Fast Path Planning for Robot Manipulators Using Numerical Potential Fields in the Configuration Space

E.Ralli and G.Hirzinger
German Aerospace Research Establishment (DLR)
Institute for Robotics and System Dynamics
Postfach 1116, D–82230 Wessling, Germany
e-mail: eleni@donau.df.op.dlr.de

Abstract: *We present a path planning algorithm for robot manipulators working in an initially known environment. An efficient method constructs the configuration space (C-space) of the robot and expands a local minima free numerical potential field in it. So the robot is able to move very fast and collision–free within its workspace (W-space). An extension of that principle allows the robot to handle autonomously new obstacles appearing in its W-space in a fast manner. We have implemented the planner on our 6 degrees of freedom (DOF) – considering 5 of them[1] – space robot ROTEX working within a tight workcell.[2] In order to reduce the off-line computation time of the C-space, the C-space may be only partially constructed taking approximately 5 minutes on a Silicon Graphics IRIS Indigo R4000. Path finding solutions for many practical situations are then produced in less than 1 minute.*

I. Introduction

The basic path planning problem consists of finding a collision-free path for a rigid robot in a static environment being cluttered with rigid objects, connecting its start and goal configurations and perhaps optimizing certain criteria (i.e. shortest path, minimum time, minimum energy etc.).

A variety of planners has been implemented and it is recognized that the solution of the path planning problem in its generality is almost intractable [2].

The computational complexity of the above problem grows drastically with the number n of degrees of freedom (DOF). A variety of 3 DOF systems has already successfully been implemented [3],[4],[5],[6],[7],[8].

The path planning problem for robots with more than 3 DOF ($n \geq 4$) presents an actual research topic of great interest and is in particular very important for real robot applications. It can be treated by the so-called exact methods, which guarantee – in case a solution does exist – to find it, at the cost of extensive computation time [9],[10].

The most promising algorithms are based on powerful heuristics [11],[12],[13]. They do not guarantee to find an existing solution, but consume much less computation time. Satisfactory results are reported for planners considering line robots even with many DOF [14],[15] or dealing with 3D robots with up to 4 DOF [16]. Further planners for real robots are slow [17],[18],[19],[20].

Extensions of the basic path planning problem, like the ability to operate collision-free in real-time in an environment containing new static or moving obstacles, increase drastically its solution difficulty and computational complexity.

Considering the above observations, it leads to the necessity for developing a fast planner for real robots with many DOF, capable of producing solutions operating in a static environment and of dealing with a changing environment providing small reaction times compared to the robot's motion speed. We have developed our planner under these aspects; so initially we had to choose a method, which seems to satisfy our requirements.

The most popular methods like cell decomposition [21] and roadmap methods [22] construct some kind of connecting channel between the start and goal configuration. As illustrated in [22], such methods are practicable for $n \leq 4$. New obstacles appearance or redefinition of start or goal configuration means complete reconstruction of the moving channel within the configuration space (C-space). This is a very time consuming procedure.

Khatib pioneered the potential field methods [23] implementing a collision avoidance algorithm for mobile robots or line robots with up to 4 DOF operating within a dynamic environment. The disadvantage of this method is the appearance of local minima and their restrictive role to the planning procedure, so that it often fails to reach the goal configuration. The local minima can be removed by harmonic potential functions; successful results have been reported for mobile robots and robot manipulators with up to 3 DOF [8],[24].

[1] Our robot consists of 6 rotational links. The rotation of the 6th axis doesn't change the position of the origin of the gripper coordinate system, but only its orientation.

[2] integrated in the spacelab of shuttle Columbia during the German D2–mission in spring '93 [1]

Latombe et al. introduced numerical potential fields [15] applying to 3D C-spaces and the "distributed representation approach" [14] handling many DOF, up to 31 as reported. The examples refer to line robots only and not always to a 3D workspace (W-space).

We use a numerical potential field in the discretized C-space of our robot manipulator and initially consider a static environment.

