On Manipulation Planning

Juan Manuel Ahuactzin *
jmal@udlapvms.pue.udlap.mx
School of Engineering Science
Simon Fraser University
Burnaby, B.C. Canada V5A 1S6

Kamal Gupta
kamal@cs.sfu.ca
School of Engineering Science
Simon Fraser University
Burnaby, B.C. Canada V5A 1S6

Emmanuel Mazer
manu@lifia.imag.fr
LIFIA, Grenoble, France

Abstract

An emerging paradigm in solving the classical motion planning problem (among static obstacles) is to capture the connectivity of the configuration space using a finite (but possibly large) set of landmarks (or nodes) in it [14, 1, 5, 7, 15]. In this paper, we extend this paradigm to manipulation planning problem, where the goal is to plan the motion of a robot so that it can move a given object from an initial configuration to a final configuration while avoiding collisions with the static obstacles in the environment. Our specific approach adapts *Adraine's Clew Algorithm* that has been shown effective for classical motion planning problem [14, 1]. In our approach, landmarks are placed in lower dimensional submanifolds of the composite configuration space. These landmarks represent stable grasps that are reachable from the initial configuration. From each new landmark, the planner attempts to reach the goal configuration by executing a local planner, again in a lower (but different) dimensional submanifold of the composite configuration space. We have implemented this approach and present initial experiments with a simple 2-dof planar arm among polygonal obstacles. This simplified domain allows us to better understand the approach.

1.0 Introduction

An important problem toward achieving autonomous task planning systems is that of automatic manipulation planning [10]. One version of this problem can be stated as follows : plan the motion of a robot so that it can move a given object from an initial configuration to a final configuration while avoiding collisions with the static obstacles in the environment. It is well known that manipulation planning is computationally more complex than the classical motion planning (piano mover's) problem [10]. It involves dynamically changing grasp and ungrasp motions that change the composite configuration space. It is known [10] that the manipulation problem can be decompsed into a sequence of subpaths – lying in lower dimensional submanifolds – seperated by grasp and ungrasp operations. These paths are called *transfer* and *transit* paths [11, 8]. Consequently, a manipulation path is composed of a sequence of transfer and transit paths. Let us introduce the problem using Figure 1. The goal for the planar arm is to take the rectangular object from the initial position (at the top of the figure 1a) to the final position (at the bottom). To move this object, the arm must grasp (with the end-effector) one of the object's edges as shown in 1b. The robot then moves toward (transfer path) the goal but the obstacles prevent the robot from continuing (c). The robot then ungrasps the object and then regrasps the object at a new grasp (transit path) (d). This new grasp permits the robot to reach the goal position (e and f).

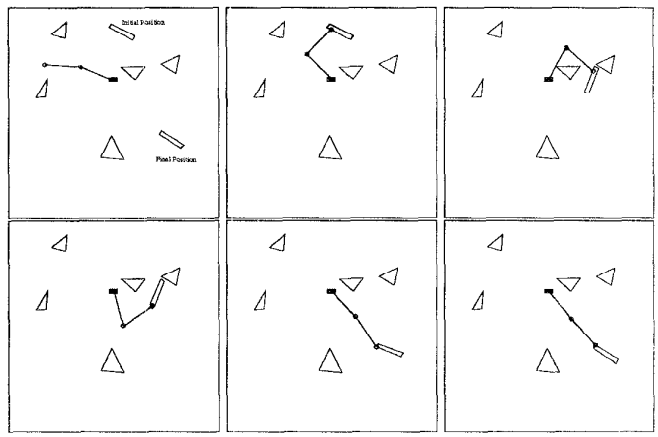

Figure 1: A manipulation path for a 2-dof planer arm.

Some previous works have addressed related problems in simple domains. [17] was the first work dealing with manipulation planning for a PUMA equipped with a parallel jaw gripper. Their underlying planner, however, was limited to manipulators with 3 or 4 degrees of freedom. In [12], the case of a circular movable ob-

*On leave from Universidad de las Americas, Puebla (Mexico).

Acknowledgemnts : This work builds upon the Ph.D. thesis of the first author and was supported in part by an NSERC grant.

