Optimal Manipulator Path Planning with Obstacles using Disjunctive Programming

Lars Blackmore and Brian Williams

***Abstract*— In this paper we present a novel, complete algorithm for manipulator path planning with obstacles. Previous approaches have used incomplete methods to make the problem tractable. By posing the problem as a Disjunctive Program we are able to use existing constrained optimization methods to generate optimal trajectories. Furthermore, our method plans entirely in the workspace of the manipulator, eliminating the costly process of mapping the obstacles from the workspace into the configuration space.**

I. Introduction

Robotic manipulators need to be able to operate safely in cluttered, 3-D environments. In order to achieve a manipulation task, a manipulator needs to be able to plan a path through its workspace to a goal location, while avoiding obstacles. Previous approaches have made this highly complex problem tractable by employing incomplete methods, such as those using randomization [1][2] to find a feasible path with high probability. While these have been effective in practice, they are not guaranteed to find a path if one exists, and any path that is found is not necessarily optimal with regard to a given criterion. Furthermore, these methods plan in the configuration space of the manipulator, and mapping the obstacles from the workspace into the configuration space is costly.

We present here a novel, complete algorithm for manipulator path planning with obstacles that uses a *disjunctive programming* [3] approach to make the problem tractable. By using a combination of *branch-and-bound* [4] and *conflict extraction* [5] alongside efficient linear and quadratic programming techniques, disjunctive programming is able to search for the global optimum in a highly efficient manner. Disjunctive programming has been successful for path planning with non-articulated systems such as vehicles [6][7][8]. We extend this work to manipulators. By using this approach we are able to plan entirely in the 3-D workspace of the manipulator, avoiding the need for mapping of obstacles.

In Section II we introduce Disjunctive Programming. In Section III we extend this work to generate optimal trajectories for manipulators moving in cluttered environments.

II. Disjunctive Programming

A disjunctive program is defined as in (1). Here, $\mathbf{x}$ is a vector of decision variables and $f(\mathbf{x})$ is a function to be minimized. The constraints are a conjunction of n clauses, with each clause being a disjunction of m_i inequalities:

Minimize $f(\mathbf{x})$ *subject to:*

$$\bigwedge_{i=1,\ldots,n} \Big(\bigvee_{j=1,\ldots,m_i} C_{ij}(\mathbf{x}) \leq 0 \Big) \tag{1}$$

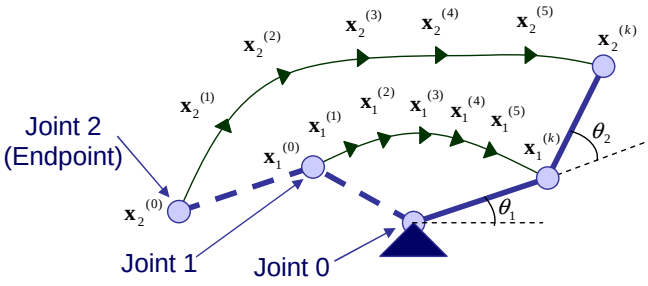

Fig. 1. Path planning for a 2-DOF manipulator. We specify positions for each of the joints, including the endpoint, at discrete time intervals $t = 0, \ldots, k$. Given a desired position for each of the manipulator joints, solving the inverse kinematics problem for the joint positions is straightforward.

If the cost function $f(\mathbf{x})$ and the constraint inequalities $C_{ij}(\mathbf{x}) \leq 0$ are linear or quadratic in the decision variables, then the disjunctive program can be solved efficiently using commercially available software [9] using a combination of branch-and-bound [4] and conflict extraction [5]. Furthermore, quadratic constraints and cost functions can be approximated as piecewise linearities; hence a program involving these can be approximated as a disjunctive linear program [7].

III. Path Planning for Manipulators using Disjunctive Programming

The key idea behind the new method is to plan a feasible, optimal path for *each of the manipulator joints* in the workspace of the robot. This path must be feasible in the sense that:

1) The joints, and the links that join them, must not collide with obstacles
2) The trajectory must be kinematically feasible
3) The trajectory must be dynamically feasible
4) The endpoint must move from the start to the goal

We use $\mathbf{x}_i^{(t)}$ to denote the position of joint i at time step t. The new method generates a finite sequence of positions $\mathbf{x}_i^{(1)}$ to $\mathbf{x}_i^{(k)}$ for each of the manipulator joints, as shown in Fig. 1. We make the assumption that manipulator dynamics have a limited effect. This applies to a great number of manipulation and robot mobility tasks where the motion is relatively slow. In these cases a lower-level controller is used to achieve a given joint position and velocity. This allows the dynamics of the system to be reduced to simple constraints that represent the performance limits of the lower level controller.