The remainder of our paper is organized as follows:

Section II discusses the set-up of the also discretized W-space, section III the construction of the C-space, and section IV the modelling and expansion of the potential field in the C-space and the path finding procedure herein.

In section V experimental results are reported. Section VI deals with the problem of a changing environment and section VII with the ability of our planner to find a path very fast in the partially constructed C-space (another than in III). In section VIII a conclusion is presented and how we intend to extend our planner.

II. The W-space

We assume that the W-space of a robot contains r obstacles B_i, (i = 1,...,r) with arbitrary shapes. Then the free space is defined by:

$$W_{free} = W - \cup_{i=1}^{r} B_i$$

We model the W-space by a binary-valued 3D array, let us call it the *W-array*. This *W-array*, as a discretized representation of the W-space, results from mapping the continuous cartesian W-space in world coordinates into an equidistant 3D-grid. In the *W-array* free regions are indicated by ones and obstacle-regions by zeros.

This distinction of our W-space in free and non-free space is a preparational step for the construction of the C-space (as illustrated in the following section) and for the on-line collision avoidance (sections VI,VII).

III. The C-space

A configuration q of a robot A is represented by a vector $(q_1, ..., q_n)^T$, that uniquely defines the position of any point of the robot in its W-space. We denote with n the dimension of the C-space, which equals to the number of DOF. The C-space is the set of all possible configurations considering the mechanical constraints [25].

We model the C-space as a joint angle space containing all possible configurations under consideration of the joint angle limits.

Now the problem is to construct the C-space obstacles CB_i (i = 1,...,r), that means to map the obstacles B_i from the W-space into the C-space. With $A(q)$ we denote the subset of the W-space, filled by A at configuration q.

Each obstacle B_i maps into the C-obstacle CB_i, defined by the set of configurations:

$$CB_i = \{q \in C \mid A(q) \cap B_i \neq 0\}$$

The set of the collision-free configurations from the C-space is then defined by:

$$C_{free} = C - \cup_{i=1}^{r} CB_i$$

This mapping from a low dimensional space (W-space) into a higher dimensional space (C-space) is an unsolvable problem. For low dimensional C-spaces this mapping can be realized with great effort [22].

Therefore we avoid the direct mapping of each obstacle B_i to the set of the corresponding collision-configurations CB_i, following another way to construct the C-space obstacles, based on the consideration:

> a configuration q of a robot A is collision-free if no intersections exist between the robot and its non-free W-space.

Similar to the W-space, we represent the discretized C-space by a n-dimensional array, let us call it the *C-array*. According to our experiments and for simplicity we further assume n = 5. We neglect the 6th joint angle in the *C-array* modelling, because the rotation of the 6th axis does not change the position of the gripper coordinate system (figure 1).

Now we have to examine if every element of the *C-array* represents a collision-free configuration or not. This would be a very time consuming procedure, if we had to examine each element of this *C-array*. We accelerate the *C-array* construction by avoiding redundant checks, i.e. if a link collision is detected, then collision for every further link, lying between the actual one and the robot gripper, is registered. Based on this observation, we developed the following algorithm, which simplifies and accelerates the collision checks accordingly:

1. Initialize each element of the *C-array* with a great number M, representing a collision-free configuration.
2. Consider the 1st link of the robot and neglect each further link. In our case (see figure 1), link 1 rotates around its own axis, therefore a collision check for every discretized value of q_1 is not necessary.
3. Consider the 2nd link of the robot and neglect each further link. Repeat the following steps for every discretized value k_1 (normalized from 1 to climit$_1$[3]) of q_1:

[3] climit$_1$,...,climit$_5$ result by the joint angle limits and the discretization step be choosen for their discrete representation. Thus climit$_i$ is the number of the particular discrete joint angle values. So the C-array contains climit$_1$ x climit$_2$ x climit$_3$ x climit$_4$ x climit$_5$ elements.

for every discretized value k_2 (normalized from 1 to climit$_2$) of q_2:

