Copyright © 2002 IFAC
15th Triennial World Congress, Barcelona, Spain

REACTIVE PATH PLANNING FOR ROBOTIC ARMS WITH MANY DEGREES OF FREEDOM IN DYNAMIC ENVIRONMENTS

Margarita Mediavilla, José L. González, Juan C. Fraile, José. R. Perán

Departamento de Ingeniería de Sistemas y Automática, E.T.S.I.I.
Universidad de Valladolid. Spain.
e-mail:marga@eis.uva.es

Abstract: It is well known that path planning for robots with many degrees of freedom is a complex task. That is the reason why the research on this area has been mostly restricted to static environments. This paper presents a new method for on-line path planning for robotic arms in dynamic environments. Most on-line path planning methods are based on local algorithms that end up being inefficient due to their lack of global information (local minima problems). The method presented in this paper avoids local minima by using a two stage framework. The robots react to dynamic environments using a local and reactive planning method restricted to a subset of its configuration space. Since the subset has few degrees of freedom the computational cost of the on-line stage is very low. An off-line stage chooses the subset of the configuration space that minimizes the probability of blockades and inefficient motions. *Copyrigh ©t 2002 IFAC*

Keywords: path planning, robotic manipulators, configuration space.

1. INTRODUCTION

Robot path planning has been an active area or research over the last 40 years. It is well known that the complexity of path planning tends to become enormous when the number of degrees of freedom of the robot is large (5 to 7 d.o.f). Few approaches deal with robots with many d.o.f. in dynamic environments. By dynamic environment we understand a workplace with still and moving obstacles where the position of the obstacles at each moment can be known, but where the entire trajectory cannot, either because of external unknown perturbations, or because it is influenced by our own path planning.

Since most path planning is designed for static environments, most robot manipulators are restricted to pre-calculated trajectories and rigid timings, and their behavior is non reactive. Increasing the reactivity of robot manipulator would have important advantages. The robots could be less dependent on pre-calculated trajectories and do their path planning on-line, they could reecho to unexpected events such as moving obstacles or faulty situations, and they could move without a previous model of their environment, based only on sensor information (such as mobile robots do).

Path planning methods based on probabilistic roadmap, random searches, the sequential framework, hierarchical searches, or harmonic functions (Gupta and Pobil, 1992) deal successfully with complex problems in reasonable computing times, but most of these methods cannot be applied to dynamic environments.

Path planning for dynamic environments is often based on local methods. The earliest approach is the potential field method by Khatib (1986) whose main drawback is the generation of local minima. Local minima-free potential fields seem to be impossible to build based only on local information (Gupta and Pobil, 1992). One interesting way of solving this problem is presented by in the elastic band framework by Quinlan and Khatib (1993), which combines local and global information. The drawback of the elastic bands method is that a collision-free path between initial and final configurations must always exist. In (Hamilton and Dodds, 1998) there is a reactive approach based on a set of behaviors, that ends up having a problem similar to the local minima of potential fields. Max and Xianyi, (1998) present a method based on neural networks. The method opens up an interesting field, but the authors do not train the net, therefore they might find local minima problems. Mataric (2001) has worked on path planning for humanoid robots with many degrees of freedom. Her approach reduces the effective number of degrees of freedom of the robots, using a human to teach the robot. This paper presents a method for path planning for robots located in dynamic environments, also seen in (Mediavilla et al., 1998; Mediavilla et al., 2001). The method is tested on a system of three five-link robots.

1.2 Description of the method.

The path planning method we present in this paper is based on what we call ***motion strategies***. These are simple and effective ways of moving a robot. Motion strategies are implemented by restricting the path search to a subset of the configuration space (*C*), of the robot, what we call reduced subspace (R-subspace) or C_R.
One single motion strategy cannot solve all the task assigned to a robot, therefore our analysis is based on grouping the tasks that the robots perform into what we call ***motion problems***. Motion problems are general types of tasks that can be solved using the same R-subspace. We need to ensure that this reduction is correct, therefore our method is completed with an off-line pre-planning stage that chooses the R-subspace that optimizes the motion.

