DYNAMIC TRAJECTORY PLANNING OF ROBOT ARMS

A. Chennakesava Reddy*, P. Ram Reddy**, B. Kotiveerachari***

*Associate Professor, Department of Mechanical Engineering, JNTU College of Engineering, Kukatpally, Hyderabad -500 072
E-mail: dr_acreddy@yahoo.com
**Registrar, JNT University, Kukatpally, Hyderabad - 500 072
***Professor, Department of Mechanical Engineering, National Institute of Technology, Warangal- 506 004

Abstract: *It is well known that trajectory planning for robots with many degrees of freedom is a complex task. This paper presents a new method for on-line trajectory planning for robotic arms in dynamic environments. The method presented in this paper avoids local minima by using a two-stage framework. The arms 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.*

1. Introduction

The complexity of trajectory planning tends to become immense when the number of degrees of freedom of the robot is large. Few approaches deal with robots having many degrees of freedom (dof) in the dynamic environments. For static environments, the trajectory planning is restricted to pre-calculated paths, rigid timings and 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 reach 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.

Trajectory planning methods based on probabilistic roadmap, random searches, the sequential framework, hierarchical searches, or harmonic functions deal successfully with complex problems in reasonable computing times, but most of these methods cannot be applied to dynamic environments [1]. Trajectory planning for dynamic environments is often based on local methods such as:

- Potential field method: the main drawback is the generation of local minima [2]
- Elastic bands method: the disadvantage is that a collision-free path between initial and final configurations must always exist [3]
- Neural networks method: the limitation is the absence of training the net [4]

This paper presents a method for trajectory planning of robotic arms located in dynamic environments. This new approach is named as ***motion strategy method.***

2. The Motion Strategies

Motion strategies are implemented by restricting the path search to a subset of the configuration space (C), of the robot. The subset of configuration space is called as reduced subspace (CR). One single motion strategy cannot solve the entire task assigned to a robot. Therefore, the motion strategy method is based on grouping the tasks that the robots can perform. This method is divided into two stages, namely:

- Off-line stage: This decides the reduced subspace, which is the best suitable for the motion of the robot arm.
- On-line stage: This selects the trajectory path, which is the best suitable for motion of the robot arm.

The motion strategy method is divided into two stages only to reduce the number of degrees of freedom of the search space.

The reduced subspaces (C_R) are chosen to be linear and two-dimensional subsets of configuration space (C). Since all the motion of the arm must be done inside them, the initial q_i and final q_f configurations of the arm must be contained in C_R. Reduced subspaces are characterized by two vectors u_1 and u_2.

$$C_R = \{q \in C / q = q_i + xu_1 + yu_2\} \quad ...(1)$$

where, x and y are independent variables and u_1 is a vector common to all possible reduced subspaces, that points out from the initial to the final configuration:

$$u_1 = \frac{q_f - q_i}{\|q_f - q_i\|} \quad ...(2)$$

The vector u_2 is one that determines the strategy of the robot. As the robot arm consists of n dof and reduced subspace is of two-dimensional, (n-2) dof are assumed for the vector u_2. To obtain an orthonormal basis, u2 is defined as:

$$u_2 \bullet u_1 = 0$$
$$\|u_2\| = 1 \quad ...(3)$$

Condition (3) is fulfilled if u_2 is obtained as a linear combination of an orthonormal basis of the null space of vector u_1. The basis of the null space with n-1 vectors v_i i=1,...n-1 can be found as:

$$v_i \bullet u_1 = 0 \quad ...(4)$$

For the robot arm, the first four dof are only considered, since the last one is the rotation that does not need to be taken into account for collision avoidance. Therefore, u_2 is a linear combination of the three vectors of the base of the null space of u_1:

$$u_2 = a \bullet v_1 + b \bullet v_2 + c \bullet v_3 \quad ...(5)$$

$$= \frac{\lambda(v_1 + \alpha * v_2 + \beta * v_3)}{\sqrt{1+\alpha^2+\beta^2}} \quad ...(6)$$

