Path Planning for Manipulators with Many Degrees of Freedom Using a Discrete Representation of Configuration Space Extensively.*

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

October 1995

Abstract

Keywords: Robotics, Computational Geometry, Data Structures.
In order to plan a short and safe path for a manipulator, given a fixed initial configuration, to any configuration as goal position, we propose to approximate the Configuration Space (Cspace) of the manipulator by a non-uniform grid, built up in a hierarchical way, adding one dimension to the Cspace in each step. During this hierarchical procedure a discrete attractive potential field on the grid is extended into the new dimension. We use the so called GET-Representation for the grid. The potential field can be adapted locally when slight changes in the positioning of objects occurs after the complete construction of grid and potential field. Thus our representation does not have to be recomputed completely when unexpectedly some slight changes in the environment are observed. In this paper we describe the construction of the GET-Representation using the fast Roider collision checking algorithm and we describe techniques necessary for the adaptation of the data structures we use for grid and potential field when slight changes to the positioning of obstacles have to be made.

*submitted for publication

1 Introduction.

For the robot motion planning problem for a manipulator with many degrees of freedom (DOF) that we want to solve a fixed initial configuration is given, while any configuration is assumed to be a candidate for the goal configuration.

Each point in the Configuration Space (Cspace) corresponds to a configuration of the robot in its Workspace and thus the robot motion planning problem can be solved in two steps. In the first step the free space in Cspace is computed. In the second step a path is planned for a point in Cspace. It is very time consuming to compute the Cspace of a robot with many DOFs exactly. For this reason we suggest to approximate the free space in Cspace by a non-uniform grid. The robot motion planning problem we want to tackle here demands the complete space of free placements in Cspace to be modelled. By doing so it becomes possible to compute for a given fixed initial position of the manipulator a path to any goal position.

In our approach we put an 'attractive potential field' on the non-uniform grid on Cspace, using the wavefront expansion technique as it is done in [Barraquand & Latombe 91] on a uniform grid. We organise the cells of the non-uniform grid as in a 2^k-tree (called quadtree for $k = 2$ and octree for $k = 3$), where k is the dimension of Cspace. We use the compact so called GET-Representation (GET-Rep) for 2^k-trees due to [Burger 92]. For the wavefront expansion we have to find all neighbouring cells of a cell. We have observed that the algorithms given in [Gargantini 82] can be generalised to any 2^k-tree and can be translated in simple algorithms for manipulating the GET-Rep (see [Van Geem 94]).

The model of the Cspace allows small changes in the position of obstacles (i.e. when an obstacle is not positioned exactly on the spot we assumed it to be in in the Workspace). In this way the computation of the free space in Cspace can be considered as a pre-processing step that can be executed in reasonable time. Moreover, the path is a good pay-off between a safe and a short path.

Good results are reported in [Conte, Longhi, Zulli 93] and [Verwer 90] using cells of different sizes for finding a path between two configurations and with similar techniques as ours. New results decribed in this paper are the construction of the GET-Rep using a fast collision checking algorithm due to [Stifter 91], and the techniques for adapting the GET-Rep and the 'potential field' on the grid when small changes have to be made. The way in which a GET-Rep can be extended in each hierarchical step, and the hierarchical wavefront extension with k different increments were already described in [Van Geem 95b], but are integrated here in their natural context.

In the next section we give some motivations for our approach. In Section 3 we briefly describe the data structures we use. In Section 4 we give our approach for constructing the non-uniform grid over Cspace and in Section 5 we concentrate on the construction of the 'potential field'. In Section 6 we give the algorithm for adapting the GET-Rep and the 'potential field' when small changes in the representation have to be made. In Section 7 we briefly discuss our path planning approach, especially the relation between the 'potential field' on the grid and the resulting path.

2 Motivations.

We use a non-uniform grid to model the free space in Cspace. This means that we use small sized cells near the boundary of the Cspace obstacles and larger sized cells further away. Compared to a uniform grid we reduce the total number of cells. The reduction depends heavily on the position of the obstacles and of the form of their boundaries.

In contrast to [Verwer 90], since we construct a model for a large part of, or even for the whole highly-dimensional Cspace, we need to make extensive use of a fast and accurate collision checker, which is provided by the Roider method collision check for superellipsoids.

Combined with the 'potential field', the non-uniform grid makes a repulsive potential superfluous. The value of the attractive potential field on cell C indicates among other things the minimal number of cells to be traversed from the initial configuration to C. This means that the path along the descent from goal configuration to initial configuration will preferably go along large cells, which are usually rather far away from the Cspace obstacles. The path is thus a good pay-off between short and safe path. This was already observed in [Van Geem 94] for a potential field having as value for cell C exactly the minimal number of cells to be traversed between the cell enclosing the initial configuration and C.

