) = \frac{1}{2}\xi\rho^2_{goal}(q),$$

where ξ is a positive scaling factor and ρ_{goal} denotes the Euclidean distance between q and q_{goal} (if the Cspace is $\mathbb{R}^N$ for some N, say 2 or 3). Obviously, the implied attractive force $\vec{F}_{att}(q)$ is then

$$\vec{F}_{att}(q) = -\xi(q - q_{goal}).$$

This force converges linearly towards 0 when the robots' configuration gets closer to q_{goal}. The force also tends towards infinity when $\rho_{goal} \to \infty$.

In order to get rid of this last observation, the attractive potential can also be defined as a conic well:

$$U_{att}(q) = \frac{1}{2}\xi\rho_{goal}(q) = -\xi\frac{(q - q_{goal})}{||q - q_{goal}||}.$$

When $q \to q_{goal}$, the conic-well does not tend toward 0, however.

5.1.3 Repulsive Potential

The repulsive potential is introduced in order to create a 'potential barrier' around the C-obstacle region for the reference point of the robot in C-space. However, it would be nice if this repulsive field does not have any influence on the behaviour of the robot on positions which are 'sufficiently far away' from the obstacles.

This can, for example, be established by defining U_{rep} as follows: decompose the C-obstacle region $\mathcal{CB}$ into (possibly overlapping) convex components $\mathcal{CB}_k$, $k = 1, ..., r$, and associate to each component a repulsive potential

$$U_{rep,k}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho_k(q)} - \frac{1}{\rho_0})^2 & \text{if } \rho_k(q) \leq \rho_0, \\ 0 & \text{if } \rho_k(q) > \rho_0, \end{cases}$$

where η is a positive scaling factor, $\rho_k(q)$ is the Euclidean distance from point q to the C-obstacle $\mathcal{CB}_k$, i.e.

$$\rho_k(q) = \min_{q' \in \mathcal{CB}_k} ||q - q'||,$$

and ρ_0 is called the distance of influence of the C-obstacle region, and then the total repulsive potential field U_{rep} is the sum of these, i.e.:

$$U_{rep}(q) = \sum_{k=1}^{r} U_{rep,k}(q).$$

The artificial repulsive force is then

$$\vec{F}_{rep}(q) = \sum_{k=1}^{r} \vec{F}_{rep,k}(q)$$

where

$$\vec{F}_{rep,k}(q) = -\vec{\nabla} U_{rep,k}(q).$$

A drawback of decomposing the C-obstacle region is that several small components close to each other arbitrarily produce a combined repulsive force bigger than the one produced by a single bigger obstacle. A possible solution is to weight each individual potential according to the size of the obstacle generating it.

Of course, η and ρ_0 might also depend on the obstacle. In fact, when an obstacle is close to q_{goal}, the value of ρ_0 should be not too big in order not to prevent the robot to attain the goal position. When an obstacle is more threatening for the robot than the others, values for η and ρ_0 of this obstacle should be chosen larger.

5.1.4 For More General Cspaces

In case the Cspace is of the form $C = \mathbb{R}^N \times SO(N)$, for some N, some extensions to the notions above have to be made.

- The gradient $\vec{\nabla}$ has to be defined on the manifold, using charts of C.
- The Euclidean distance in the previous, is to be replaced by a metric d on C, i.e. $\rho(q) = d(q, q_{goal})$.

For more details, the reader is referred to [Latombe 91b], pp. 303-305.

Unfortunately, the computation of $\rho(q)$ and $\vec{\nabla}(q)$ for each q is difficult.

5.2 Path Planning, Using Potential Fields

Potential fields are of course just a model in which planning is done, the real planning has to use these potentials for actual path finding. Three simple path planning techniques are available for the potential field approach.

5.2.1 Depth-First Planning

This technique constructs a path as the product of successive path segments starting at the initial point q_{start}. Each segment is oriented along the negated gradient of the potential function computed at the configuration attained by the previous segment. The amplitude of the segment is chosen such that the segment lies entirely in free space. This approach only uses the potential field as such.

Let q_i and q_{i+1} be the origin resp. end extremities of the i-th segment in the path. Locally (in a chart of the Cspace-manifold, to be a little more precise), let $x_j(q_k)$, $j = 1, .., m$, denote the m coordinates of q_k, for arbitrary k, in the chart. Then

$$x_j(q_{i+1}) = x_j(q_k) + \delta_i \vec{t_j}(q_i),$$

with

$$\vec{t_j}(q_i) = \frac{\vec{F}(q_i)}{||\vec{F}(q_i)||}$$

and considered in the orthonormal basis on the tangent space induced by the chart:

$$\vec{F} = -\vec{\nabla} U$$

(for exact notation and details, see [Latombe 91b], p. 311), and δ_i denoting the length of the i-th increment (in Euclidean metric on $\mathbb{R}^m$). A good choice of the value of δ_i (small enough to be sure no collision with any of the obstacles occurs) gives us a path. It follows the steepest decent of the potential function and therefore it might get stuck in local minima.