Proceedings of the 1995 IEEE International Symposium on Assembly and Task Planning (ISATP '95)
0-8186-6995-0/95 $10.00 © 1995 IEEE

ject and a circular robot in 2D polygonal environment is investigated. A cell decomposition is generated and then used to obtain a finite number of nodes for representing the accessible configurations. The number of the nodes depend on the number of vertices of the obstacles. The problem is then reduced to an exploration of the graph. An algorithm for a polygonal mobile robot with multiple movable objects is presented in [2]. This implementation uses a finite number of grasps and placements for each of the movable object. A manipulation graph with a finite number of nodes is constructed incrementally to solve the problem. Note that this approach breaks down for redundant robots, i.e., if the possible configurations of the robot for a given grasp are infinite. In [8], dual-arm manipulation planning in 2D environments is considered. A finite number of grasp points are defined for the movable object and a randomized planning thechnique [3] is used to obtain a free path for the movable object. During the search, the planner makes sure that at least one grasp exists for each of the robots (using inverse kinematics). The robots follow the trace of the grasp point trying two keep the same type of grasp. When the trace can not be followed any further, a transit path is executed to change the grasp. [9] has demonstrated some manipulation tasks in sparse 3-dimensional environments with multiple manipulators for PUMA 560 type manipulators. The approach, however, needs that inverse kinematics solution be available for each of the arms.

One of the emerging paradigms (although there are differences in the specific approaches) in solving motion planning problems is to capture the connectivity of the configuration space using a finite (but possibly large) set of landmarks (or nodes) in it [14, 1, 5, 7, 15]. [14, 1] presented an algorithm, called *Ariadne's Clew Algorithm* (ACA) that can be used to search for a path in a continuous domain. It is composed of two sub-algorithms : *EXPLORE* and *SEARCH*. The *EXPLORE* algorithm "explores" the reachable space from a given initial point by placing landmarks in it. The landmarks are so placed that a path from the initial position to any landmark is known. The *SEARCH* algorithm is a local planner that verifies if the goal configuration can be reached from a newly placed landmark. Both algorithms are expressed and solved as optimization problems using a special set of paths – the Manhattan paths. The completeness of this algorithm has been proved in [1].

In this paper, we extend this paradigm to manipulation planning problem. Landmarks are placed in lower dimensional sub-manifolds of the composite configuration space. These landmarks represent stable grasps that are reachable from the initial configuration. From each new landmark, the planner attempts to reach the goal configuration by executing a local planner, again in a lower (but different) dimensional submanifold of the composite configuration space. Our initial experiments are with a simple 2-dof planar arm among polygonal obstacles. In our preliminary examples, a pre-defined set of grasp points on the movable object is given. Furthermore, it is assumed all collision-free placements of the movable object are stable. This simplified domain allows to better understand the approach. We are now in the process of extending the approach to realistic 3-D environments and manipulators with many degrees of freedom.

Our approach offers the following advantages over the previous methods for manipulation planning (i) it does not assume that an inverse kinematics solution for the manipulator is available, and (ii) it does not assume a finite number of robot configurations for each grasp, i.e., it is directly applicable to redundant manipulators.

1.1 Basic Definitions and Problem

1.1.1 Notation

Let $\mathcal{W}$ denote the workspace of the robot $\mathcal{A}$. A configuration of the robot in $\mathcal{W}$ is completely specified by $\hat{q}_{\mathcal{A}} = (x_1, x_2, \ldots, x_n)$, where n is the number of degrees of freedom of $\mathcal{A}$. $\mathcal{C}_{\mathcal{A}}$ denotes the configuration space of $\mathcal{A}$ [13]. The static obstacles in $\mathcal{W}$ are denoted by $\mathcal{B}_{i=1,2,3,\ldots}$ and the *movable object* by $\mathcal{M}$. Let $\hat{q}_{\mathcal{M}} = (y_1, y_2, \ldots, y_k)$ denote the relative configuration of the object refence frame $\mathcal{F}_{\mathcal{M}}$ to the world's reference frame $\mathcal{F}_{\mathcal{W}}$. The configuration space of $\mathcal{M}$ is denoted by $\mathcal{C}_{\mathcal{M}}$. We will make the assumption that $\mathcal{C}_{\mathcal{A}}$ and $\mathcal{C}_{\mathcal{M}}$ are compact (closed and bounded) sets.