In the following sections we show that the feasibility requirements (1) through (4) can be expressed as constraints

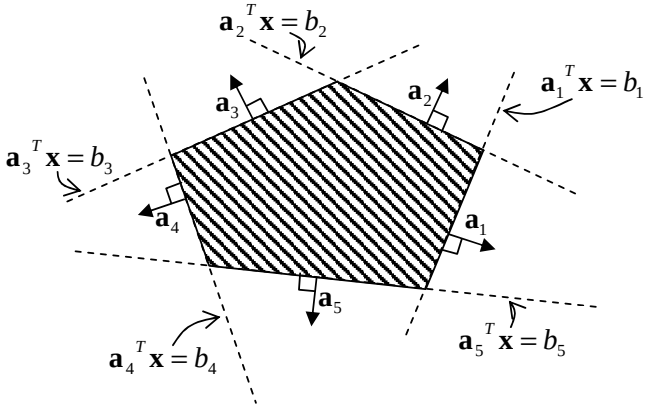

Fig. 2. Two-dimensional obstacle modeled as a convex polyhedron. The vectors $\mathbf{a}_1 ... \mathbf{a}_N$ are the unit outward normals to the N line segments that define the obstacle.

on the position of the manipulator joints, and that furthermore, these constraints are at most quadratic in the position variables. This allows the entire problem to be framed as a Disjunctive Quadratic Program and solved efficiently using commercially-available software.

A. Obstacle Avoidance using Disjunctive Linear Constraints

In this section we first review how obstacle avoidance for a vehicle, modeled as a single point, can be expressed using disjunctive linear constraints [6]. Next, we extend this to apply to manipulators.

1) Obstacle Avoidance for Single Point: A polyhedral obstacle, such as that shown in Fig. 2, is defined using N straight-line segments. The vehicle collides with the obstacle if its position $\mathbf{x}_t$ at any time t is within the obstacle. In order to avoid collision with a given obstacle defined as in Fig. 2 its position must satisfy, for all time steps $t = 1, \ldots, k$:

$$\bigvee_{i=1 \ldots N} \mathbf{a}_i^T \mathbf{x}_t > b_i. \tag{2}$$

Hence the constraint that ensures there is no collision with a given obstacle is a disjunction of linear constraints on the position of the vehicle at all time steps t. Multiple obstacles lead to a conjunction of these disjunctions. If there are M obstacles, where obstacle j is defined by N_j straight lines of the form $\mathbf{a}_{ij}^T \mathbf{x} = b_{ij}$, then collision is avoided if and only if:

$$\bigwedge_{j=1,\ldots,M} \bigvee_{i=1,\ldots,N} \mathbf{a}_{ij}^T \mathbf{x}_t > b_{ij}. \tag{3}$$

2) Obstacle Avoidance for Manipulators: Using (3), joint l avoids collision with the M obstacles if and only if, for all time steps t,

$$\bigwedge_{j=1,\ldots,M} \bigvee_{i=1,\ldots,N} \mathbf{a}_{ij}^T \mathbf{x}_l^{(t)} > b_{ij}. \tag{4}$$

We assume that the links are straight lines between adjacent joints. The position of a point on the link between joint l and joint $l+1$, is given by (5) where $\lambda \in [0, 1]$.

$$\mathbf{x} = \lambda \mathbf{x}_l^{(t)} + (1 - \lambda)\mathbf{x}_{l+1}^{(t)} \tag{5}$$

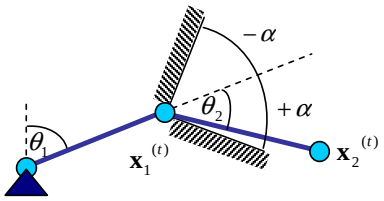

Fig. 3. Planar manipulator with symmetric joint angle limits on θ_2.