- Get the exact position of link 2 in its W-space through the forward kinematics.
- To accelerate the collision checking procedure consider first a number of points, let us call them "critical points", from 2nd link (see figure 1). If they lie in the W_{free}, that means if the respective elements of the *W-array* have the value 1, then go on testing further points in the 2nd link. Interrupt the search, if a non collision-free point of the 2nd link has been detected.
- Assuming link 2 collides for the k–th discretized value of q_1 and the l–th value of q_2, then enter the value 0 at the following elements of the *C-array*:

$$C\text{-}array_{i_1 i_2 i_3 i_4 i_5} = 0$$

where

$$\left.\begin{array}{l} i_1 = k \\ i_2 = l \\ i_3 = 1, ..., climit_3 \\ i_4 = 1, ..., climit_4 \\ i_5 = 1, ..., climit_5 \end{array}\right\} \text{redundant collisions}$$

4. Move on in a similar way to the following links, that means consider only the actual link and neglect the following ones. In addition, make use of the C-array elements set to zero to avoid unnecessary collision checking.

For example, if we consider the 3rd link and its position in the W-space according to the k-th discretized value of q_1, the l-th of q_2 and the m-th of q_3, we check (before doing a collision test) if

$$C\text{-}array_{klmi_4 i_5} = 0$$

with

i_4 an arbitrary value in {0,...,climit$_4$},
i_5 an arbitrary value in {0,...,climit$_5$}.

In this manner, interpreting the construction of the C-array elements as a tree expansion, only the "living leafs" (elements with the value M, also non-colliding configurations) are entered step by step into the graph and a selective expansion takes place.

Notes:

a. During the collision test for link 5 we do not neglect the 6th link and end-effector, but we consider them as an extension of the 5th link.
b. We begin the collision check for link 5 by examining if the end-effector or link 6 hit on the robot-base, because this is not guaranteed through joint-angle limits.

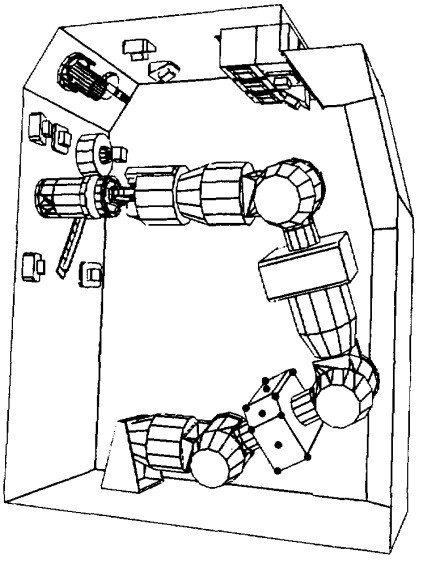

Figure 1: The ROTEX robot within its workcell. The critical points from link 2 due to the collision avoidance are marked.

IV. Potential field expansion in the C-space and path searching in it

According to section III, the C-array has been built in such a way, that its elements contain the values 0 (collision-configurations) or M (non-collision-configurations).

Then a numerical potential field is expanded, based on a "wave expansion" algorithm. In the following we describe this procedure:

1. Given a goal configuration $q(q_1,...,q_5)$ in the C-space, we check out if it collides in the W-space. If not, then we discretize q to the non-colliding discretized configuration $k(k_1,...,k_5)$, that means the element $C\text{-}array_{k_1 k_2 k_3 k_4 k_5}$ contains the value M. Then we assign the value 1 to this element.
2. The so-called 1-neighbors of k (only one coordinate differs by one discretization step from the coordinates of k) are considered, that means we consider the 10 neighbors having coordinates included in the set:

$$\{(k_1+1, k_2, k_3, k_4, k_5), (k_1-1, k_2, k_3, k_4, k_5), ..., (k_1, k_2, k_3, k_4, k_5-1)\}$$

These neighbors are checked if they are "legal", in the sense that their coordinates belong to the predefined

joint limits and they correspond to collision-free configurations. We consider the subset of the "legal" neighbors, to which no potential value has been assigned yet:

$$\text{for every } k_i : \quad 0 \le k_i < \text{climit}_i$$
$$\text{and} \quad C\text{-}array_{k_1 k_2 k_3 k_4 k_5} = M$$

Elements satisfying these conditions are assigned the value 2 and temporarily stored in a list L_1.