The configuration space of the entire system is then $\mathcal{C} = \mathcal{C}_{\mathcal{A}} \times \mathcal{C}_{\mathcal{M}}$ and its dimension is $(n + k)$. The projections of $\hat{q} \in \mathcal{C}$ in $\mathcal{C}_{\mathcal{A}}$ and $\mathcal{C}_{\mathcal{M}}$ will be denoted and defined as follows : $\pi_{\mathcal{A}} : \hat{q} \in \mathcal{C} \rightarrow \hat{q}_{\mathcal{A}} \in \mathcal{C}_{\mathcal{A}}$

$\pi_{\mathcal{M}} : \hat{q} \in \mathcal{C} \rightarrow \hat{q}_{\mathcal{M}} \in \mathcal{C}_{\mathcal{M}}$

1.1.2 Transit, Transfer and Manipulation Paths

Let ${}^{\mathcal{W}}\hat{q}_{\mathcal{M}}$ denote a stable configuration of $\mathcal{M}$ described w.r.t. $\mathcal{F}_{\mathcal{W}}$. The submanifold $\mathcal{C}^{stable}_{{}^{\mathcal{W}}\hat{q}_{\mathcal{M}}} \subset \mathcal{C}$ is defined as $\mathcal{C}^{stable}_{{}^{\mathcal{W}}\hat{q}_{\mathcal{M}}} = \{\hat{q} \in \mathcal{C} | \pi_{\mathcal{M}}(\hat{q}) = {}^{\mathcal{W}}\hat{q}_{\mathcal{M}}\}$. A *transit path* is a continuous map $\hat{\varphi} : [0, 1] \rightarrow \mathcal{C}^{stable}_{{}^{\mathcal{W}}\hat{q}_{\mathcal{M}}}$. When the robot is moving with the movable object grasped in its gripper, the configuration of $\mathcal{M}$ relative to the gripper remains constant and the configuration of $\mathcal{M}$ w.r.t $\mathcal{F}_{\mathcal{W}}$ is changing. Let us denote these relative configurations by ${}^{G}\hat{q}_{\mathcal{M}}$ and ${}^{\mathcal{W}}\hat{q}_{\mathcal{M}}$ respectively. Moreover, let $\Phi({}^{\mathcal{W}}\hat{q}_{\mathcal{M}})$ be the function that transforms the configuration of $\mathcal{M}$ from $\mathcal{F}_{\mathcal{W}}$ to the gripper's reference frame $\mathcal{F}_G$. For a given grasp configuration ${}^{G}\hat{q}_{\mathcal{M}}$, we define the sub-mainfold $\mathcal{C}^{grasp}_{{}^{G}\hat{q}_{\mathcal{M}}} \subset \mathcal{C}$ as $\mathcal{C}^{grasp}_{{}^{G}\hat{q}_{\mathcal{M}}} = \{\hat{q} \in \mathcal{C} | \Phi(\pi_{\mathcal{M}}(\hat{q})) = {}^{G}\hat{q}_{\mathcal{M}}\}$. Note that the dimension of this submanifold is n (the number of dof of $\mathcal{A}$), and that Φ depends on $\hat{q}_{\mathcal{A}}$. A *transfer path* is continuous map $\hat{\tau} : [0, 1] \rightarrow \mathcal{C}^{grasp}_{{}^{G}\hat{q}_{\mathcal{M}}}$. Let $\mathcal{CB}$ denote the configuration space obstacles. The free space $\mathcal{C}_{free}$ is defined as $\mathcal{C} \backslash \mathcal{CB}$. A *free transfer path* $\hat{\varphi}$ (resp. free transit path, $\hat{\varphi}$) is then defined as a mapping $\hat{\varphi} : [0, 1] \rightarrow \mathcal{C}_{free} \cap \mathcal{C}_{\mathcal{W}\hat{q}_{\mathcal{M}}}$.

A *single manipulation path* $\hat{\sigma}$ is defined as a concatenation (product) [10] of a free transit path $\hat{\varphi}$ and a free transfer path $\hat{\tau}$, $\hat{\sigma} = \hat{\varphi} * \hat{\tau}$. A *manipulation path of order* ℓ, denoted by $\hat{\sigma}^{\ell}$ is defined as a concatenation of ℓ single manipulation paths, i.e., $\hat{\sigma}^{k} = \hat{\sigma}_1 * \hat{\sigma}_2 * \ldots * \hat{\sigma}_{\ell}$.