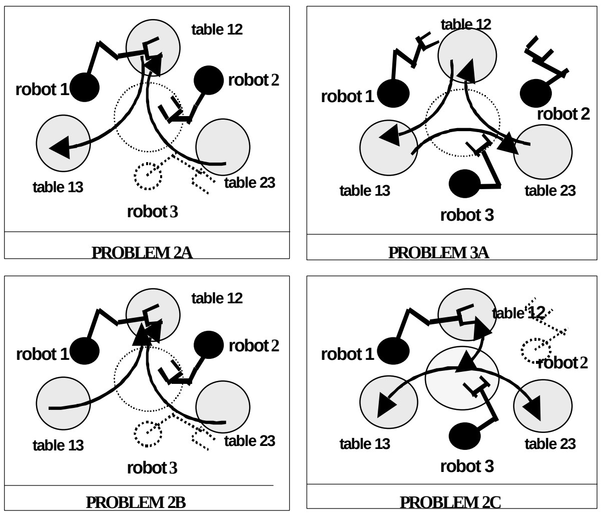


Figure 1. Motion problems.

Thus, our method has the following stages:

- ***off-line stage.*** This stage decides which strategies are most suitable for the motion of the robot.
- ***On-line stage.*** When the robots move they search their path inside the R-subspace chosen in the off-line stage. Since the R-subspace has a low dimension the path planning algorithm is very fast and reactive.

In this first approach to path planning based on strategies we have used R-subspaces that only have two degrees of freedom. Despite their simplicity these subspaces can solve some interesting problems in a very effective way. There might be motion problems that cannot be solved using a R-subspace, or whose probability of success using a R-subspace is very low. In those cases out off-line analysis would still be useful to be ensure that more complex planners are needed.

One of the key issues related to path planning is weather a method is complete or not. Since we are reducing the number of degrees of freedom of the search space, it is very likely that our algorithm is not complete. But, on the other hand, we are dealing with moving and unknown obstacles, therefore, we can always find obstacles that make our way impossible, no matter what path planning we use! When we deal with path planning for uncertain environments we might need to forget about mathematically complete algorithms and search for just a "good planner". The researchers that make soccer competitions of mobile robots do not search for a path planning method that guarantees that our team is going to win 3-0 to the opposite team, they just deal with the uncertainties of the real world as well as they can.

This paper describes our approach to path planning based on motion strategies and its application to a multi robot system of three five-link manipulators. Section 2 describes the basic features of the method. In section 3 the off-line analysis stage is described, while section 4 describes the on-line stage. Section 5 shows some results of the application of this method. Finally the conclusions are drawn in section 6.

2. MOTION PROBLEMS, STRATEGIES AND R-SUBSPACES.

2.1 Definition of strategies and R-subspaces.

The R-subspaces we have chosen are linear and two dimensional subsets of *C* (configuration space). Since all the motion of the robot must be done inside them, the initial $\mathbf{q}_I$ and final $\mathbf{q}_F$ configurations of the robot must be contained in C_R. R-subspaces will be characterized by a base of two vectors $\mathbf{u}_1$ and $\mathbf{u}_2$:

$$\mathbf{C_R} = \{\mathbf{q} \in \mathbf{C} \ / \ \mathbf{q} = \mathbf{q_I} + \mathbf{x}\mathbf{u}_1 + \mathbf{y}\mathbf{u}_2\} \qquad (1)$$

where x and y are independent variables and $\mathbf{u}_1$ is a vector common to all possible R-subspaces, that points out from the initial to the final configuration:

$$\mathbf{u}_1 = (\mathbf{q_F} - \mathbf{q_I}) / \| \mathbf{q_F} - \mathbf{q_I} \| \qquad (2)$$

The vector $\mathbf{u}_2$ is the one that determines the *strategy* of the robot. For example, in a PUMA type robot with six revolute joints, if $\mathbf{u}_2 = (0\ 1\ 1\ 1\ 0\ 0)$ the motion along $\mathbf{u}_2$ implies that the second, third and fourth joints of the robots rise while the rest remain unchanged. This is what we call "go-up" motion. Motion inside C_R is, therefore, a combination of movements towards the goal (parallel to $\mathbf{u}_1$), and "go-up" movements (parallel to $\mathbf{u}_2$).
Since we are working with robots with *n* degrees of freedom and our R-subspaces are two-dimensional, we have *n-2* degrees of freedom to choose the vector $\mathbf{u}_2$ and therefore, the strategy of the robot. To obtain an orthonormal basis we would need $\mathbf{u}_2$ such that:

$$\mathbf{u}_2 \cdot \mathbf{u}_1 = 0 \qquad (3)$$
$$\| \mathbf{u}_2 \| = 1 \qquad (4)$$