3. Repeat step 2 with all elements of L_1, constructing sequentially the lists $L_2,...,L_m$, which contain elements with the value 3,...,m+1 respectively, until no neighbor with the value M can be found.

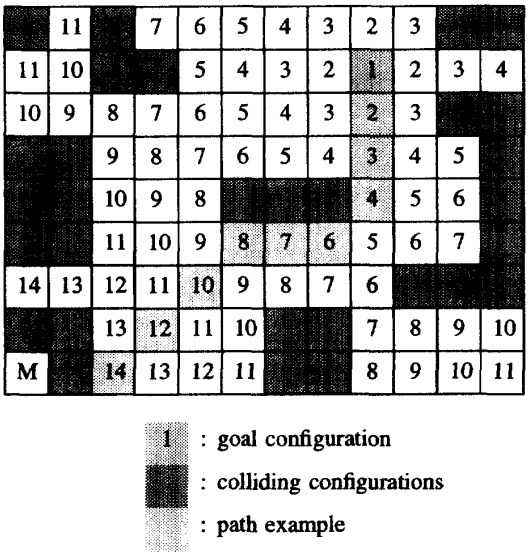

Figure 2: 2D C-space and potential field expansion therein.

A path between the given start and goal configuration exists, if the element of the C-array corresponding to the start configuration contains a value different from 0 or M.

An example for a 2D C-space, the expansion of the numerical potential field therein and a path example are illustrated in figure 2. In this example the path between the given configurations is found by considering the 8 2-neighbors (up to two coordinates differ by a single discretization step) of the particular start configuration. Then the "legal" neighbor with the smallest value is chosen.

We want to point out two significant advantages of our algorithm:

a. The existence of a path connecting two arbitrary configurations can be examined by considering local information only (the potential field value of the start configuration).

b. If a solution path exists, it is guaranteed that it will be found (no local minima exist, which would prevent the planner from reaching the goal configuration).

V. Experimental results

We have implemented our algorithm on a Silicon Graphics IRIS Indigo R4000 Workstation.

The robot operates in a narrow workcell (1016mm × 823mm × 403mm) containing several objects, which form additional obstacles, as can be seen in figure 1. Hence, collision avoidance plays a very important role.

The construction of the W-space array takes approx. 3.5 sec for a resolution of 1 cm. The largest amount of the off-line computation time is consumed for the construction of the C-array. For 2.86×10^6 configurations it takes approx. 55 minutes. In this case the expansion of the potential field in the C-array needs only 7.1 sec.

The joint angle resolution is different for each angle, so that a step angle change for each DOF provides approximately the same movement of the robot wrist.

Each motion step requires the evaluation of 242 neighbors, which is the number of 5–neighbors for a 5D space ($3^n - 1$, n: the dimension of the C-space). Dependent on the C-space resolution, an on-line collision check of robot positions lying between two sequential discretized configurations has to be made. We apply a linear interpolation in the cartesian space. Taken this into account, the computation time for each step takes approx. 0.4 sec – 0.6 sec. A typical path, such as shown in figure 3 is produced in approx. 2.8 sec.