5.2.2 Best-First Planning

For this technique, one first places a fine regular grid over the configuration space. It is assumed that q_{start} and q_{goal} are configurations in this grid, two neighbors in free space can be connected always by a straight line also in free space and the grid is bounded and forms a rectangloid.

First iteratively a tree T is constructed whose nodes are configurations in the grid and whose root is q_{start}. At each iterative step, the algorithm examines the neighbors of the leaf of T that has the smallest potential value, retains the neighbors not already in T at which the potential function is smaller than some large treshold M (so that one doesn't consider going up to some place where the potential function becomes infinite), and installs the retained neighbors in T as successors of the currently considered leaf.

The algorithm terminates when q_{goal} is reached (succes) or when all configurations accessible from q_{init} have been fully explored (failure). Each node has a pointer to its parent node, so if q_{goal} is attained, the path can be generated by tracing the pointers from q_{goal} to q_{init}.

This procedure follows a discrete approximation of the negated gradient of the potential function until it reaches a local minimum. It deals with local minima by filling them up until a saddle point is attained.

The approach is reported to be practical only when the dimension of the Cspace is small, say less than 5. The filling up of local minima is becoming too expensive for higher dimensions.

5.2.3 Variational Planning

In this approach a functional J of a path τ is constructed and then optimised over all paths. The optimisation of the functional is done by integrating the potential along a complete path between the starting and the goal position, using variational calculus.

With U a potential function, constructed for path planning, J could be defined as

$$J(\tau) = \int_0^1 U(\tau(s))ds + \int_0^1 ||\frac{\vec{dr}}{ds}||ds.$$

The first term in J is meant to produce the optimization a free path. The second term is optional and aimed to produce shorter paths. Other criteria can be encoded as well.

In [Suh & Shin 88] this approach is used for finding a path which is a payoff between the safest (more precise, a center line path, staying 'in the middle' between the obstacles: this might not be the same as the safest path) and the shortest path. Both travelling distance and robot safety are included in the cost function.

Since this method essentially consists of minimizing a functional J by following its negative gradient, it can get stuck in local minima, just like the depth-first approach. For large dimensions the optimisation can be quite costly. The advantage is the possibility to encode extra criteria in J.

5.3 The Global Navigation Function

5.3.1 Fluid Dynamics

Closely related to the potential field approach, is the fluid dynamics metaphor ([Keymeulen & Decuyper 90]). When a fluid is continuously pomped through a system of conduits, the fluid particles move on trajectories which constitute the shortest path connecting the pump and the outlet. Also when one of the pipes is suddenly closed, thus interrupting the flow, the fluid particles automatically adapt to this change in topology and find a new way. The solution for the motion planning problem (2-D) presented in [Keymeulen & Decuyper 90] consists in representing the factory's floor plan as a pipe-system with a fluid pump installed at the robots' initial position and an outlet at its destination. The robot itself is simulated as a fluid particle (a point). From some Physics-analysis of the model they deduct that a change in the obstacles should be modeled as waves propagating through the system. When a change is encountered, it is modeled automatically. The initial setting of the floor plan is modeled by prescribing boundary conditions on the equations that describe the fluid for each wall present.

The authors have implemented this model on a massively parallel array processor. The factory floor is represented as a grid of space cells each containing some fluid or a piece of wall. All cells compute in parallel the solution of the wave equation. After some initial time, the solution becomes stable and the moves of particles can start. There is some local communication, exchanging some local information. The implementation done by the authors of the report is said to evolve in real time and to adapt quickly to changes in topology.

5.3.2 Vortex Fields

Another approach to prevent local minima, is the use of vortex fields instead of repulsive potentials. In [De Medio & Oriolo 90] the authors explain how the vortex fields force the robot (reference point in Cspace) to turn around the obstacles. Repulsive potentials make sure that no collision will appear, while vortex fields guide the robot around the obstacles and can be considered to be similar to a velocity field in fluid dynamics. For simplicity reasons, in [De Medio & Oriolo 90] only planning in 2-dimensional space is described. Suppose that one has a point obstacle located in the origin of the Euclidean xy-coordinate system. The vortex field for this point obstacle is then defined as:

$$V(p) = \begin{bmatrix} V(x) \\ V(y) \end{bmatrix} = \begin{bmatrix} \frac{-y}{x^2+y^2} \\ \frac{x}{x^2+y^2} \end{bmatrix}.$$

This field assigns in each point p a vector which is tangent to the circle with center in the origin, is oriented in counterclockwise direction, and has modulus inversily proportional to the square of its distance to the point obstacle, so to rapidly decrease as p 'moves away' from it. Adding such vortex fields to the attractive potential field, will establish a total potential without local minima, and the vortex fields will garantee obstacle avoidance.