For our purposes it is necessary to have a good adaptability of the data structure for both the grid and the potential field, when an obstacle has been observed on a slightly different position than originally expected. The GET-Rep of the 2^k-tree can be adapted easily when cells are merged to a larger sized one or when a cell is split up in smaller sized cells. The adaptation of the potential field is similar, but some local waves and sometimes some larger scaled waves are necessary for updating the values. However, while updating on larger scale, a minor adaptation can be done first, which allows us to find a path of somewhat poorer quality.

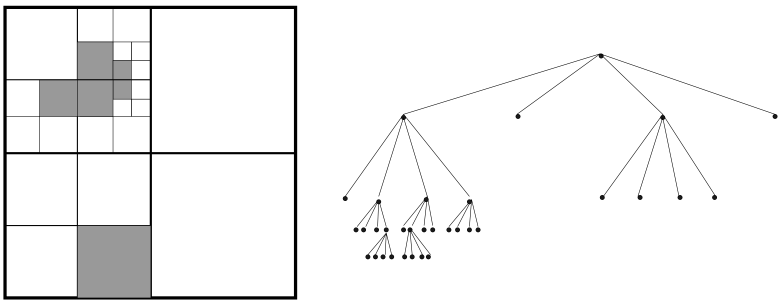
3 Data Structure.

A quadtree is a tree representation of a partition of an area into smaller subareas, possibly of different sizes. Let be given a rectangle divided in smaller rectangles, where the partitioning is done by cutting a rectangle into 4 smaller rectangles of equal size. Then we represent this partitioning by a tree, where each node corresponds to a particular rectangle. The root corresponds to the original rectangle. Each child of a node n to one of the 4 smaller rectangles in which the rectangle corresponding to n was divided.

An octree is the straightforward generalisation of the quadtree to the 3-dimensional case. Here one starts with a cube and partitions it into 8 smaller cubes of equal size. In this case, every parent node will have 8 children. A 2^k-tree is the straightforward generalisation of the quadtree to the k-dimensional case.

A very compact representation of quad- and octrees, the so called GET-Rep, can be found in [Burger 92]. Starting from the tree, where we assume that the children of each node are ordered according to the position of the respective cells, here one constructs two lists with binary entries. The first list, called DFA [1], is constructed by running through the tree in a breadth-first transversal order. When passing through a node, a '0' is added to the list when the node is a leaf, and a '1' when the node has children. An important observation is that the subdivision of a cell into 2^k smaller cells corresponds to a 2^k-tuple of entries in the DFA. Each leaf corresponds to a cell. If the cell contains a forbidden configuration, a '0' is put in the second list with binary entries (referred to as DFB), and a '1' for the free cells. The list of values for the potential field is organised as the

[1] Stands for 'Datafield A' in [Burger 92].



DFA: (1) 1010 0111 0000 0001 0000 0100 0000 0000

DFB: 00 00001 0010100100 00101000

Figure 1: A 2-dimensional subdivision, its quadtree, and its GET-Rep.

DFB list, but with integer or rational numbers as entries. In Figure 1 a 2-dimensional example is drawn.

Two k-dimensional cells in the non-uniform grid are called neighbours if there is a hyperplane that contains a face of each of them. Because of the ordering according to the position of the respective cells, the GET-Rep has similar properties as the representation for quad- and octrees designed by [Gargantini 82] for the purpose of finding the neighbouring cells scattered about the tree. The algorithms for neighbour computation in [Gargantini 82] can be translated to algorithms for neighbour computation in GET-Rep, as described in [Van Geem 94].

4 Grid Construction.

We can easily merge two GET-Rep's for the same manipulator and two different obstacles, as we will describe in the next subsection. We propose to compute GET-Rep for each obstacle in parallel and to merge the resulting GET-Rep's also in parallel with a divide-conquer-combine strategy. For the same reason we can also restrict ourselves to convex obstacles. In the following subsections we will concentrate on the computation of a GET-Rep for just one convex obstacle.

4.1 Merge Two GET-Rep's.

For the computation of the merge of two Get-Rep's, we use the following algorithm. We consider a function dfaTOdfb() to be given, computing for entry i in a DFA, the corresponding entry in the DFB. This function is easy to write.

For both DFA's, each entry in one of them has to correspond to an entry in the other. If such a corresponding entry does not exist we introduce a dummy entry: \$ (in algorithm: [A]).

When a large cell occupied by an obstacle is discovered in one of the DFA's by checking in the DFB's, all children in the 2^k-tree corresponding to the other DFA have to be removed. We replace them by dummy 2^k-tuples, consisting of \$-entries only (in algorithm: [B]).

The resulting DFA entry is computed with a logical OR. For the dummy entry \$ we define: \$ OR \$ = \$, \$ OR 1 = 1, \$ OR 0 = 0, 1 OR \$ = 1, and 0 OR \$ = 0 (in algorithm: [C]).

IN: (DFA1,DFB1),(DFA2,DFB2),

OUT: merge (DFAr,DFBr).