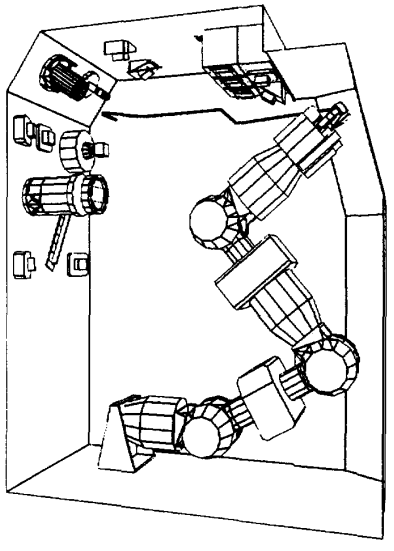

Figure 3: Typical path within the workcell.

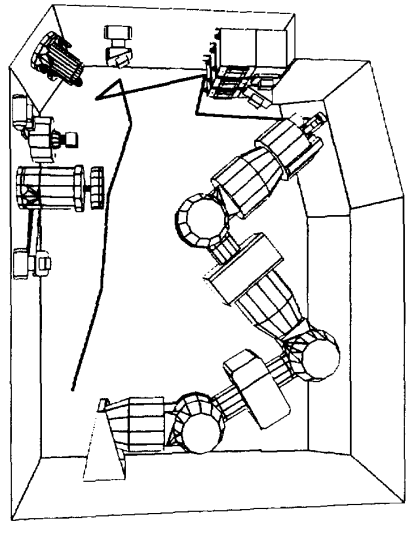

Figure 4: Complex path within the workcell.

One main advantage of our algorithm is its ability to construct very quickly another path connecting a new start configuration to the goal configuration. For a more complicated path than shown in figure 3, it needs approx. 4.6 sec (figure 4). Comparable planners [17],[18],[19] require higher computation times lying in the order of few minutes. A direct and exact comparison is in these cases not possible because both of the different working environment of the particular robot (i.e. a cluttered environment or one with only a few obstacles) and of the different computers, on which the algorithms have been implemented. In [20] the same problem is handled, but the planner requires significant higher computation time.

In the case of a new goal configuration, the potential field has to be reexpanded from this new goal configuration. As mentioned above, this procedure takes only approx. 7.1 sec.

It can be concluded that our planner is able to operate very fast in a known environment, completing real-time requirements not only for slow motions, but for large velocities, too. After these encouraging results, we decided to test out our planner in an initially known environment, augmented with new obstacles appearing during operation.

VI. Operating in an environment with new obstacles

The apparent problem could be solved with the transformation of the new W-space obstacles into the C-space, but as illustrated in section III, this mapping is a very difficult, if not even impossible procedure. In addition, it makes sense only if the new obstacles are static ones, this means they will not change their position and orientation within the W-space till the experiment is completed.

Thus our basic idea is to provide an additional collision checking during the path finding procedure. This on-line collision avoidance has been implemented in a different way as stated in section III:

- Because of the neighborhood of sequential configurations, it is known which of the coordinates of the next configuration differ from the current configuration coordinates (one to maximum five).
- Taking this into account we check only the links which have to move during the transition to the following configuration. We begin the collision check at the last link.

The appearance of new colliding configurations plays an important role in the path finding procedure. For each searching step the 242 5-neighbors are considered. The configurations with the smallest potential value (i.e. v) are taken into account and if they collide, then a neighbor-configuration having a potential value increased by 1 (v+1) has to be chosen. If all the 5-neighbors with the potential value v+1 collide, then the subset from the 5-neighbors with potential value v+2 are considered and so on. This procedure may cause local minima in the potential field. In relation to this strategy we make the following observations:

1. If all 5-neighbors have greater potential values than the actual one, then the current configuration is a local minimum of the potential field. Following the above stated strategy local minima can be avoided. To prevent getting trapped into the same local minimum in the next searching step, we forbid the choice of configurations belonging to the current path.
2. The selection of a non-colliding neighbor from the particular set of equipotential neighbors means to handle an ambiguity. This ambiguity can be overcome by choosing the first "legal" neighbor found. In this case a zigzag path probably arise. In order to eliminate these zigzag moves we have also implemented other methods, which are based on certain sorting criteria applied to a subset of equipotential 5-neighbors. We favorize the following one: the line connecting the start and goal configurations (in the cartesian space) is computed as well as the distance from the particular neighbor of the above subset to this line. Then the neighbor showing the shortest

distance is selected, so we call this criterion the "shortest distance criterion".