Condition (3) is fulfilled if $\mathbf{u}_2$ is obtained as a linear combination of an orthonormal basis of the null space of vector $\mathbf{u}_1$. Therefore, we may find a basis of the null space with *n-1* vectors $\mathbf{v}_i$ *i*=1,...n-1 such that:

$$\mathbf{v}_i \cdot \mathbf{u}_1 = 0 \qquad (5)$$

In our five link robots for example, we have only used the first four degrees of freedom of the robot, since the last one is the rotation of the gripper that does not need to be taken into account for collision avoidance (the robots do not carry large objects). Therefore our $\mathbf{u}_2$ is a linear combination of the three vectors of the base of the null space of $\mathbf{u}_1$:

$$\mathbf{u}_2 = a \cdot \mathbf{v}_1 + b \cdot \mathbf{v}_2 + c \cdot \mathbf{v}_3 \qquad (6)$$

Equation (4) poses one extra restriction, since we are only interested if the relative size of the elements of vector $\mathbf{u}_2$, not in its modulus. We may, therefore, give an arbitrary value to one, lets say *a*=1, and choose arbitrarily the rest of the parameters *b*, and *c*. These two parameters define all the possible directions of $\mathbf{u}_2$, although the sign of the first element might be positive or negative. Therefore we may define $\mathbf{u}_2$ using three parameters:

$$\mathbf{u}_2 = \frac{l(\mathbf{v}_1 + a \cdot \mathbf{v}_2 + b \cdot \mathbf{v}_3)}{\sqrt{1 + a^2 + ß^2}}$$

(7)

Where $\lambda = \pm 1$ is what we call the local direction of the strategy and α and β have arbitrary values. We call **strategy** the set of parameters $L = (\alpha, \beta, \lambda)$. The physical meaning of these parameters depends on the basis $\mathbf{v}_i$ of the null space that we choose. When a robotic system is composed of several robots, one has to think about a global strategy for the system. This strategy would be describe by the parameters of the *m* moving robots:

$$L = (\alpha_1, \beta_1, \lambda_1, \alpha_2, \beta_2, \lambda_2 .....\alpha_m, \beta_m, \lambda_m) \qquad (8)$$

This definition of vector $\mathbf{u}_2$ is not the only one we may choose, since any pair of vectors – as long as they are not parallel— define a linear 2D subspace in *C*. A strategy is similar to, but not exactly the same, as an R-subspace. One strategy might lead to two different R-subspaces if the initial and final configurations of the robots are different.

2.2 Characterization of motion problems.

The definition of motion problems depends on the purpose of our system and must be done by the user. To illustrate the idea of the motion problems we shall use the multi robot system where we have applied our method (Mediavilla et al., 1998; Mediavilla et al., 2001). It is composed of three Scorbot-er IX robots with five degrees of freedom each, and several working platforms (see figure 1). Each robot can access two side working platforms and the central table. This system has a communications and control structure that enables joint operation. The control system is located in three computers that are connected to each other by Ethernet and to each one of the robots controllers using the serial port. The control software gets the values of the encoders of the robots and sends to each robot the desired link positions at regular intervals. The tasks assigned to the system are mainly pick-and-place operations between the working platforms. Only gross motion is considered, and it is assumed that no joint manipulation of pieces is needed. In our multi robot system we have defined the following motion problems (see figure1):

? Problem 2A. Two robots move between the tables that are located at their sides. They cross traveling in the same direction.

? Problem 2B. Two robots move between the tables that are located at their sides. They cross traveling in opposite directions.

? Problem 2C. One of the robots moves between side tables and the other moves between one side table and the central table

? Problem 3A. Three robots move between side tables traveling all of them in the same direction.

We must take into account that these motion problems are ***general types of conflicts***. They do not represent the motion the robots from one specific configuration in one table to another configuration in another table. These motion problems represent *any kind on task that the robots might do between working areas.* The initial and final configurations of the robots might vary inside the initial and final areas, and the relative delay of the robots too. This way the robots might find many different types of conflicts in the same motion problem.