A: introduce dummy 2^k-tuples

*: for (i=0; i < max(length(DFA1),length(DFA2)); i++)

  #: if (((DFA1(i)==0) and (DFB1(dfaTOdfb(i))==1)) or ((DFA2(i)==0) and (DFB2(dfaTOdfb(i))==1)))

    *: DFAr(i)=0;

    *: DFBr(dfaTOdfb(i))=1;

    B: adapt next levels: replace in DFA2 or DFA1 resp. the corresponding 2^k-tuple by \$...\$ (2^k times).

  #: else

    C: DFAr(i)=(DFA1(i) OR DFA2(i));

    *: if (DFAr(i)==0)

      · DFBr(dfaTOdfb(i))=(DFB1(dfaTOdfb(i)) OR DFB2(dfaTOdfb(i)));

*: return (DFAr,DFBr);

4.2 Hierarchical Model of Robot Arm.

In order to construct the non-uniform grid on Cspace, we propose a hierarchical method on the robot arm. In each step we consider only one link, starting with the link closest to the base support. In case the first joint is a rotational one and the first link is modelled as a cylinder then for a convex polyhedron as obstacle we can approximate easily the extreme still acceptable values for the angle of the first joint also by a projection method as described in [Van Geem 95a].

For the other links we propose to approximate them by superellipsoids. Superellipsoids are well-known primitives in solid geometric modelling and enclose objects that are rather symmetric with respect to an axis very accurately (see [Barr 81] and [Zarrugh 85]). The Roider method, introduced in [Roider & Stifter 87a] and [Roider & Stifter 87b] for 2D and generalised to 3D in [Stifter 88a], [Stifter 88c], and [Stifter 92], allows a very fast collision checking for two convex objects and especially also for two superellipsoids or for a polyhedron and a superellipsoid (see [Stifter 91]).

In [Stifter 91] a method is given for enclosing polyhedrons (amongst other geometrical objects, see also [Stifter 88b]). Therefore it suffices to approximate the usually non-convex swept volume by a polyhedron. Fortunately most of the area where the swept volume is locally concave is already swept by the previous link. Since we only want to check for collisions of the i-th link in the i-th step, we only consider the cells that are free of collisions for the first $i-1$ links and we will only observe collisions or near-collisions for the swept volume of the i-th link. In some cases we do not have to consider the whole swept volume of the i-th link, since it might happen that a previously studied link has swept part of the area already. When a robot arm is given, we propose to study in a similar way the swept volumes of the links of this robot. This only has to be done once and can be considered as a preprocessing step. The kinematics of robot arms has already been studied in a general way, e.g. [Gorla & Renaud 88]. We believe that the swept volume of a link can be studied easily and classified geometrically in a few cases so that for all swept volumes classified in the same set the procedure of enclosing them in a superellipsoid is the same. We will address ourselves to this enclosure problem in a forthcoming paper.

4.3 Hierarchical Grid Construction.

So, the hierarchical construction of the non-uniform grid for a robot arm with k DOFs consists of k steps. Each previously not examined part of the swept volume of a link of the robot arm is modelled by one (possibly more) superellipsoids. In the i-th step the i-th dimension of Cspace is added to the non-uniform grid constructed in the previous steps. In the first step the link closest to the base support is considered. For this link the free configurations can be computed adequately and these determine the first subdivision along the first coordinate axis in Cspace. In step i we only examine the area that was free up to the $i-1$-th step. We add a dimension, subdivide the new dimension in such a way that it can easily be represented as a GET-Rep (see also the next subsection), and do a collision check for each cell. A cell determines an interval for each of the first i joints, and thus a swept volume of the i-th link with respect to the first i joints.

Observe that if we model the last $k-i$ joints by their swept volume and also do a collision check for it (colouring some more cells as forbidden region), we can try to find a path in the part of the Cspace that is built up only for the first i joints. If a path is found, it will only need movements in the first i joints. If it is not found, we only need to consider the cells where the swept volume of the last $k-i$ links collides (the 'grey' cells) for further collision checking. After the next step in the construction of the grid we can expand the potential field that was already computed.

Briefly, the policy of our method is that for the first few links the GET-Rep is constructed and U is not. From a certain link on (e.g. the third or fourth of a 6 or 7 DOF manipulator) we also consider the swept volume of the last few links and determine the grey cells and compute U for the first few dimensions, using different increments for different directions (joints). If no path is found yet, a new dimension is added and the already computed U can be extended to the new dimension. As long as no grey cells are computed, all white cells for the first few links have to be investigated. As soon as the grey cells are determined, the white cells are free for all links and thus only the grey cells have to be investigated further.

4.4 Building Hierarchically the GET-Rep.

In order to construct the GET-Rep in a hierarchical way, we need two routines. The first one adds an extra dimension to the Cspace, the second one processes the information from the computation of new black and white cells in the cells under investigation (the cells for which no collision occurs for the first i links or the cells known to be grey in the previous hierarchical step).