3. In the worst case each neighbor-configuration collides, so it is impossible to plan the next motion step and to reach the goal configuration.

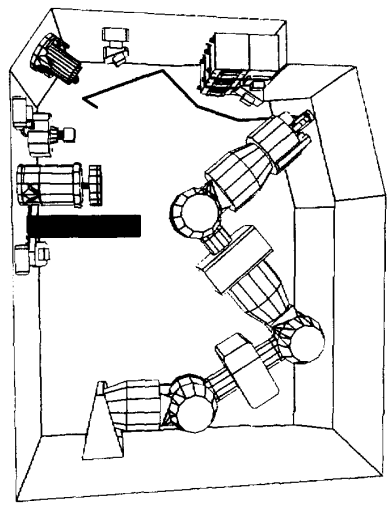

Figure 5: Path within a changed environment.

Figure 5 shows the computed path for the same start and goal configurations as in figure 3, but within a changed environment. The computation time for each step contains 0.4 – 2 sec and for the complete path approx. 5.2 sec, under application of the "shortest distance criterion" explained above. If the first "legal" neighbor is chosen, then the computation time is about 10% lower.

Assuming that the new obstacles will remain in the workcell for the rest of the experiment, we enter the value 0 in the respective element of the C-array during operation. So our planner **learns** the changes that have been made in the W-space.

As stated above, the elements set to zero might cause local minima in the potential field. Usually there is enough time between the execution of two tasks in a workcell. So a reinitialization of the potential field is possible, which will cause the local minima to disappear.

As mentioned in section V, the computationally most intensive task is the construction of the C-array. The need for fast path finding for many practical situations led us to the following ideas.

VII. Path planning in a partially constructed C-space

In this case we don't apply the methods presented in section III, but the following ones: the collision avoidance algorithm does not take in account all the links of the robot, but only the origin of the gripper coordinate system.

With this algorithm we only partially construct the C-space, call it C'-space, such that $C_{free} \subset C'_{free}$. We represent the discretized C'-space by the C'-array.

Therefore an on-line collision avoidance while path searching is required (similar to section VI). We apply the searching technique illustrated in VI and enter the value 0 to the respective element of the C'-array, each time a collision-configuration has been found.

The off-line construction of the C'-array requires about 5 minutes. The path planning problem illustrated in figure 6 needs only 3.8 sec. For more complicated paths a re-expansion of the potential field is recommended for further smoothing of the path. The example illustrated in figure 7 is solved within 88 sec, time for a reexpansion included.

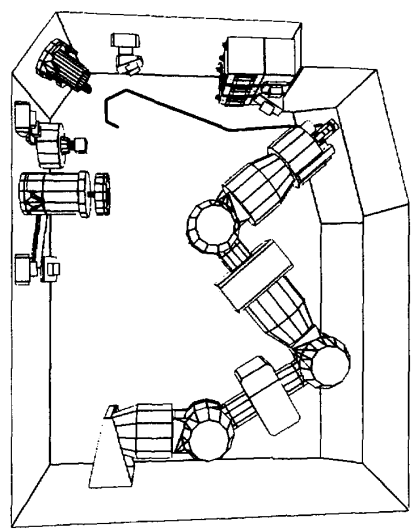

Figure 6: Typical path in the partially constructed C-space.

We have experimented with other C'-spaces too, i.e. with C'-spaces resulting from the collision avoidance of one link only, or from the collision avoidance based on some critical points of the robot links, like those illustrated in figure 1.