3. OFF-LINE STAGE

The first part of our off-line stage is the choice and definition of the *motion problems*, which was discussed in section 2. The second part is the evaluation of strategies in terms of how appropriate they are for a motion problem. This process is basically an optimisation that uses one evaluation index. We have used two indexes: one is what we call Estimation of the Probability of Faults, (EPF), and the other is the average time spent in several tasks. Both are described in section 3.1, while the optimisation process is described in section 3.2. The optimisation process is described in figure 2.

3.1 Estimation of the Probability of Faults (EPF) and average time of a strategy

The purpose of the EPF is finding an appropriate index to evaluate strategies. We define *fault* in this context as a deadlock or a livelock of the robotic system.

We will call P(L) the estimation of the probability of having a fault when using strategy L on a certain motion problem. This estimation is based on a simulation of the robotic system that imitates its real operation. The simulation of the system performs *N* different tasks that belong to the motion problem, and counts the number of faulty tasks, *r*. This experiment is binomial and the real probability of faults p(L) can be estimated by the sampling probability P (L)=*y*=*r*/*N*. We know that a task is faulty when the time needed to perform it is larger that a certain amount. Let us call t^1_k, …, t^j_k,..t^m_k the time spent by each one of the robots *j*=1, 2, ...m, in each one of the tasks *k*=1....N. If t^j_k = t_{max} we set $t^j_k = t_{max}$ and report a ***fault of task k.***

The other index that we use is the average time spent by the three robots solving a motion problem with one strategy. The process that leads to the estimation of these indexes for the strategy L is:

1. find N_k tasks for each one of the *k* robots involved in the motion problem,
2. run the simulation of the system until all the task of all the robots are completed
3. count the time spent in each one of the tasks: t_i^k, i=1,...N_k and the number of faulty tasks r_k
4. calculate the average global time of the simulation: $T(L)=(\Sigma_k \Sigma_{i=1}^{Nk} t_i^k)/N$, where $N=\Sigma_k N_k$. T(L) and P(L)=*y*=*r*/*N* (the EPF) are the optimization indexes.

If the experiment we carry out is binomial we can set confidence intervals to the estimation of P(L) and T(L). This implies that the *N* observations must be independent from each other. The experiments we do are a continuous simulation of the operation of the system, therefore the tasks might be correlated to each other. We have used two ways to break this dependency: the first one is choosing the initial and final points of the motion of each robot randomly inside the working area. Another source of randomness is introduced by making the robots wait a random time before starting a new task. A correlation analysis of the times t^j_k gives us information about how correlated these experiments are.

3.2 Optimization of strategies

The EPF analysis gives us a criterion to decide whether one strategy is better than others, but the ultimate goal of our analysis is the choice of one strategy (or a reduced set of strategies) that solve each motion problem. Since the number of parameters that define one strategy involving *m* robots with *n* degrees of freedom are $(n-2)\cdot m$, this is a complex problem that grows exponentially with the number of degrees of freedom of the robot. One of the approaches to this optimisation problem we have tried involves the use of non-linear programming techniques. The function to minimise would be the average time spent by the m robots in N tasks:

$$T(L)=\Sigma_{j=1}^{m}\Sigma_{k=1}^{N} t^j_k \quad (9)$$

This optimisation should avoid those strategies that have faults, since they have a large value of the task time $t^j_k=t_{max}$. It also avoids those strategies that are slow, and therefore not very efficient. Since the function that we want to minimise is, probably, a highly non-linear and not analytical function the optimisation method that we have decided to use is the flexible polyhedron search --simplex method by Nelder and Mead (Lemeshow et al., 1990)-- which does not use the derivatives of the function. The optimisation of this function is problematic, because we cannot guarantee a global optimum, since local minima will probably exist.

The other approach we have used is based on choosing an arbitrary set of values for the parameters α, β, λ, ... of L and testing all the possible combinations of these values. This approach is simply a set of trials, that are going to be a good starting point and give us interesting information about our strategies. The results described in section 5.