Proceedings of the 1995 IEEE International Symposium on Assembly and Task Planning (ISATP '95)
0-8186-6995-0/95 $10.00 © 1995 IEEE

Let $\mathcal{P} = \{{}^{\mathcal{W}}\hat{q}_1, {}^{\mathcal{W}}\hat{q}_2, \ldots, {}^{\mathcal{W}}\hat{q}_k\}$ be the set of stable placements of $\mathcal{M}$ in $\mathcal{W}$. The entire sub-mainfold generated by $\mathcal{P}$ is then, $\mathcal{CP} = \mathcal{C}^{stable}_{{}^{\mathcal{W}}\hat{q}_{\mathcal{M}_1}} \cup \mathcal{C}^{stable}_{{}^{\mathcal{W}}\hat{q}_{\mathcal{M}_2}} \cup \ldots \cup \mathcal{C}^{stable}_{{}^{\mathcal{W}}\hat{q}_{\mathcal{M}_k}}$. Similarly, let $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$ be the set of grasps and $\{{}^{G}\hat{q}_{\mathcal{M}_1}, {}^{G}\hat{q}_{\mathcal{M}_2}, \ldots, {}^{G}\hat{q}_{\mathcal{M}_m}\}$ the set of relative position of $\mathcal{M}$ w.r.t. $\mathcal{F}_G$ resulting from $\mathcal{G}$. The set of submanifold generated by $\mathcal{G}$ is then, $\mathcal{CG} = \mathcal{C}^{grasp}_{{}^{G}\hat{q}_{\mathcal{M}_1}} \cup \mathcal{C}^{grasp}_{{}^{G}\hat{q}_{\mathcal{M}_2}} \cup \ldots \cup \mathcal{C}^{grasp}_{{}^{G}\hat{q}_{\mathcal{M}_m}}$.

The manipulation path planning problem can now be stated as follows : Given an initial configuration $\hat{q}_\circ \in \mathcal{C}_{free}$ and a desired final configuration $\hat{q}_{\mathcal{M}\bullet}$ of $\mathcal{M}$, a set of stable placements, $\mathcal{P}$, and a set of grasps, $\mathcal{G}$, find a manipulation path $\hat{\sigma}^k$ such that $\hat{\sigma}^k(0) = \hat{q}_\circ$ and $\pi_{\mathcal{M}}(\hat{\sigma}^k(1)) = \hat{q}_{\mathcal{M}\bullet}$.

1.2 Overview of the ACA

The algorithm is composed of two sub-algorithms : $EXPLORE$ and $SEARCH$. Let $\hat{q}_\circ = (\hat{q}_{\mathcal{A}_\circ}, {}^{\mathcal{W}}\hat{q}_{\mathcal{M}\circ})$ be the initial configuration of the system. $EXPLORE$ tries to identify the set of reachable configurations in $\mathcal{CP} \cap \mathcal{CG}$ from $\hat{q}_\circ$. It does so by placing landmarks in $\mathcal{CG} \cap \mathcal{CP}$ in such a way that a manipulation path from $\hat{q}_\circ$ to any landmark is known. Let L_n denote the n^{th} landmark, with ${}^{\mathcal{W}}\hat{q}_{\mathcal{M}_{(n)}} = \pi_{\mathcal{M}}(L_n)$ and ${}^{G}\hat{q}_{\mathcal{M}_{(n)}}$ are the configuration of $\mathcal{M}$ w.r.t. $\mathcal{F}_{\mathcal{W}}$ and $\mathcal{F}_G$ respectively[1]. Using this notation we have that $L_n \in \mathcal{C}^{stable}_{{}^{\mathcal{W}}\hat{q}_{\mathcal{M}_{(n)}}} \cap \mathcal{C}^{grasp}_{{}^{G}\hat{q}_{\mathcal{M}_{(n)}}}$. $EXPLORE$ tries to spread the landmarks all over the connected space from $\hat{q}_\circ$. To do so, it tries to put the next landmark as far as possible from the current ones. By construction, a new landmark is reachable from at least one of the previously placed landmarks via a *single manipulation path*. Each time we obtain a new landmark, we use $SEARCH$ to try to go to the goal. The $SEARCH$ algorithm, a fast local planner, tries to plan a free *transfer path* from the current landmark L_n to the final configuration ${}^{\mathcal{W}}\hat{q}_{\mathcal{M}\bullet}$ by minimizing the distance between the current configuration of $\mathcal{M}$ and ${}^{\mathcal{W}}\hat{q}_{\mathcal{M}\bullet}$; if it fails another landmark is placed by $EXPLORE$. Essentially, a tree of landmarks is formed with $\hat{q}_\circ$ as the root. Figure 2 shows the tree and the configurations where the landmarks have been placed. The submanifolds of $\mathcal{CP}$ are schematically represented by a rectangle and those of $\mathcal{CG}$ by an ellipse. Each of the landmarks (drawn with a $\bullet$), is connected to its parent by a single manipulation path (continuous line). In order to give an idea of the reachable configurations in $\mathcal{CP} \cap \mathcal{CG}$ from a particular landmark, we have used dashed rectangles to show the reachable configurations from the root (shown as $\square$). The number of different grasps reachable from a placement is then the number of ellipses intersecting the square and the number of different placements reachable with a given grasp is the number of rectangles intersecting the ellipse.