Suppose we start with the GET-Rep of an i-dimensional Cspace. Since we want to keep the cell construction symmetric (we want the cells to be $i+1$ dimensional cubes), we first have to 'double' the GET-Rep on each level of the corresponding 2^i-tree. Since we have to 'double' all children of a 'doubled' '1' entry in the DFA of the GET-Rep, we best 'double' the GET-Rep level by level. For example, starting from a 1 dimensional Cspace with DFA

$$10,\ 00$$

we double the first level to

$$1010,\ 00\ 00$$

and then the second level to

$$1010,\ 0000\ 0000.$$

'Doubling' a level, consists of two actions. First we transform a 2^i-tuple T into a 2^{i+1}-tuple by concatenating two copies of the original 2^i-tuple. And second, we insert on each level a copy of the 2^i-tuples corresponding to the subtrees of the '1' entries in tuple T on the appropriate place. When this is done for all 2^i-tuples on level j, we repeat the process for all 2^i-tuples on level $j+1$. This is the first of the two routines. We illustrate the routine once more with an example of a GET-Rep of a 2 dimensional grid to a 3 dimensional one. Say we start with

$$1010,\ 0000\ 1000,\ 0000.$$

Then doubling on the first level results in

$$10101010,\ 0000\ 1000\ 0000\ 1000,\ 0000\ 0000.$$

Doubling on the second level results in

$$10101010,\ 00000000\ 10001000\ 00000000\ 10001000,\ 0000\ 0000\ 0000\ 0000.$$

Finally we also double the third level, resulting in

$$10101010,\ 00000000\ 10001000\ 00000000\ 10001000,$$

$$00000000\ 00000000\ 00000000\ 00000000.$$

This process has to be carried out analogously for the DFB of the GET-Rep.

The second operation consists of splitting up some $i+1$ dimensional cells in smaller ones. Here we have to turn some '0' entries in the DFA into '1' entries, corresponding to the extreme values for the $i+1$-th parameter computed during the hierarchical construction of the $i+1$-th dimension of Cspace. In the DFB some cells will be marked white and others black.

5 U Construction.

As in [Barraquand & Latombe 91] we use a wavefront expansion approach for the computation of a potential field (U) on the non-uniform grid. The non-hierarchical construction goes as follows.

The list for U-values, organised as the DFB, initially contains the value 'infinity' for all entries. The cell enclosing the initial configuration gets the value '0' for U in the U-list. Then we compute the neighbours of that cell and assign the value '1' to the free cells. In the next step we look for the neighbours of the cells with U-value '1' and assign the value '2' to the free ones that did not get a value in a previous step. We proceed in this way until none of the cells with highest U-value has a free neighbour. A great disadvantage of the non-hierarchical construction is the high amount of neighbour computations needed, especially for k larger than 5 (as reported in [Van Geem 94]).

A hierarchical construction not only of the GET-Rep but also of the potential field results in a potential field with a slightly different but also very useful meaning. A path along the descent of this potential field prefers moves of the links closest to the base support. For each link it still has the advantage of being a good pay-off between safe and short path, still preferring the larger sized cells.

The main motivation for thinking of a hierarchical way of constructing U is that then it might happen that we can already find a path in a low dimensional Cspace. Since Cspace is constructed starting from the joint closest to the fixed link of the robot arm, executing a path found in a low dimensional Cspace will result in moves of the first links only. These links are usually the ones that are moved preferably. This planning approach leeds in a natural way to gross motion planning, by specifying the goal position of the robot configuration as a large cell. Additional fine motion could then be planned locally, in and about the large cell that was specified as the goal in the gross motion planning faze. And this can be repeated on line for the not yet needed goal configurations until all free Cspace is modelled.

5.1 The Hierarchical Way.

In a grid there are three kinds of cells: white, grey and black cells. If we would give a U-value to grey cells, a path could be planned through a grey cell. This can cause the well known problems of planning with grey cells and heuristic approaches, since we are not sure that a path through a grey cell is going to be feasible: this depends on the links that were not yet considered for the construction of the Cspace. Theoretically it can even happen that a grey cell turns black completely. Therefore we decided not to give a U-value to the grey cells.

We can start computing a potential field as soon as the initial configuration of the robot is in a white cell. We then can give a U-value to each white cell that can be connected with the cell where the initial configuration is by a path through white cells. Some white cells may not be reached: they might lie in another connectivity component of the free space in Cspace or currently unreachable because some grey cell is lying in the way.

When a new dimension is added to the Cspace and grey cells turned (partially) into white, grey or black (only grey cells turn into grey ones), we can determine in which cell (for the new dimension) the initial configuration lies.

5.2 Algorithm for Extending the Potential Field.

The potential field is stored in a list like the DFB list. Here the entries are the integer number of the potential field assigned to the corresponding cell in the grid. During the operations for the extension of the grid to the $i+1$-th dimension, this list changes in the same way as the DFB.