4 ON-LINE STAGE

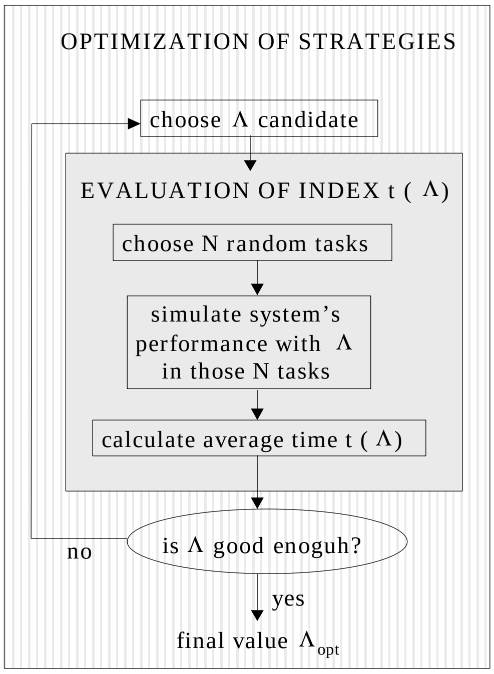
The on-line stage of our method is based on moving the robots inside C_R using a reactive and local algorithm. We have used a simple algorithm based on three behaviors:

1. ***Go to the goal***. This is the default behavior of the robots. It is used when the robot does not detect obstacles closer than a certain distance, *h*. If the configuration of the robot at moment *k* is $\mathbf{q}_k$, the next one will be: $\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta\cdot\mathbf{u}_k^F$, where $\mathbf{u}_k^F = \mathbf{q}_f\text{-}\mathbf{q}_k / \| \mathbf{q}_f - \mathbf{q}_k \|$ points in the direction of the goal, and Δ is a scalar fixed step.

2. ***Avoid obstacles***. This behavior is used when the firs one cannot be followed. The robot checks the neighbor configurations of $\mathbf{q}_k$ according to vectors $\mathbf{u}_k^F$ and $\mathbf{u}_k^T$ and following a local direction, (for example right). The neighbor configurations would be: $\mathbf{q}_k + \Delta\cdot\mathbf{u}_k^F$, $\mathbf{q}_k + \Delta\cdot\mathbf{u}_k^F - \Delta\cdot\mathbf{u}_k^T$, $\mathbf{q}_k - \Delta\cdot\mathbf{u}_k^T$, $\mathbf{q}_k - \Delta\cdot\mathbf{u}_k^F - \Delta\cdot\mathbf{u}_k^T$ and $\mathbf{q}_k - \Delta\cdot\mathbf{u}_k^F$. The robot moves to the neighbor configuration that is closest to its objective but whose distance to the obstacle is greater than *h*. Vector $\mathbf{u}_k^T$ is perpendicular to $\mathbf{u}_k^F$ and belongs to C_R. This vector defines one of the local directions[1].

3. ***Change local direction***. If the previous behavior cannot be done, the robot changes its local direction by doing $\mathbf{u}_k^T = -\mathbf{u}_k^T$.

[1] $\mathbf{u}^T$ is perpendicular to $\mathbf{u}^F$ and belongs to C_R. It can be easily calculated if $\mathbf{u}^F$ is defined in terms of the base $\mathbf{u}_1$ and $\mathbf{u}_2$. For example, if $\mathbf{u}^F = a\ \mathbf{u}_1 + b\ \mathbf{u}_2$, the vector $\mathbf{u}^T = -b\ \mathbf{u}_1 + a\ \mathbf{u}_2$ is perpendicular to $\mathbf{u}^F$ and indicates one of the possible local directions.


OPTIMIZATION OF STRATEGIES
choose Λ candidate
EVALUATION OF INDEX t (Λ)
choose N random tasks
simulate system's performance with Λ in those N tasks
calculate average time t (Λ)
is Λ good enoguh?
no
yes
final value Λopt
Figure2: Off-line optimization of strategies

5. EXPERIMENTS AND RESULTS

The experiments that we have performed in our system are based on motion problem 2 A and motion problem 3 A. In our multi robot system composed of three Scorbot-er IX robots the robots basis are 1.1m apart from each other while the maximum reach of the gripper is 0.85m. The five links of the robots are all revolute, since it is a PUMA type robot. The tasks of the robots are based on problem 2 A and problem 3 A. The robots move back and forth between their side tables, and the initial an final points of their motions vary randomly inside a sphere in configuration space that corresponds to a movement of the gripper of the robot of ε_{alea}=0.1m. The robots wait a random time before doing the next tasks. Those delay times are obtained using a random exponential distribution of mean μ_{exp}=10. A task is considered faulty if the robot takes more than 200 times the time it would take to do the tasks without obstacles. ε=0.05m is the spatial resolution used in our path planning.