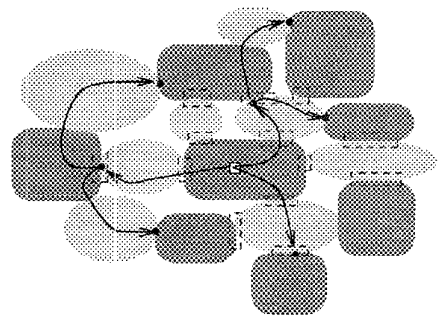

Figure 2: Schematic representation of the placement of landmarks.

More formally, let $\mathcal{L}_n$ denote the set of existing landmarks at step n, and Σ_n the set of single manipulation paths which start from the configuration corresponding to one of the landmarks, L_i and terminate at the grasp submanifold j accessible from L_i, i.e., $\Sigma_n = \{\hat{\sigma} : \hat{\sigma}(0) = L_i \in \mathcal{L}_n \text{ and } \hat{\sigma}(1) \in \mathcal{C}^{grasp}_{{}^{G}\hat{q}_{\mathcal{M}_j}} \cap \mathcal{CP}\}$.

To illustrate, suppose, $n = 2$ and $\mathcal{L}_2 = \{L_1, L_2\}$. Now given a manipulation path $\hat{\sigma} \in \Sigma_2$ (that is, a single manipulation path starting either at L_1 or L_2), the algorithm tries to choose a path $\hat{\sigma}_2$ that maximizes the distance[2] $||\hat{\sigma}(1) - \mathcal{L}_2||$. The extremity of this path gives us L_3, i.e., $\hat{\sigma}_2 : \max_{\hat{\sigma}\in\Sigma_2} \min_{L_i\in\mathcal{L}_2} ||\hat{\sigma}(1) - L_i||$, and $L_3 = \hat{\sigma}_2(1)$ and $\mathcal{L}_3 = \mathcal{L}_2 \cup \{L_3\}$.

We can therefore express the $EXPLORE$ algorithm as an optimization problem :

$$EXPLORE(n) = \left\{ \begin{array}{l} \text{Maximize } ||\mathcal{L}_n - \hat{\sigma}(1)|| \\ \hat{\sigma} \in \Sigma_n \end{array} \right.$$

It has been shown in [1] that if $\exists \hat{q}_\bullet \in \mathcal{CP} \cap \mathcal{CG}$ such that $\pi_{\mathcal{M}}(\hat{q}_\bullet) = \hat{q}_{\mathcal{M}\bullet}$ and $\hat{q}_\bullet$ is an accessible configuration from $\hat{q}_\circ$ then $\forall \varepsilon \geq 0\ \exists N$ such that at the N^{th} iteration of $EXPLORE$, exists a landmark at a distance ε from $\hat{q}_\bullet$.

$SEARCH$ algorithm is essentially a fast local planner that verifies if a given configuration is reachable from a landmark. Let ε denote the desired resolution and we call the $SEARCH$ algorithm, $SEARCH\_TRANSFER(L_n, {}^{\mathcal{W}}\hat{q}_{\mathcal{M}\bullet})$ which returns the value *true* if it finds a path and *false* otherwise. The ACA is written as follows :

```
ARIADNE'S_CLEW(q̂₀, ^W q̂_M•)
begin
    n = 0;
    L_1 = {L_1 = q̂₀};
    do
        n = n + 1;
        Place a new landmark L_{n+1} with EXPLORE(n);
        ε_n = ||L_{n+1} - L_n||;
        L_{n+1} = L_n ∪ {L_{n+1}};
        transfer = SEARCH_TRANSFER(L_{n+1}, ^W q̂_M•)
    while( ¬transfer and (ε_n > ε))
```

[1] Note that ${}^{\mathcal{W}}\hat{q}_{\mathcal{M}_{(n)}}$ represents the n^{th} placement corresponding to L_n whereas ${}^{\mathcal{W}}\hat{q}_{\mathcal{M}_n}$ represents the n^{th} placement in $\mathcal{P}$, and in general (in fact, often) these will be different. Similary, we have written ${}^{G}\hat{q}_{\mathcal{M}_{(n)}}$ to specify the grasp corresponding to L_n and make the distinction with the n^{th} grasp of $\mathcal{G}$.