Let k denote the number of degrees of freedom of the robot arm and d the maximum number of cells along an axis in Cspace. There are two kinds of waves that have to be performed for the U-value assignment. Firstly, during the doubling of white cells that were reachable from the initial configuration, they all get the value of the i-dimensional cell as a temporary value. When the $i+1$-dimensional cell containing the initial configuration is found, we perform a wave starting from that cell but only expanding in the new direction. In all other white areas generated from reachable white cells in the i-dimensional grid, we perform a wave in the new direction only, visiting only cells generated from the same i-dimensional cell, starting from the cell with same coordinate in the new dimension as the cell containing the initial configuration.

Secondly, we perform waves for the assignment of a U-value to all newly reachable white cells, generated from a previously unreachable white cell or from a grey cell. During whitening a cell in a grey area, we look for its white reachable neighbours in the old directions. We assign to this new white cell the smallest U-value of these neighbours, added to the appropriate increment. These new white cells are stored as candidate sources, ordered in sets of cells with the same U-value.

From these sources on we perform a wavefront expansion in any direction, through all previously grey and all previously unreachable white areas. We start with a wave from those candidate sources with smallest U-value. Each cell that inherits a U-value through this wave is removed as candidate source. When the wave stops and some candidate sources are left, we repeat the process.

The increment depends on the direction of inheritance. When a cell inherits its U-value from a neighbour in the j-th direction, then the increment is equal to d^{k-j}.

After these computations, all reachable white cells in the $i+1$-dimensional grid will have a U-value, which finally allows us, to use the path planning approach described in [Van Geem 94], following the descent of the potential field.

Note that we can start computing a potential field in the $i+1$-dimensional subspace of Cspace if the projection of the initial configuration of the robot onto the $i+1$ first dimensions is in a $i+1$-dimensional white cell. We then can give a U-value to each white cell that can be connected with the cell where the initial configuration is by a path through white cells. When in the $i+1$-dimensional grid the initial configuration is in a white cell while it was not yet in a white cell in the i-dimensional grid, then the first kind of waves is not performed and in the second kind of waves the cell containing the initial configuration is the only candidate source with potential value equal to '0'.

6 Slight Adaptations.

The operations discussed in this section appear whenever some change in the environment of the robot is observed. We start with a given model of the Cspace, in the form of a GET-Rep and a potential field U on the non-uniform grid. Overall we can say that there are two kinds of updates for U. The first one involves only local changes to the grid and to U. The second one involves the impact of the change on long distance from the place where the event took place. When we only apply the first one, we still get a potential field without local mimima and with the certainty that a path will be found if there exists one. However, we loose the nice property that the path will be a pay-off between safe and short path. The second kind of update of U will repair this property by recomputing over a long distance the values of the potential function. We will call the 'wavefront' computation for the first kind of update the *local wave* and for the second kind of update the *update wave*. We now come to the four operations.

6.1 Delete an Obstacle.

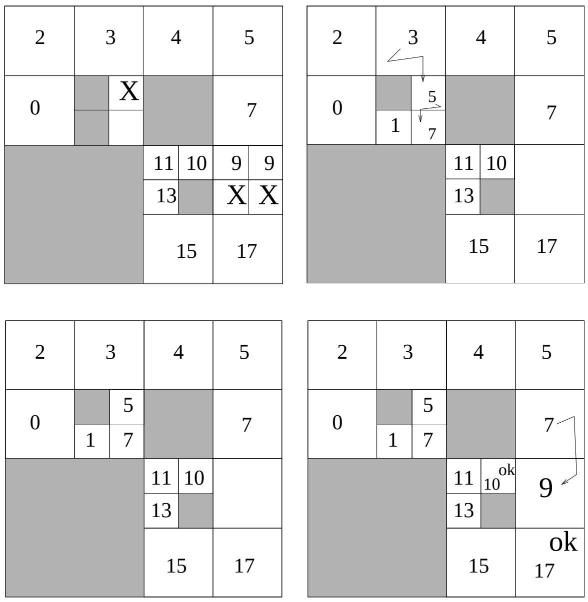


Figure 2: local wave.

When an obstacle has to be deleted, some black cells turn into white ones. These new white cells form a new white area. Sometimes deleting an obstacle also implies that 2^k cells can be merged to one larger cell, and this can happen subsequenty on several levels. Locally the potential field can be 'repaired', in the sense that we can assign a U-value to the new white cells just by looking at its direct neighbours. This is just a local wave.

The local wave works in the following way. Find among all neighbours of the area of new white cells, the one with the lowest value and start up a wave in the new white cells from there. The values assigned to the new white cells have to lie between the lowest and the highest U-value among the neighbouring cells of the new white area. The increment has to be small enough in order to make sure that the U-values for the new white cells do not go over the higher bound. This increment can be determined by considering the amount of new white cells and their size. If the whole new white area corresponds to exactly one new cell (after merging together several smaller cells), the U-value from the local wave will simply be the minimum value of all values on the merged smaller sized cells.