We can choose a point on the link, for example the middle (halfway between each joint), and constrain this point to miss obstacles by setting $\lambda = \frac{1}{2}$ and using (4).

$$\bigwedge_{j=1,\ldots,M} \bigvee_{i=1,\ldots,N} \mathbf{a}_{ij}^T \left(\lambda \mathbf{x}_l^{(t)} + (1 - \lambda)\mathbf{x}_{l+1}^{(t)}\right) > b_{ij}. \tag{6}$$

We therefore pick suitably spaced values of λ between zero and one, and apply (6) for these values. In this way we can constrain the link to avoid collision with obstacles. By using finer spacing of the values of λ, we can arbitrarily improve the performance.

Note that for fixed λ, (6) is linear in the position of the joints. Hence the constraints that the joints and the links do not collide with the obstacles are conjunctions of linear disjunctions. This will be used to pose the path planning problem as a Disjunctive Program.

B. Manipulator Kinematic Constraints using Quadratic Constraints

In this section we show that the kinematic constraints on the manipulator can be expressed using quadratic constraints on the joint positions.

We assume that a manipulator is made up of L joints connected by straight links, where joint 0 is the base, and joint L is the endpoint. This arrangement leads to kinematic constraints on the positions of the joints. In particular, the distance between joint l and joint $l+1$ must be exactly d_l, where d_l is the length of link l, which connects joint l and joint $l+1$.

These kinematic constraints can be expressed as follows, for all $l = 0, \ldots, L-1$ and for all t:

$$(\mathbf{x}_{l+1}^{(t)} - \mathbf{x}_l^{(t)})^T (\mathbf{x}_{l+1}^{(t)} - \mathbf{x}_l^{(t)}) = d_l \tag{7}$$

Note that these constraints are quadratic in the joint positions. This will be used to pose the path planning problem as a Disjunctive Quadratic Program.

Additional kinematic constraints may arise due to joint angle restrictions. For some manipulator configurations, joint angle restrictions can be expressed as quadratic constraints on the joint positions. Here, we demonstrate that this is true for all planar manipulators with joint angle ranges that are symmetric about zero.

Consider the planar manipulator shown in Fig. 3. The angle of joint 1, θ_2, can range between $+\alpha$ and $-\alpha$ degrees. Taking the dot product of the vector $\mathbf{x}_2^{(t)} - \mathbf{x}_1^{(t)}$ and the vector $\mathbf{x}_1^{(t)} - \mathbf{x}_0^{(t)}$ gives:

$$(\mathbf{x}_2^{(t)} - \mathbf{x}_1^{(t)})^T (\mathbf{x}_1^{(t)} - \mathbf{x}_0^{(t)}) = d_1 \cdot d_0 \cdot cos\theta_2 \tag{8}$$

The restricted range of joint 2 therefore yields the following constraint on the joint positions:

$$(\mathbf{x}_2^{(t)} - \mathbf{x}_1^{(t)})^T(\mathbf{x}_1^{(t)} - \mathbf{x}_0^{(t)}) \geq d_1 \cdot d_0 \cdot cos\alpha \qquad (9)$$

This, once again, is a quadratic constraint on the joint positions.

Quadratic constraints can also be used for all planar manipulators with non-symmetric joint angle ranges less than $180°$. This is achieved by combining constraints such as (9) with constraints on the dot product between link l and the vector perpendicular to link $l+1$. The full development is not given here for brevity. Finally, many configurations of three-dimensional manipulators with joint angle restrictions can be handled. A concise description of which manipulators can be handled is an area of ongoing research.

To summarize, kinematic constraints on a manipulator can be expressed as quadratic constraints on the positions of the joints at all time steps t.

C. Manipulator Dynamic Constraints

In this section we specify constraints representing the simplified dynamics of the manipulator. In particular, we constrain the velocities of the joint positions.

We assume that the speed of the manipulator is kept small by the user. Since the joint angular velocities are far from their physical limits, we can constrain the velocities of the joint positions, rather than that of the joint angular velocities.

By taking an Euler approximation of the joint velocity, the velocity constraints take the form:

$$\frac{\mathbf{x}_l^{(t+1)} - \mathbf{x}_l^{(t)}}{\Delta t} \leq V_{max}, \qquad (10)$$

where Δt is the duration of the discrete time interval.