Three remarks where made by the authors. First, to stop the robot turning around the obstacle for ever, a vortex remover procedure is introduced. The angle between the direction of the vortex vector and the attractive vector becomes zero when the obstacle is passed. Second, the selection of the direction of vortex rotation has to be done because it influences the length of the path and in some cases (when moving a manipulator with fixed basis is planned) a bad choice can even lead to not finding an existing path. In the latter case one can choose for the direction which generates moves towards the robot base. Third, in the case of articulated manipulators, link interaction has to be taken into account. The links will also feel an influence by the vortex field.

5.3.3 L^1-distance

Another navigation function is described in [Latombe 91b], using a so called wavefront technique. In fact, the Cspace is modeled as a uniform grid, where some cells are 'filled' by the obstacle and where the 'empty' cells represent a free position in Cspace. At initialisation time, the 'filled' cells are labeled with '1' and the others with '0'. It is assumed that q_{start} and q_{goal} both are in an 'empty' cell. The navigation function used here is the L^1 distance, also called Manhattan distance, to the cell with goal position in it. This distance function U can easily be computed by a so called 'wavefront expansion' technique. At first, U is set to 0 at q_{goal}. In the next step, in all neighboring empty cells U is set to 1. Then recursively all empty neighbors of cells with $U = d$, except for those which already have a value, will get value $d + 1$ for U. When q_{start} is reached, a path can be planned by following the decent of distance.

A drawback of this method is that the path always grazes the obstacles, like the visibility graph approach in the pure geometric version of the planning problem. To improve this, a skeleton can be constructed first. This skeleton S has one dimension less then the Cspace and is a subset of the set of free positions. Starting from the boundary of this free space, the L^1 distance from each position in free space to the boundary is computed using the wavefront approach. Then S is the set of positions where these waves meet, i.e. in the 'middle' between the obstacles. This looks somewhat like the Voronoi diagram approach in the pure geometric problem. Obviously, the complexity of the L^1-wavefront approach is linear in the number of empty cells in the Cspace. Therefore in the higher dimensional Cspaces this approach is used with some modification. One chooses some points in the workspace of the manipulator. With each of these points a potential field is associated in the set of free placements in the workspace. Then the potential function U is computed in the Cspace in those cells where it seems reasonable to have U (the number of such cells is much smaller than the total number of cells), with an arbitration function depending on the values of the potential fields in the workspace. Since the workspace is 2 or 3 dimensional, it is reasonably easy to compute the potential fields. Since the number of cells in Cspace is smaller, planning can be done more efficiently also in high dimensional Cspaces.

We refer the reader to [Borgefors 86] for a thorough study of different possible definitions of 'neighbors' for the construction of the Manhattan distance on the uniform grid in 2 dimensional space by the wavefront approach (at least, one could interprete the paper in such a way). The author compares the geometric behaviour of the expansion of the area of cells to which a distance value is assigned.

The wavefront expansion algorithm above has been implemented in a way that reduces the memory needed for the representation of Cspace and some reduction of the complexity of the problem using a hierarchical multilevel discretisation and a multisource navigation function, as a part of a simulation system, so is reported in [Conte et al. 93]. Doing a step in Cspace along one of the directions of a grid means that the robot moves along only one of the DOF's at a time (e.g. only one joint of a robotarm changes position at each moment), as is pointed out in [Conte, Longhi & Zulli 93]. Visiting smaller cells will result in a higher number of changes in moves of the point in Cspace and thus in the follow-up of joints that are moving.

In the following picture a uniform grid over a two-dimensional Cspace with two obstacles (triangles) is given, and the Manhattan distance is filled in in the grid cells.

2	1	2	3	4	5	6
1	$0, q_{goal}$	1	2	3	4	5
2	1	2	3	4	5	6
3	2	3	4			7
4	3	4	5			8
5			6	7	$8, q_{start}$	
6		8	7	8		

6 Future Plans: Prospects and Ideas

A direction for future research is the combination of certain extensions to the geometric problem. Various combinations of problems which have so far been considered separately, (e.g. planning with kinematic constraints *and* uncertainty *and* moving obstacles) will have to be explored.