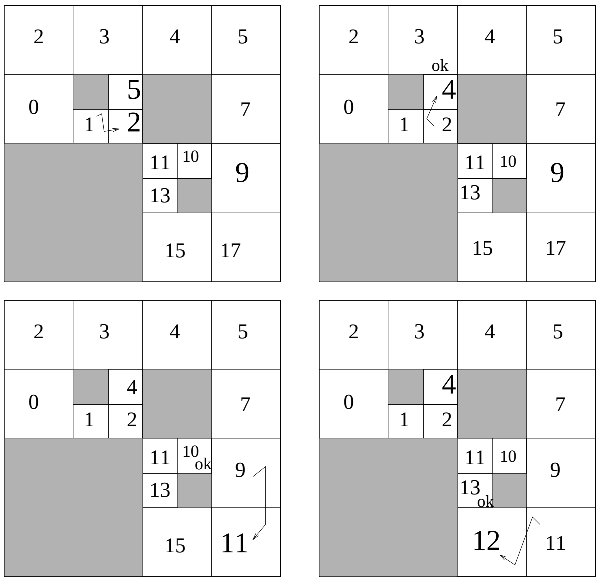


Figure 3: update wave.

To perform the update wave, we start from the cell(s) in the new white area with a neighbour with lowest U-value of all neighbours of the new white region and proceed in the usual way. The update wave does not update those cells (outside the new white area) that have already a U-value equal or smaller than the value the wave would assign to them. The wave does not go on from such cells.

It is irrelevant of how many connected components the new white area consists since the update wave disperses first from all cells with the current U-value and only after that step from the cells with lowest U-value larger than the current one. In Figures 2 and 3 we show on an example such local and update waves. In Figure 2 the first adaptation of the potential field is done for three cells that came to be free (the ones with 'X' in it). First four smaller cells are merged together, then the new white cells get a suitable U-value. In Figure 3 the potential field is restored completely as if the cells that were freed had never been blocked.

6.2 Insert an Obstacle.

When a new obstacle has been observed, its insertion into Cspace will result in new black cells. These black cells might replace white cells completely or partially. In the latter case, the white cell has to be split up and part of the subcells will become black.

In some cases the new black cells will block a passage in Cspace, in other cases no such dramatic change will occur. These two possibilities have to be treated seperately. When a passage is blocked, we cannot expect that a local wave is able to adapt U, since U has to be changed drastically. This can only be done by an update wave. When no passage is blocked, a very easy and very local change will be sufficient for fixing up the potential field in order to find 'a' path.

6.2.1 Fixing: No Blocked Passage.

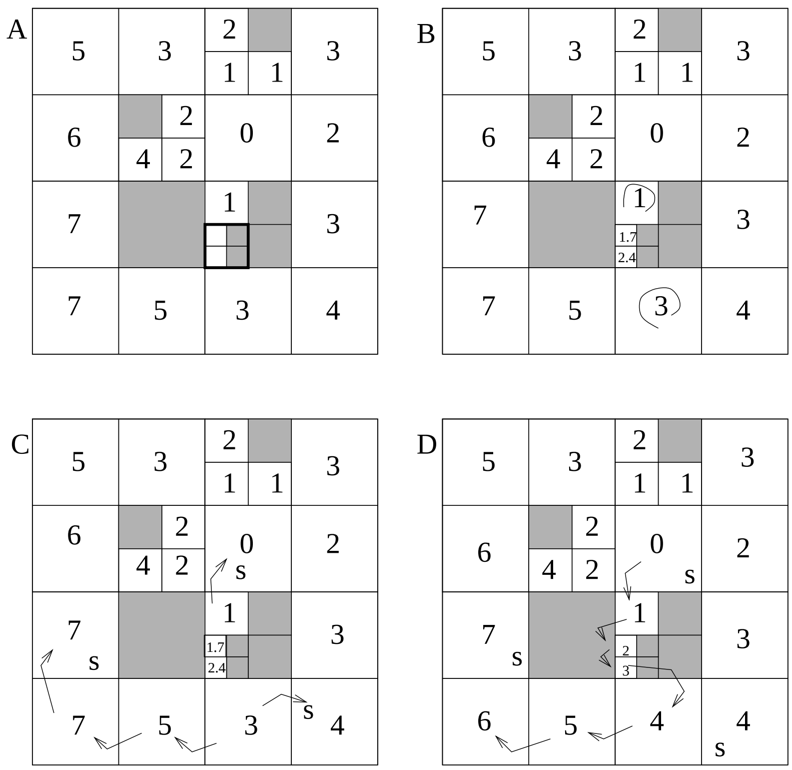


Figure 4: A: new black cells, B: fixing, C: wave in opposite direction, D: update wave from sources (s)

Deciding on whether or not we are in this case can be done simply by looking at the neighbours of the new black area. If they all can inherit their U-value from one of their neighbouring cells that were uneffected by the insertion of the obstacle, then we are in this case. The white cells resulting from a subdivision of a white cell that turned black partially, are considered in this test as if they had the U-value of the large cell from which they originate. This search is the first step of the treatment for the local wave anyway, as described in the next subsection. Intuitively, in this case there is still a passage to all the neighbours of the new black region.