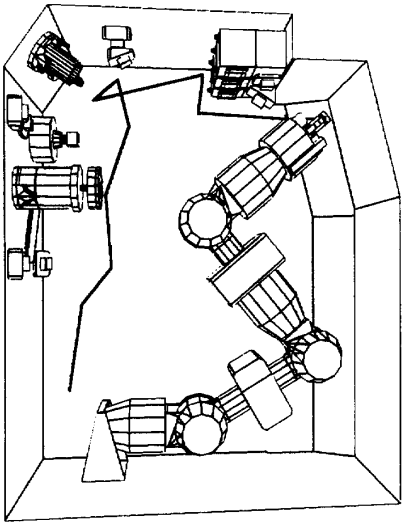

Figure 7: Complex path in the partially constructed C-space.

VIII. Conclusion and further work

The algorithm presented is able to quickly produce a collision-free path in a partially constructed C-space (section VII), and in another working mode to compute off-line all the collision-configurations of a discretized C-space. This initialization takes place once and is the main reason for the ability of our method to produce a path in real-time (section V), even in a changed environment (section VI).

Our planner can be characterized as a gross motion planner for transfer motions, producing "through-points" in the C-space and guarantees a collision-free path. A further extension of the planner would be necessary to provide the capability of generating exact trajectories, which are required for assembly tasks. Until now, we solve such problems by introducing a mode-switch into a sensor-based operating mode [1],[26].

The problem of dealing with a dynamically changing environment is of great interest today. Freund et al. present a very interesting concept handling multi-robot systems and moving or stationary obstacles [27], but introducing restrictions in the number of DOF of the robots and in the dimension of the W-space. Reports using high parallelized algorithms [28] show no satisfactory reaction times. Our algorithm is parallelizable with respect to the construction of the C-space and the evaluation of the neighbor values during the path searching procedure. The essence of our future work will deal with handling moving obstacles, but preserving good reaction times.

The path planning method presented provides a very helpful tool, which has been applied up to 5 DOF for real 3D robots. It can be adopted to other robot configurations without significant effort. The knowledge of the joint angle limits and the forward kinematics map is necessary. The exact dimensions of every joint and link are required for the collision avoidance.

The limits of the extension to more than 5 DOF are set by the main memory size of the computer and the real-time capabilities of the evaluation of the surrounding neighbors from the particular start configuration (section V). For 6 DOF the extension is practicable even on conventional hardware platforms, but for more than 6 DOF very powerful computers are required (the size of the *C-array* grows exponentially with the number of DOF).

In summary, we were able to exceed the limit of 4 DOF for fast path planning of real robots. The results present a powerful (sections V,VII) and extendable method (section VI).

References

[1] G. Hirzinger, "ROTEX – The First Space Robot Technology Experiment", *Preprints of the Third International Symposium on Experimental Robotics, Kyoto, Japan*, pp. 302–320, Oct.28–30, 1993.

[2] J. Reif, "Complexity of the mover's problem and generalizations", *Proceedings of the 20th Symposium on the Foundations of Computer Science*, pp. 421–427, 1979.

[3] E. Mazer, G. Talbi, J. Ahuactzin, and P. Bessière, "The Ariadne's Clew Algorithm", *SAB 92*, Honolulu 1992.

[4] S. Akishita, T. Hisanobu, and S. Kawamura, "Fast Path Planning for Moving Obstacle Avoidance by Use of Laplace Potential", *Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Yokohama, Japan*, pp. 673–678, July 26–30, 1993.

[5] D. Kortenkamp, M. Huber, C. Cohen, U. Raschke, C. Bidlack, C. Congdon, F. Koss, and T. Weymouth, "Integrated Mobile-Robot Design", *IEEE Expert*, pp. 61–73, August 1993.

[6] R. Jarvis, "Collision-Free Path Planning in Time Varying Obstacle Fields Without Perfect Prediction", *Proceedings of the International Conference on Advanced Robotics (ICAR), Tokyo, Japan*, pp. 701–706, 1993.

[7] C. Seshadri and A. Ghosh, "Optimum Path Planning for Robot Manipulators Amid Static and Dynamic Obstacles", *IEEE Transactions on Systems, Man and Cybernetics (SMC)*, vol. 23, no. 2, pp. 576–584, March/April 1993.