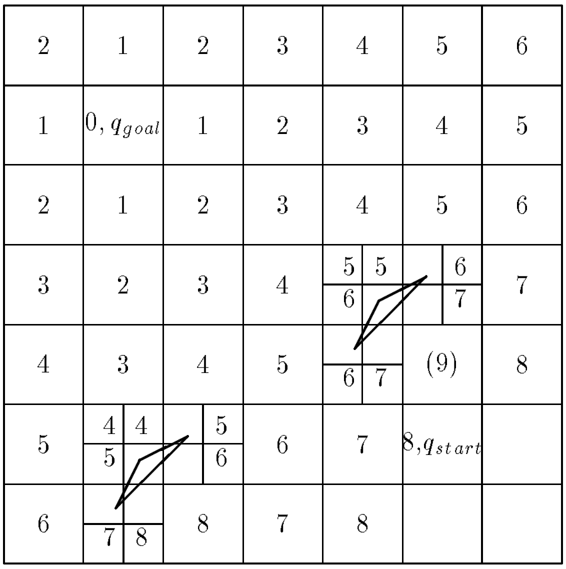
Also it makes sense to look for a pay-off between the shortest path and the safest path. Usually, in real applications one is interested in safe and fast paths, not so much in the path that is theoretically the fastest or theoretically the shortest. It is known for example that for the geometrical problem the visibility graph solution gives us the shortest path and the Voronoi diagram solution a safe path. Similarly, the wavefront approach exists in two versions (see above). Striking such a compromise between distance minimalisation and safety, is done in [Suh & Shin 88]. The authors state that due to uncertainties in robot dynamics during path execution, it is important to consider such a pay-off. The authors describe a 2D case for a robot represented as a disc and a 3D extention of the method for a robot represented as a sphere. A larger path, a so called channel, is computed between the obstacles.

On the one hand, the approach with the wavefront algorithm and the Manhattan distance is a complete solution (i.e. always finds a path if there is one and reports failure otherwise) to the Motion Planning Problem and a variant of the potential field approach, but seems to have the drawback of the size of the grid, since it grows exponentially with the dimension of the Cspace. On the other hand, the potential field approach has had reportedly good performance for high dimensional Cspaces, e.g. [Aylett& Milton 91] and [Barraquand & Latombe 91] (up to 31 DOF's). Here, the planner uses the good properties of the potential function and escapes from the local minima using efficient methodes like a Monte Carlo technique. Potential fields also seem to react on slightly moving obstacles by a small perturbation which stabilizes after a short time (cf. [Keymeulen & Decuyper 90]).

Also, the Euclidean distance can be seen as the limit of the Manhattan distance when the uniform grid is refined infinitely (when the size of the grid-cells shrinks toward 0) and thus the potential attraction field of Section 5.1.2 can be seen as the limit of the Manhattan distance field.

If we want to use some grid in order to model a Cspace of dimension m bigger than 2 or 3 (e.g. up to 31), we would like to reduce the size. The 2^m-tree decomposition (for $m = 2$ it is also called a quadtree, for $m = 3$ an octree) is a well-known representation for the Cspace, and has the advantage that it is a finer grid nearby obstacles and a rougher grid where there are no obstacles around. It also needs less cells for representing the Cspace. When a Manhattan-like distance is 'filled in' in the grid-cells of the quadtree, one easily sees that, when the path along the cells is constructed in the straightforward analogous way, this path preferes to develop using the 'big' cells which are further away from the obstacles. This is an indication to the suspicion that non-uniform grid representations with a 'distance field'-construction on them, lead to an approximation of the sum of an attractive and repulsive potential fields, similar to the approximation of the Euclidean distance by the Manhattan distance on a unifirm grid.

The following picture of the same Cspace as in the previous section gives an idea about the way in which the Manhattan-like distance could be filled in in the non-uniform grid. (In this picture the Manhattan-like distance is filled in according the rule that an empty cell in the non-uniform grid will get an induced value only from a neighboring cell of the same size or of a bigger size.)



The path generated by the quadtree and the Manhattan-like distance is still fast, since it has similar properties as the path generated by the wavefront approach described above. I.e., the path will go along the cells in such a way that the current cell at any time on the path will have a "distance" 1 less than the distance in the previously visited adjacent cell. Since the representation is done by a non-uniform grid, the size of the cells along the path gives us some information about the allowed error during the execution of the path. So also uncertainty seems to be incorporated in a natural way in this proposed approach. In fact, it looks as if we allow a neighbor of size one level lower (smaller) in the tree to induce the value of U, then we also allow the robot (point) to come closer to an obstacle (and thus to follow a faster but less safe path) by one format of cells. One could make the observation that the behaviour of the Manhattan-like distance near obstacles is similar to the velocity of water particles in a river, where the particles close to the river bank move slower than the particles in the middle of the river.