Where $\lambda = \pm 1$ is the local direction of the strategy and α and β have arbitrary values. The set of parameters $\sigma = (\alpha, \beta, \lambda)$ is called strategy. The physical meaning of these parameters depends on the basis v_i of the null space. When a robotic system is composed of several robots, a global strategy is to be considered for the system. This strategy would describe by the parameters of the m moving robots:

$$\sigma = (\alpha_1, \beta_1, \lambda_1, \beta_2, \lambda_2 ..... \alpha_m, \beta_m, \lambda_m) \quad ...(7)$$

Each robot arm 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 robot arm controllers using the serial port. The control software gets the values of the encoders of the robots and sends to each robot arm 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 this present work, the following motion problems are considered:

- Case-1: Two robots move between the tables that are located at their sides. They cross traveling in the same direction as shown in Fig-1a.
- Case-2: Three robots move between side tables all of them travelling in the same direction as shown in Fig-1b.

These two motion problems represent any kind of task that the robot arms might do between working areas. The initial and final configurations of the arms might vary inside the initial and final areas, and the relative delay of the arms too.

3. Off-line Stage Analysis

The off-line stage is for the choice of, definition of the motion problems and the evaluation of strategies in terms of how appropriate they are for a motion problem. This process is basically an optimization process that uses one evaluation index. In the present study, two indexes are used:

- Estimation of the Probability of Faults (EPF)
- The average time of strategy (ATS)

3.1 Estimation of the Probability of Faults (EPF)

The purpose of the EPF is to find an appropriate index for evaluating the strategies. The fault is defined as a deadlock or a livelock of the robotic system. P (σ) is the estimation 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 process is binomial and the real probability of faults P'(σ) can be estimated by the sampling probability P (σ) = y = r/N. Let, $t^1_k, \ldots, t^j_k, \ldots t^m_k$ the time spent by each one of the robot arms j=1, 2, ...m, in each one of the tasks k =1....N. If $t^j_k = t_{max}$, set $t^j_k = t_{max}$ and record a ***fault of task k.***

3.2 Average Time of a Strategy (ATS)

This index is used to find 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 σ is:

- Find N_k tasks for each one of the k robots involved in the motion problem,
- Run the simulation of the system until all the task of all the robots are completed
- 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
- Calculate the average global time of the simulation
- $T(\sigma) = (\Sigma_k \Sigma_{i=1} N_k t^i_k) / N$

Where, $N = \Sigma_k N_k$

$T(\sigma)$ and $P(\sigma) = y = r/N$ (EPF) are the optimization indexes.

As the search process is binomial, there could be confidence intervals to the estimation of $P(\sigma)$ and $T(\sigma)$. This implies that the N tasks must be independent from each other. Since the search proc-

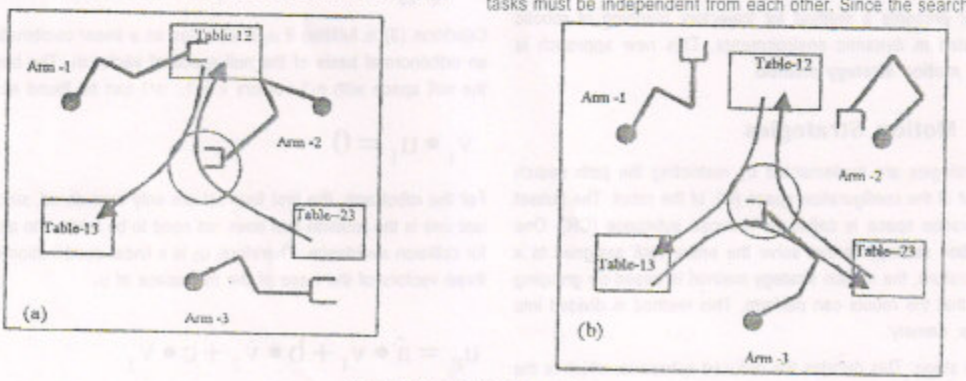

Fig. 1 Illustration of motion problems

ess is of continuous simulation; the tasks may be correlated to each other. The correlation is broken by two ways:

- Choosing the initial and final points of the motion of each robot arm randomly inside the working area
- Making the robot arms wait a random time before starting a new task.