[2] Note that this is a distance between a point and a set.

Proceedings of the 1995 IEEE International Symposium on Assembly and Task Planning (ISATP '95)
0-8186-6995-0/95 $10.00 © 1995 IEEE

if ($transfer$)
return($history\_of(L_{n+1})$);
else not path exist with resolution ε;
end

The "*history_of*" routine is simply a concatenation of the manipulation paths of the ancestors of L_{n+1} and the transfer path found by *SEARCH*.

1.3 Implementing *EXPLORE* and *SEARCH* with Manhattan Paths

1.3.1 Manhattan paths

As is probably apparant by now, we will implement EXPLORE and SEARCH as optimization problems, however, with a special class of paths – the *Manhattan paths* which consist of moving one robot link at a time. The main motivation in considering Manhattan motions is that (i) they can be represented by a vector of $\mathbb{R}^n$, (note that in general a path is represented by a function not a vector), (ii) they define a naturally redundant search space which is well suited for writing the trajectory planning problem as an optimization problem in $\mathbb{R}^n$, and (iii) the class is resolution complete in that if a trajectory exists from one configuration $\hat{q}_0$ to another $\hat{q}_\bullet$ and the minimal distance of this trajectory to the $\mathcal{C}$-*obstacles* is $\varepsilon > 0$, then there exists a Manhattan path from $\hat{q}_0$ to $\hat{q}_\bullet$ [1].

Given a continuous space $X \subset \mathbb{R}^n$ and $\hat{q}_0 = (x_1, \ldots, x_n) \in X$, we define a single Manhattan path in X starting from $\hat{q}_0$ as the function $\hat{\gamma} : [0,1] \rightarrow X$ where for $\alpha \in [0,1]$:

$$\hat{\gamma}(\alpha) = (\gamma_1(\alpha), \gamma_2(\alpha), \ldots, \gamma_n(\alpha)) \in X \quad \text{and}$$

$$\gamma_i(\alpha) = \begin{cases} x_i & \text{for} \quad 0 \leq \alpha \leq \frac{(i-1)}{n} \\ x_i + \Delta_i * (n * \alpha - i + 1) & \text{for} \quad \frac{(i-1)}{n} \leq \alpha \leq \frac{i}{n} \\ x_i + \Delta_i & \text{for} \quad \frac{i}{n} \leq \alpha \leq 1 \end{cases}$$

with $\Delta_i \in \mathbb{R}$ being the range of motion for joint i and is carried out in duration $\frac{1}{n}$. $\hat{\gamma}$ is therefore completely defined by $\hat{\gamma} = (\Delta_1, \Delta_2, ..., \Delta_n)$. Note that the semantic of this path is "move link 1 a distance Δ_1" followed by "move link 2 a distance Δ_2", and so on. Furthermore, the product of ℓ single Manhattan paths in X is a Manhattan path of order ℓ. Let $\hat{q}_0 \in X$ be a point in X. The Manhattan path space of order k at $\hat{q}_0$, denoted by $\Omega(X; \hat{q}_0, k)$ is the set of all the Manhattan paths of order $\ell \leq k$ starting at the point $\hat{q}_0$.