The L^1-distance-like function can be extended to higher dimensions, and computed in a similar way as the L^1-distance with the wavefront approach, with some adaptations nearby the obstacles, where the size of the cells becomes smaller and the number of neighbors bigger.

6.1 Some Tools

A compact data-structure is proposed and implemented at RISC ([Burger 92]) for the quad- and octree representation starting on a grid instead of from one single node.

The Potential Field approach, and the grid representation leeds in a natural way to massive parallelism. Implementations are reported, e.g. in the paper [Keymeulen & Decuyper 90].

The representation of the Cspace seems to influence the planning with Manhattan-like distances, so a more thourough study of the literature on this subject might be useful. Some literature surveys are at hand, like [Dittenberger] (also on the model of obstacles in Cspace), [Latombe 91b], [Burger 92], and various journals and books on geometric modelling.

References

[Aylett& Milton 91] R.Aylett & C.Milton, Achievable Autonomy in Next-generation Robots, ESA/ESTEC AI and KBS in Space Workshop, Noordwijk, May 1991, ©ARRL, May 1991.

[Barraquand & Latombe 91] J.Barraquand & J.-C. Latombe, Robot Motion Planning: A Distributed Representation Approach, The International Journal of Robotics Research, Vol.10, No.6, December 1991.

[Borgefors 86] G. Borgefors, Distance Transformations in Digital Images, Computer Vision, Graphics, and image processing 34, pp. 334-371, 1986.

[Burger 92] K. C. Burger, A Kernel-Based Construction System (KBC), diploma thesis, RISC-Linz, July 1992.

[Conte et al. 93] G. Conte, T. Leo, S. Longhi and R. Zulli, Emulation of Articulated Robots, Preprints of 2nd Int. Workshop on Robotics in Alpe-Adria Region (RAA'93), June 13-15, 1993, Krems, Austria.

[Conte, Longhi & Zulli 93] G. Conte, S. Longhi and R. Zulli, CAD-Based Generation of Collision-Free Paths for Robotic Manipulators, (preprint, will be published in Control Engineering Practice, Dec.1993).

[Canny 88] J. F. Canny, The Complexity of Robot Motion Planning, MIT Press, 1988.

[Collins 75] G. E. Collins, Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition, Lect. Notes in Computer Science, 33, Springer Verlag, New York, pp. 135-183.

[De Medio & Oriolo 90] C. De Medio & G. Oriolo, Robot Obstacle Avoidance Using Vortex Fields, (proceedings of) Advances in Robot Kinematics, S. Stifter & J. Lenarčič (eds.), Springer-Verlag, 1991.

[Dittenberger] K. Dittenberger, dissertation, Graph Decomposition and Retraction: An Approach to Collision-Free Path Planning, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, NY 12180.

[Fink & Wend 91] B. Fink & H. -D. Wend, Fast Collision-free Motion Planning for Robot Manipulators Based on Parallelised Algorithms, Preprints, Symposium on Robot Control (Syroco) '91, Wien, 16-18 Sept. 1991.

[Fu & Lee 91] J.J. Fu & R.C.T. Lee, Voronoi Diagrams of Moving Points in the Plane, International Journal of Computational Geometry and Applications, Vol.1, No.1, 1991, pp. 23-32.

[Keymeulen & Decuyper 90] D. Keymeulen & J. Decuyper, A Reactive Navigation System Based on a Fluid Dynamics Metaphor, AI MEMO nr 90-5, Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Belgium, 1990.

[Latombe 91a] J. -C. Latombe, Motion Planning: Theory and Applications, pp. 23-32 in Computer Applications in Production and Engineering: Integration Aspects, Elsevier Science Publishers B.V. (North Holland), 1991.

[Latombe 91b] J. -C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991, Boston.

[Schwartz, Hopcroft & Sharir 87] Planning, Geometry, and Complexity of Robot Motion, eds. J. T. Schwartz, J. Hopcroft & M. Sharir, Ablex Publishing Corporation, Norwood, New Jersey, 1987.

[Schwartz & Sharir 83] J. T. Schwartz & M. Sharir, "On the 'Piano Movers' Problem", in [Schwartz, Hopcroft & Sharir 87] pp. 51-96.

[Suh & Shin 88] Suk-Hwan Suh & Kang G. Shin, A Variational Dynamic Programming Approach With a Distance-Safety Criterion, IEEE Journal of Robotics and Automation, Vol.4, No.3, June 1988.