4. Off-line Stage Optimization

The EPF analysis gives a criterion to decide whether one strategy is better than others, but the ultimate goal is the choice of one strategy 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) x m, this is a complex problem that grows exponentially with the number of degrees of freedom of the robot. Non-linear programming techniques are used for optimization. The minimization function is based on the average time spent by the m robots in N tasks:

$T(\sigma) = \Sigma_{j=1} \Sigma^{m}_{k=1} N t^{j}_{k}$ ...(8)

This optimization avoids 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. The optimization technique is simplex method as shown in Fig-2.

5. On-line Stage Analysis

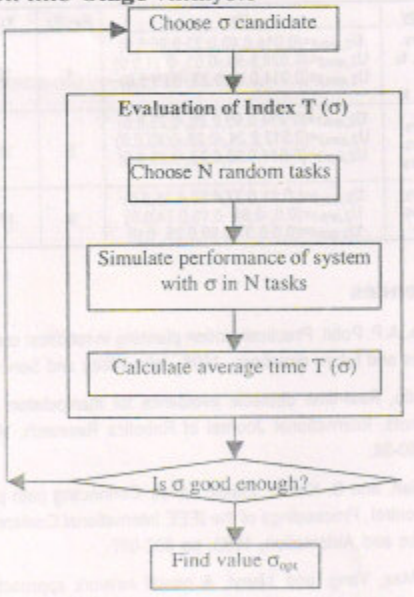

Fig-2. Off-line stage optimization of strategies

The on-line stage is based on moving the robots inside C_R using a reactive and local algorithm. A simple algorithm is developed based on three behaviors:

- **Go to the goal:** This is the default behavior of the robots. It is used when the robot arm does not detect obstacles closer than a certain distance, h. If the configuration of the robot at moment k is q_k, the next one will be:

$q_{k+1} = q_k + \Delta \times u_k^F$,

where, $u_k^F = q_f - q_k / \| q_f - q_k \|$ points in the direction of the goal, and Δ is a scalar fixed step.

- **Avoid obstacles:** This behavior is used when the first one cannot be followed. The robot arm checks the neighbor configurations of q_k according to vectors u_k^F and u_k^T and following a local direction.
- The neighbor configurations would be:

$q_k + \Delta \times u_k^F$,

$q_k + \Delta \times u_k^F - \Delta \times u_k^T$

$q_k - \Delta \times u_k^T$,

$q_k - \Delta \times u_k^F - \Delta \times u_k^T$

$q_k - \Delta \times u_k^F$

The robot arm moves to the neighbor configuration that is closest to its objective but whose distance to the obstacle is greater than h. Vector u_k^T is perpendicular to u_k^F and belongs to C_R. This vector defines one of the local directions.

- **Change local direction:** If the previous behavior cannot be done, the robot changes its local direction by doing $u_k^T = - u_k^T$

5. Results and Discussion

The bases of robot arms are 1.1m apart from each other and the maximum reach of the gripper is 0.85m. The five links of the robots are all revolute. The tasks of the robots are based on case-1 and case-2. The robot arms move back and forth between their side tables, and the initial and 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 = 0.1m. The arms wait a random time before doing the next tasks. The delay times are obtained using a random exponential distribution of mean μ =10. A task is considered faulty if the arm 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 the trajectory planning.

In order to know the correctness of EPF analysis, the case-1 was tested using one particular strategy. The strategy chosen leads to the following vectors u^2_i for robots i =1, 2 of the reduced subspace:

u^1_2 = (0.35, 0.70, 0.37, 0.47, 0) for robot 1

u^2_2 = (-0.2, 0.85, -0.28, -0.37, 0) for robot 2

Therefore arm-1 follows a go-up strategy, while arm-2 bends up and down. The results of the correlation analysis of the simulation for N = 600 tasks of case-1 are given in Table-1. It can be seen that the correlation is not high, therefore the experiments can be considered independent from each other and the estimation of the probability of faults and the average time are acceptable.