The only cells to worry about are the previously white cells that are subdivided and partially turned black. We treat each of these cells as if they were the only cell that was subdivided by the insertion of the obstacle. We look for the largest and the smallest U-value of all previously white (and partially or completely white still) neighbouring cells of the previously white cell under treatment. All subcells of the previously white cell under treatment will get a U-value between these largest and smallest U-values. A wave is started up from the cell(s) where this smallest U-value was (were) found. Only the new white subcells get a U-value. The increment used in this local wave depends on the size of the subcells (how high the new subtree in the corresponding 2^k-tree is), and the strategy used for the construction of the original potential field U. This can be computed with the information available. This is sketched in Figure 4, A and B.

6.2.2 Update Wave.

Some neighbours of the new black area inherited their U-value from one of the white cells that turned (partially) black and cannot inherit the same value from any of their other neighbours. Some new cells appeared and have to get an appropriate U-value. These cells have to be treated with the update wave we describe in this subsection.

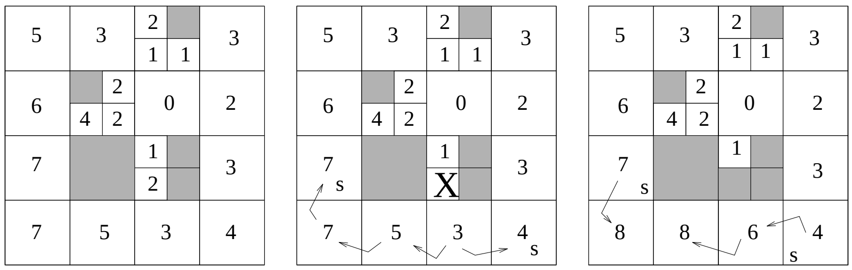


Figure 5: Passage Blocked: Wave in Opposite Direction, Update Wave. (s: candidate sources)

First we search for all the neighbours of the new black area that have to be treated with the update wave. Then we start looking for cells (potential sources for the update wave) from where these cells can inherit a new value. Sometimes such a potential source does not exist. In that case we have an area in Cspace that cannot be reached from the initial position of the robot (we have another connected component, created by the inserted obstacle). In order to decide whether such a source cell exists and, if there exists one, to find sufficiently many of them, we start up a *wave in the opposite direction* (away from the new black area). We start from the neighbouring cells of the new black area with lowest U-values. We call this process also a 'wave' because we have to consider all the neighbours of these cells first before we move to the cells with larger U-values neighbouring to the new black area or previously treated cells. We only treat cells that cannot get their U-value from any other cell than from one of those treated before or from a white cell now (partially) turned into a black one. For this reason, the wave has to stop, either at white cells which are potential sources or at black cells. When it stopped, we may have found potential sources for the update wave in those cells where the 'wave in the opposite direction' died out. We now let the update wave start at the potential sources, adapting the U-values. A potential source is activated only when the inheritance of the U-values reach the value of U at that potential source.

Figure 5 illustrates this for the case in which a passage is blocked. An example in case no passage is blocked can be found in Figure 4.

6.3 Model More Accurately.

Starting from an obstacle and a construction algorithm for Cspace, we get a more accurate subdivision of some black cells. That is, some black cells are subdivided into smaller cells and some of those smaller cells become white. We can assume that no complete black cell turns into a white cell, since we assume that the information we had in the previous grid was correct (i.e. in the black cells some configurations are forbidden, but maybe not all of them). The adjustment of U can now be done exactly like in the approach for deleting an obstacle, described above.

6.4 Move an Obstacle Slightly.

Assume that a non-uniform grid on the Cspace of a robot arm in an environment with several obstacles is given and, that it is observed that an obstacle is not exactly on the position for which the grid was computed. This slight move of this obstacle results in minor changes in Cspace: some larger white cells might be subdivided and turn partially black, some black cells might turn partially white and sometimes get merged with some neighbouring white cells to result in a larger white cell. The adjustment of U can be done in a similar way as for the previously discussed operations, by consecutively adding and deleting black cells.

7 Path Planning.

The path is found by following the descent of the potential field and in the same way as in [Van Geem 94], that is, independent of the way in which the potential field on the grid is constructed. Starting with the cell enclosing the goal position we find the neighbours of a cell along the path and proceed with the neighbour with lowest U-value until the initial position is reached. Due to the different increments moves in the first links will be preferred. As pointed out already in [Van Geem 94] the computation of a path is really fast when the potential field is given, because the algorithm computes only neighbours of cells along the path and the number of cells along the path is close to the minimum number of cells in a path. For the fixed initial position a path can be found to any goal position.