We now denote the Manhattan path space of order k in $\mathcal{C}^{stable}_{\mathcal{W}\hat{q}_\mathcal{M}}$ starting at $\hat{q} = (\hat{q}_\mathcal{A}, {}^{\mathcal{W}}\hat{q}_\mathcal{M})$ as $\Omega(\mathcal{C}^{stable}_{\mathcal{W}\hat{q}_\mathcal{M}}; \hat{q}, k)$. For brevity, we simply use Ω. Let us consider the configurations of the robot and the movable object shown in the figure 1. We can graphically represent the space of Manhattans paths of order 1 ($\hat{\gamma} = (\Delta_1, \Delta_2)$). Note that $\Delta_1, \Delta_2 \in [-2\pi, 2\pi]$ and that the space $\Omega(\mathcal{C}_{\mathcal{W}\hat{q}_\mathcal{M}}; \hat{q}, 1)$ can be represented by $[-2\pi, 2\pi] \times [-2\pi, 2\pi]$ and is shown in Figure 3. The space is formed by two complementary subsets, denoted by $\Omega\mathcal{B} = \{\hat{\gamma}^k \in \Omega : \exists \alpha \in [0,1] : \hat{\gamma}^k(\alpha) \in \mathcal{CB}\}$, and $\Omega_{free} = \Omega / \Omega\mathcal{B}$. Note that $\Omega\mathcal{B}$ (Ω_{free} resp.) depends on $\mathcal{C}^{stable}_{\mathcal{W}\hat{q}_\mathcal{M}}, \hat{q}$ and k. Because of brevity we are not showing it explicitly in our notation. For the initial configuration of the robot shown in figure 1, the resulting $\Omega\mathcal{B}$ (dark region) and Ω_{free} (white region) are shown in figure 3.

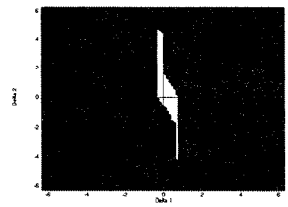

Figure 3: The $\Omega\mathcal{B}$ and Ω_{free} for a Manhattan path space of order 1.

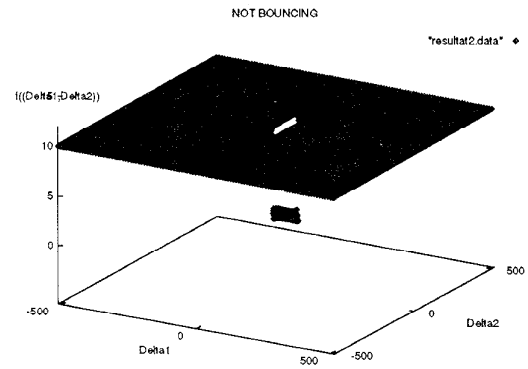

Figure 4: Potential space obtained by f_a.

A similar construction is used to define Manhattan space for the transfer trajectories.

1.3.2 Planning Paths: Optimization in Manhattan Path Space

Let $\hat{q}_0 = (\hat{q}_{\mathcal{A}_0}, \hat{q}_{\mathcal{W}_{\mathcal{M}_0}})$ be a configuration in $\mathcal{CP}$ and ${}^{G}\hat{q}_{\mathcal{M}_\bullet}$ be the configuration for a grasp of $\mathcal{M}$. The transit path planning problem can now be stated as a minimization problem :

$$\min_{\hat{\gamma}^k \in \Omega(\mathcal{C}^{stable}_{\mathcal{W}\hat{q}_\mathcal{M}}; \hat{q}_0, k)} f_a(\hat{\gamma}^k, {}^{G}\hat{q}_{\mathcal{M}_\bullet}) \quad \text{where}$$

$$f_a(\hat{\gamma}^k, {}^{G}\hat{q}_{\mathcal{M}_\bullet}) = \begin{cases} \|\Phi(\hat{\gamma}^k(1)) - {}^{G}\hat{q}_{\mathcal{M}_\bullet}\| & \text{if } \hat{\gamma}^k \in \Omega_{free} \\ +\infty & \text{otherwise} \end{cases}$$

Clearly, if there is a transit path $\hat{\gamma}^k_\star \in \Omega_{free}$ taking the robot to the goal grasp configuration then $f_a(\hat{\gamma}^k_\star, {}^{G}\hat{q}_{\mathcal{M}_\bullet}) = 0$. See figure 4 for the potential space defined by this function over $\Omega(\mathcal{C}_{\mathcal{W}\hat{q}_\mathcal{M}}; \hat{q}_0, 1) = [-2\pi, 2\pi] \times [-2\pi, 2\pi]$. For the simple planar arm, the f_a is equivalent to the distance between the end-effector and the grasp point (see figure 1), however, for a real robot, a *pre-shaping* of the grasp would be easier to find [4].