[8] J.-O. Kim and P. Khosla, "Real–Time Obstacle Avoidance Using Harmonic Potential Functions", *Proceedings of the IEEE Int. Conference on Robotics and Automation, Sacramento, California*, pp. 790–796, 1991.

[9] B. Paden, A. Mees, and M. Fisher, "Path planning using a Jacobian-based freespace generation algorithm", *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1732–1737, 1989.

[10] F. Avnaim, J. Boissonnat, and B. Faverjon, "A practical exact motion planning algorithm for polygonal objects admist polygonal obstacles", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 166–1661, 1988.

[11] T. Lozano-Pérez, "Automatic Planning of Manipulator Transfer Movements", *IEEE Transactions on Systems, Man and Cybernetics (SMC)*, vol. 11, no. 10, pp. 681–698, 1981.

[12] R. Brooks and T. Lozano-Pérez, "A Subdivision Algorithm in Configuration Space for Findpath with Rotation", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no. 2, pp. 224–233, 1985.

[13] D. Zhu and J. Latombe, "New Heuristic Algorithms for Hierarchical Path Planning", *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 9–20, February 1991.

[14] J. Barraquand and J.-C. Latombe, "Robot Motion Planning : A Distributed Representation Approach", *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, December 1991.

[15] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical Potential Field Techniques for Robot Path Planning", *IEEE Transactions on Systems, Man and Cybernetics (SMC)*, vol. 22, no. 2, pp. 222–241, March/April 1992.

[16] P. Pal and K. Jayarajan, "Fast Path Planning for Robot Manipulators Using Spatial Relations in the Configuration Space", *IEEE International Conference on Robotics and Automation, Atlanta*, vol. 2, pp. 668–673, May 1993.

[17] T. Lozano-Pérez, "A Simple Motion-Planning Algorithm for General Robot Manipulators", *IEEE Journal on Robotics and Automation*, vol. RA – 3, no. 3, pp. 224–238, June 1987.

[18] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom", *Proceedings of the IEEE International Conference on Robotics and Automation, Raleigh,NC*, pp. 1152–1159, 1987.

[19] K. Kondo, "Motion Planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration", *IEEE Transactions on Robotics and Automation*, vol. RA-7, pp. 267–277, June 1991.

[20] G. Duelen and C. Willnow, "Path Planning of Transfer Motions for Industrial Robots by Heuristically Controlled Decomposition of the Configuration Space", *Proceedings of the Euriscon Conference, Korfu, Greece*, 1991.

[21] L. Gouzènes, "Strategies for Solving Collision-Free Trajectories Problems for Mobile and Manipulator Robots", *International Journal of Robotics Research*, vol. 3, no. 4, pp. 51–65, 1984.

[22] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[23] O. Khatib, "Real–Time Obstacle Avoidance for Manipulators and Mobile Robots", *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[24] Y. Hwang, "A Potential Field Approach to Path Planning", *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 23–32, 1992.

[25] T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach", *IEEE Transactions on Computers*, vol. C-32, no. 2, February 1983.

[26] B. Brunner, G. Hirzinger, K. Landzettel, and J. Heindl, "Multisensory Shared Autonomy and Tele-Sensor-Programming — Key issues in the Space Robot Technology Experiment ROTEX", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan*, pp. 2123–2139, July 26–30, 1993.

[27] E. Freund and H. Hoyer, "Real-Time Pathfinding in Multirobot Systems Including Obstacle Avoidance", *The International Journal of Robotics Research*, vol. 7, no. 1, pp. 42–70, Feb. 1988.

[28] E. Mazer, J. Ahuactzin, E. Talbi, and T. Chatroux, "Parallel Motion Planning with the Ariadne's Clew Algorithm", *Preprints of the Third International Symposium on Experimental Robotics, Kyoto, Japan*, pp. 223–228, Oct.28–30, 1993.