Table-1: Correlation analysis of simulation of N tasks for Case-1

	Arm -1	Arm-2
Mean	50.20	47.80
Variance	324.00	316.84
Xcorr first six terms	-0.11, 0.09, 0.02, 0.005, 0.0016, 0.045	-0.11, 0.144, -0.11, 0.028, -0.10, 0.07
Correlation coefficient	0.047	0.080

5.1 Optimization with Simplex method

The results of one optimization of T (σ) using the simplex method written in MATLAB program can be seen in Table-2. These optimizations lead to coherent results, 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

5.2 Systematic Search

Since the simplex method did not seem to converge to strategies that were good global optimums, a systematic search was performed which explores all the space of reasonable values of σ. Three values were chosen for α {-2, 0 and 2) and four values for β {-2.5, -1, 1, and 2.5) plus the two values for the local direction $\lambda = \pm 1$.

Table-2: Results of optimization for Case-1

Strategy	Arm-1 (fold)	Arm-2 (go up)
$\sigma_{initial}$	0.00, -2.50	2.00, -2.5,
σ_{final}	-0.04, -2.7	2.34, -2.46
No. of Steps	200	200
Optimization time	100 min	100 min
Tolerance of index	0.5	0.5
T ($\sigma_{initial}$)	101	101
T (σ_{final})	95	95
N	600	600
Faults	0	0

All these combination give 576 strategies. Since the time spent during a simulation of 600 tasks with two arms is approximately 30 seconds, trying all the strategies takes around 5 hours. One set of 300 tasks was simulated 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, the fault rate P (σ) = r / N =0 / 4200 = 0.

The following of 6 strategies were chosen as the most appropriate for case-1, all of them have no faults in all the tasks. The average time spent by the robot arm in them is shown in parenthesis:

- up h+ vs. up h- (92.5) arm-1 go up with hand p - arm-2 go up hand down
- up h+ vs. fu h- (95.6) arm-1 go up with hand p - arm-2 fold up hand down
- up h- vs. fu h- (96.5) arm-1 go up with hand own - arm-2 fold up hand down
- up h+ vs. do h+ (90.2) arm-1 go up with hand p - arm-2 go down hand up
- up h + vs. do h - (87.6) arm-1 go up with hand p - arm-2 go down hand down
- up h- vs do h + (86.0) arm-1 go up with hand own - arm-2 g down hand up

The best strategies are those where arm-1 goes up while arm-2 goes down or folds up. These strategies seem to be very reliable for motion case-1; therefore the further optimization was not needed.

The results obtained with case-1 have been extended to case-2, that deals with the three arms that go back and forth between side tables and start crossing in opposite directions. The combinations of the strategies that detected no faults in case-1 were checked. The results are not as good as the ones obtained with case-1, but the failure rate is still very low. The results can be seen in Table 3.

6. Conclusions

A dynamic trajectory planning method for robot arms has been presented in this paper. It deals with the motion among moving obstacles. The path of the robot arm is calculated in an 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 arm would have to face. This method has been applied to a multi robot system composed of 5-joints robots. The results show that on-line path planning for two 5-dof robot arms is successful.

Table 3: Results of the best strategies for case 2

Strategy	U_2	Faults	T(σ)
up h+ vs. up h+ vs. fu h up h- vs. fu	$U_{2,arm1}$=(0.014,0.93,0.23,-0.25,0.0) $U_{2,arm2}$=(0.028,0.55,-0.65,-0.51,0.0) $U_{2,arm3}$=(0.014,0.93,0.23,-0.25,0.0)	4	168
up h- vs up h -vs up h +vs	$U_{2,arm1}$=(0.014,0.93,0.23,-0.25,0.0) $U_{2,arm2}$=(0.012,0.24,-0.28,-0.93,0.0) $U_{2,arm3}$=(0.014,0.93,0.23,-0.25,0.0)	5	171
up h +vs up h +vs up h +	$U_{2,arm1}$=(-0.01,0.37,0.89,0.25,0.0) $U_{2,arm1}$=(0.0,-0.64,-0.16,0.74,0.0) $U_{2,arm1}$=(0.0,0.37,0.89,0.25, 0.0)	9	174

Referrences

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

2. O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, International Journal of Robotics Research, Vol.5, 1986, pp 90-98.

3. S. Quinlan, and O. Khatib, Elastic bands: Connecting path planning and control, Proceedings of the IEEE International Conference On Robotics and Automation, 1993, pp 802-807.

4. Meng, Max, Yang, and Xianyi, A neural network approach to real-time trajectory generation, Proceedings of the IEEE International Conference on Robotics and Automation, 1998, pp1725-1730.