Correlation analysis. In order to know if EPF analysis is correct we have tested problem 2A using one particular strategy. We have obtained similar results in all the strategies we have tested. The strategy chosen leads to the following vectors $\mathbf{u}_2^i$ for robots *i*=1, 2 of the R-subspace:

$\mathbf{u}_2^1$ = (0.35, 0.70, 0.37, 0.47, 0) for robot 1
$\mathbf{u}_2^2$ = (-0.2, 0.85, -0.28, -0.37, 0) for robot 2

Therefore robot 1 follows a "go-up" strategy, while robot 2 "bends-up- hand down". In table 1 we can see the results of the correlation analysis of the simulation f N=600 tasks of problem 2 A. We can see that the correlation is not high, therefore our experiments can be considered independent from each other and our estimation of the probability of faults and the average time is acceptable.

Table 1. Correlation analysis of a simulation of N tasks

PROBLEM 2A (random tasks) N=600,ε=0.05m, ε_{alea}=0.1m, μ_{exp}=10

ROBOT 1 (0 faults)		ROBOT 2 (0 faults)	
Mean t_k^1	50.19 points	Mean t_k^2	47.79 points
Std t_k^1	18.0	Std t_k^2	17.8
Xcorr t_k^1 t_k^1 First 6 terms normalizad	-0.11, 0.09, -0.02, 0.005, 0.0016, 0.045	Xcorr t_k^2 t_k^2 First 6 terms normalizad	-0.11, 0.144, -0.11, 0.028, -0.10, 0.070
Correlation t_k^1 t_k^2	0.047	2/vN	0.08

Optimization with Simplex method. The results of one optimization of T(L) using the simplex method by Nead and Melder as programmed in MATLAB (Himmelblau, 1972) can be seen in table 2. These optimizations lead us to coherent results (the optimization converges), but the optimum strategies were never too far away from the initial value from where the optimization started. This is probably due to the presence of local minima in the index function, although it could also mean that the simplex method is not adequate for this problem

Systematic trials. Since the simplex method did not seem to converge to strategies that were good global optimums we performed a systematic search that explored all the space of reasonable values of L. We chose three values of α (-2 , 0 and 2) and four values of β (-2.5, -1, 1, and 2.5) plus the two values of the local direction λ=±1, and look at all the combinations of those values.

Table 2. Results of the optimization of one strategy using simplex method

strategy	**ROBOT 1: fold**	**ROBOT 2: go up**
$L_{initial}$	0, -2.5	2, –2.5
$L_{optimum}$	-0.04, -2.7	2.34, -2.46
no. steps	200	
Time of optimization	100 minutes	
Tolerance of index in optimization	0.5 steps	
T(L_{ini})	101 steps	
T(L_{opt})	94.5 steps	
N	600	600
faults	0	0

All these combination give us 576 strategies. Since the time spent doing a simulation of 600 tasks with two robots is around 30 seconds, trying all the strategies takes around 5 hours. We have simulated one set of 300 tasks on each one of those strategies, plus two sets of 1800 tasks. 47 of these strategies do not detect any fault in any of the simulations. Therefore, using the notation of section 4.2 we have a fault rate P (L) = r/N =0 / 4200 =0.

We have chosen the following of 6 strategies as the most appropriate for problem 2 A, all of them have no faults in all the tasks. The average time spent by the robot in them is shown in parenthesis:

- ***up h++* vs. *up h - -*** (92.5) robot 1 go up with hand up – robot 2 go up hand down
- ***up h+* vs. *fu h –*** (95.6) robot 1 go up with hand up – robot 2 fold up hand down
- ***up h-* vs. *fu h - -*** (96.5) robot 1 go up with hand down – robot 2 fold up hand down
- ***up h+* vs. *do h ++*** (90.2) robot 1 go up with hand up – robot 2 go down hand up
- ***up h +* vs. *do h –*** (87.6) robot 1 go up with hand up – robot 2 go down hand down
- ***up h-* vs *do h +*** (86.0) robot 1 go up with hand down – robot 2 go down hand up

The best strategies are those where robot 1 goes up while robot 2 goes down or folds up. These strategies seem to be very reliable for motion problem 2 A, therefore we have considered that further optimization was not needed.