The path is a chain of cells. We claim that a trajectory planner can be designed that uses optimally the information hidden in such a representation of a path: e.g. the size of a cell depends on the distance to the nearest obstacle, in large cells joints can be moved fast.

8 Conclusions.

In order to solve the path planning problem for a manipulator given a fixed initial position and for any configuration as goal position, we proposed to construct a non-uniform grid on the Configuration Space represented by the GET-Representation and the use of a discrete potential field on the grid. New results decribed in this paper are the construction of the GET-Rep using the fast collision checking algorithm, the policy for using the hierarchical wavefront extension with k different increments within the general settings of the robot motion planning problem, and the techniques for adapting the hierarchically built GET-Rep and 'potential field' on the grid when small changes have to be made.

References

[Barr 81] **Barr,A. H.** , Superquadrics and Angle-Preserving Transformations, IEEE Computer Graphics and Applications, Vol. III, January 1981, pp. 11-23.

[Barraquand & Latombe 91] **Barraquand, J. , and J.-C. Latombe J. -C.** , Robot Motion Planning: A Distributed Representation Approach, The International Journal of Robotics Research, Vol.10, No.6, December 1991, pp. 628-649.

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

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

[Gargantini 82] **Gargantini, I.** , Linear Octrees for Fast Processing of Three-Dimensional Objects, Computer Graphics and Image Processing 20, pp. 365-374 (1982).

[Gorla & Renaud 88] **Gorla, B. ,and Renaud, M.** , Models of Robot Manipulators, in Geometry and Robotics, pp. 306-335, Lecture Notes in Computer Science No. 391, eds. Boissonnat, J. -D., and Laumond, J. -P. , Proceedings of Workshop, Toulouse, May 1988.

[Roider & Stifter 87a] **Roider B. , and Stifter, S.** , Collision of Convex Objects, In Proceedings EUROCAL'87, Leipzig, GDR, Springer LNCS 378, pp. 258-259, 1987.

[Roider & Stifter 87b] **Roider B. , and Stifter, S.** , Collision of Convex Objects in the Plane, In Proceedings of the Tenth Tagung Berichte aus Informatikinstituten, Vienna, Austria, Oldenburg Publishing Company, 1987.

[Stifter 88a] **Stifter, S.** , A Medley of Solutions to the Robot Collision Problem in 2 and 3 Dimensions, PhD thesis, RISC-Linz series 88-12.0, 1988. Also: VWGÖ Publishing Company, Dissertations at the University of Linz, Vol. 81, 1989.

[Stifter 88b] **Stifter, S.** , An Algorithm to Enclose Objects in a Robot Working Cell by Superellipsoids, Technical Report, RISC-Linz, 88-72.0.

[Stifter 88c] **Stifter, S.** , A Generalisation of the Roider Method to Solve the Robot Collision Problem in 3D, In Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC '89), Rome, Italy, July 4-8, Springer LNCS 358, pp. 332-343, 1988.

[Stifter 91] **Stifter, S.** , Collision Detection in the Robot Simulation System SMART, International Journal of Advanced Manufacturing Technology, Vol. 7, 1992, pp. 277-284.

[Stifter 92] **Stifter, S.** , The Roider Method: A Method for Static and Dynamic Collision Detection, Issues in Robotics and Non-Linear Geometry, JAI press, 1992.

[Van Geem 94] **Van Geem, C.** , On Using a Manhattan Distance-like Function for Robot Motion Planning on a Non-Uniform Grid in Configuration Space, Technical Report, RISC-Linz, 94-74, To appear in the Austrian Issue of the International Journal Flexible Automation and Integrated Manufacturing (FAIM).

[Van Geem 95a] **Van Geem, C.** , Computation of a Non-Uniform Grid on the Configuration Space of a Robot Arm Amidst Obstacles: Intersection Problem, Proceedings of the 11th European Workshop on Computational Geometry (CG'95), 23-24 March 1995, Linz (RISC), Austria.

[Van Geem 95b] **Van Geem, C.** , Hierarchical Construction of a Potential Field on a Non-Uniform Grid on the Configuration Space of a Robot Arm, In Proceedings of the 4th International Workshop on Robotics in Alpe-Adria Region (RAA'95), 6-8 July 1995, Poertschach, Austria

[Verwer 90] **Verwer, B. ,J. ,H.** , A Multiresolution Work Space, Multiresolution Configuration Space Approach To Solve The Path Planning Problem, in Proceedings of the IEEE Conference on Robotics and Automation, pp. 2107-2112, Cincinnati, Ohio, May 1990,

[Zarrugh 85] **Zarrugh M. Y.** , Display and Inertia Parameters of Superellipsoids as Generalized Constructive Solid Geometry Primitives, In Proceedings of the 1985 ASME International Computers in Engineering Conference and Exhibition, August 4-8, 1985, Boston, MA, Vol. 1, pp. 317-328.