D. Optimality

A number of different optimality criteria can be encoded using linear or quadratic functions of the joint positions, depending on the application. Here we show that the average kinetic energy of the manipulator can be expressed as a quadratic function.

We use $\mathbf{x}_{cl}^{(t)}$ to donate the vector from the origin to the center of mass of link l at time t, and m_l to donate the mass of link l. Then the linear approximation of the velocity of the center of mass of link l is:

$$\mathbf{v}_l^t = \frac{\mathbf{x}_{cl}^{(t-1)} - \mathbf{x}_{cl}^{(t)}}{\Delta t} \qquad (11)$$

Then the average kinetic energy of the manipulator over the planning horizon is:

$$J = \frac{1}{k} \sum_{t=1,\ldots,k} \sum_{l=1,\ldots,L} \frac{1}{2} m_l \mathbf{v}_l^{(t)T} \mathbf{v}_l^{(t)} \qquad (12)$$

Since $\mathbf{x}_{cl}^{(t)}$ is on the link, it is a linear function of the joint positions $\mathbf{x}_l^{(t)}$ and $\mathbf{x}_{l+1}^{(t)}$, as in (5). From (11), $\mathbf{v}_l^t$ is a linear function of the joint positions. Hence the average kinetic energy J is a quadratic function of the joint positions. This means that a minimum kinetic energy criterion can be used in a Disjunctive Quadratic Program. Alternative criteria such as minimum time can be encoded in a manner similar to that used for vehicle path planning [7].

The final requirement, that the manipulator endpoint moves from the start position $\mathbf{s}$ to the goal position $\mathbf{g}$, is encoded as follows:

$$\mathbf{x}_L^{(0)} = \mathbf{g} \qquad \mathbf{x}_L^{(k)} = \mathbf{s} \qquad (13)$$

IV. Summary

The optimal trajectory planning problem for manipulators, with obstacles in the workspace, can therefore be expressed as a Disjunctive Quadratic Program. The decision variables in this program are the positions of the joints at time $t = 1, \ldots, k$. The kinematic and dynamic constraints on the trajectory can be expressed using linear and quadratic constraints, while various optimality criteria can be expressed as linear or quadratic functions of the decision variables. Finally, obstacle avoidance can be expressed using disjunctions of linear constraints.

This method allows the use of efficient Disjunctive Programming techniques that are commercially available. These constrained optimization techniques are complete, in that they will return a feasible solution if one exists. Also, since planning is carried out in the workspace domain, mapping the obstacles into the configuration space of the manipulator is unnecessary.

A final remark is that, in the case of a manipulator with a mobile base, such as a rover with a manipulator arm, the trajectory planning for the base mobility system (the rover) can be carried out along with the trajectory planning for the manipulator, in a unified manner. In other words, in a cluttered environment, a task can be specified in terms of the motion of the manipulator endpoint, and by solving a single Disjunctive Quadratic Program, our method can plan a trajectory to achieve this goal by moving both the rover *and* the manipulator in a safe, optimal manner.

References

[1] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots under obstacle and dynamic balance constraints," in *Proc. IEEE Int. Conf. Robot and Autom.*, 2001.

[2] R. Bohlin and L. Kavraki, "Path planning using lazy prm," in *Proc. IEEE Int. Conf. Robot and Autom.*, 2000.

[3] E. Balas, "Disjunctive programming," *Annals of Discrete Math*, vol. 5, pp. 3–51, 1979.

[4] A. Prekopa, *Nonlinear and Mixed-Integer Programming - Fundamentals and Applications*. Oxford University Press, 1995.

[5] R. Stallman and G. Sussman, "Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis," *Journal of Artificial Intelligence*, vol. 8, 1977.

[6] T. Schouwenaars, B. D. Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. European Control Conference*, 2001.

[7] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proc. American Control Conference*, 2002.

[8] T. Leaute and B. C. Williams, "Coordinating agile systems through the model-based execution of temporal plans," in *Proc. 20th National Conference on Artificial Intelligence*, 2005.

[9] ILOG. CPLEX Product Datasheet. [Online]. Available: http://www.ilog.com/download/docs/DS-CPLEX2005.pdf