Three robots The results obtained with problem 2 A have been extended to problem 3A, that deals with three robots that go back and forth between side tables and start crossing in opposite directions. We have checked the combinations of the strategies that detected no faults in problem 2 A. The results are not as good as the ones obtained with problem 2 A, but the failure rate is still very low. The results can be seen in table 3.

Table 3. Results of the best strategies for problem 3 A.

Problem 3 A. Three robots between side tables.
600 tasks, ε=0.05 m, ε_{alea} =0.1 m, μ_{exp}=10

Strategy	$\mathbf{u}_2$	faults	T(L)
up h+ vs. fu h- vs. up h-	$\mathbf{u}_{2,\ robot1}$=(0.014 0.93 0.23 0.25 0) $\mathbf{u}_{2,robot2}$=(0.028 0.55 -0.65 –0.51 0) $\mathbf{u}_{2,robot3}$=(0.014 0.93 0.23 -0.25 0)	4	168
up h- vs. fu h – vs. up h +	$\mathbf{u}_{2,robot\ 1}$=(0.014 0.93 0.23 -0.26 0) $\mathbf{u}_{2,robot2}$=(0.012 0.24 -0.28 -0.93 0) $\mathbf{u}_{2,robot3}$=(0.014 0.93 0.23 0.26 0)	5	171
up h+ vs. do h + vs. up h +	$\mathbf{u}_{2,robot1}$(-0.01 0.37 0.89 0.26 0) $\mathbf{u}_{2,robot2}$=(0 -0.64 –0.16 0.74 0) $\mathbf{u}_{2,robot3}$=(0 0.37 0.89 0.26 0)	9	174

6. CONCLUSIONS

A new on-line path planning method for articulated robots has been presented in this paper. It deals with a problem that has not been fully solved to date: the motion among moving (and unpredictable) obstacles. The planner is a combination of local and global methods: the path of the robot is calculated in a on-line stage using local information, but an off-line stage shows which strategies of motion are most successful for the type of problems that the robot would have to face. This method has been applied to a multi robot system composed of 5-joints robots. The results show on-line path planning for two 5 d.o.f. robots with a very high success rate.

REFERENCES

Gupta, K., Pobil, A. P. (editors) (1998).Practical motion planning in robotics: current approaches and future directions. John Willey and Sons Ltd.

Hamilton, K., Dodds, G.I. (1998). Reactive Planning of Robot Arms in Single and Cooperative Tasks. Proceedings of the IEEE International Conference on Robotics and Automation, pp 336-340.

Himmelblau, (1972). D. M. Applied Nonlinear Programming. Mc Graw Hill, 1972.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. Int. J. of Robotics Research, 5(1):90-8.

Lemeshow, S, D. Hosmer, J. Kler and S. Lwange, (1990). Adequacy of sample size in health studies. World Health Organization. John Willey and Sons.

Mataric, M. (2001). Principled and efficient methods for control and learning in robot teams and humanoids. Proceedings of the LAAS-CNRS 9th International Symposium on Intelligent Robotic Systems, Toulouse.

Mediavilla, M., Fraile, J. C. (2001). A multi robot System of three manipulators with on-line path planning abilities. Proceedings of the LAAS-CNRS 9th International Symposium on Intelligent Robotic Systems,Toulouse

Mediavilla, M., Fraile, J.C. , Dodds, G. (1998) Optimization of collision free trajectories in multi robots systems. Proceedings of the IEEE International Conference on Robotics and Automation.

Meng, Max , Yang, Xianyi, (1998). A neural network approach to real-time trajectory generation. Proceedings of the IEEE International Conference on Robotics and Automation, 1725-1730.

Quinlan, S. and Khatib, O. (1993)Elastic bands: Connecting path planning and control. In. Proc. of the IEEE Int. Conf. On Robotics and Automation, volume 2, pages 802-7.