A very similar optimization formulation exists for planning transfer paths. Let ${}^{\mathcal{W}}\hat{q}_{\mathcal{M}_\bullet}$ be the goal place-

Proceedings of the 1995 IEEE International Symposium on Assembly and Task Planning (ISATP '95)
0-8186-6995-0/95 $10.00 © 1995 IEEE

ment of $\mathcal{M}$ and $\hat{q}_o = (\hat{q}_{\mathcal{A}_o}, {}^{\mathcal{W}}\hat{q}_{\mathcal{M}o}) \in \mathcal{C}^{stable}_{\mathcal{W}\hat{q}_{\mathcal{M}_o}} \cap^G \hat{q}^{grasp}_{\mathcal{M}_j}$ an initial configuration. A transfer path from $\hat{q}_o$ to $\hat{q}_\bullet$ with $\pi_{\mathcal{M}}(\hat{q}_\bullet) = {}^{\mathcal{W}}\hat{q}_{\mathcal{M}\bullet}$ can be computed as follows :

$$\min_{\hat{\gamma}^k \in \Omega(C^{grasp}_{G\hat{q}_{\mathcal{M}_j}}; \hat{q}_o, k)} fb(\hat{\gamma}^k, {}^{\mathcal{W}}\hat{q}_{\mathcal{M}}) \quad \text{where}$$

$$fb(\hat{\gamma}^k, {}^{\mathcal{W}}\hat{q}_{\mathcal{M}\bullet}) = \begin{cases} \|\pi_{\mathcal{M}}(\hat{\gamma}^k(1)) - {}^{\mathcal{W}}\hat{q}_{\mathcal{M}\bullet}\| & \text{if } \hat{\gamma} \in \Omega_{free} \\ +\infty & \text{otherwise} \end{cases}$$

1.3.3 Searching Ω_{free}: The Bouncing Technique

The actual search space for the minimization problem is Ω_{free} and that the region $\Omega\mathcal{B}$ is quite large. It would therefore be computationally efficient to be able to limit the search explicitly to Ω_{free}. It is clear that all valid paths in Ω_{free} can be represented by a vector $\hat{x} \in I\!R^{n*k}$ but that not all $\hat{x} \in I\!R^{n*k}$ represent a path in Ω_{free}, since $\Delta_i \in \hat{\gamma}^k$ could code an invalid value such that the path $\hat{\gamma}^k$ goes into $\mathcal{CB}$. However, intervals $[\Delta_i^{min}, \Delta_i^{max}]$ can be easily obtained (as in [13]) such that $\forall \Delta_i \subset \hat{\gamma}^k \subset \Omega_{free}$, $\Delta_i \in [\Delta_i^{min}, \Delta_i^{max}]$. The computation of these intervals permits us to rewrite the functions $\gamma_i \in \hat{\gamma}^k$ so that the resulting paths are always collision-free that is to map Δ_i in $[\Delta_i^{min}, \Delta_i^{max}]$. The basic idea is to bounce off the obstacle[1]. Below we give an elegant mathematical representation of this bouncing in terms of a periodic function. The periodic function chosen is a triangular wave with amplitude $a = ||\Delta_i^{max} - \Delta_i^{min}||$ and period $2a$. Any x value (along the horizontal axis) can now be mapped to range $[\Delta_i^{min}, \Delta_i^{max}]$. We use this to redefine the manhattan paths γ_i as follows:

$$\gamma_i(\alpha) = \begin{cases} x_i & \text{for } 0 \le \alpha \le \frac{(i-1)}{n} \\ x_i + Triang(a, \Delta_i)(n\alpha - i + 1) & \text{for } \frac{(i-1)}{n} \le \frac{i}{n} \\ x_i + \Delta_i & \text{for } \frac{i}{n} \le \alpha \le 1 \end{cases}$$

Note that the range of link $(i+1)$ depends on the moves of link $1 \ldots i$, i.e., Δ_{i+1}^{min} and Δ_{i+1}^{max} are functions of Δ_j for $j = 1, 2, \ldots, i$. Such a use of a periodic function to define the Manhattan paths permits us to code only feasible paths; there is no way to code a path belonging to $\Omega\mathcal{B}$. See how figure 4 is transformed in figure 5. The advantage of this coding is obvious in that it is well suited for the optimization technique used to minimize the functions f_a and f_b.

1.3.4 Defining Local Planners

We now illustrate simple local planners for transit paths. The case for transfer paths is similar. Let $\hat{q}_o$ be an initial configuration in $\mathcal{C}_{free}$. A local planer for transit paths can be written as follows :

SEARCH_TRANSIT$(\hat{q}_o, \hat{q}_{\mathcal{M}\bullet}, k)$
begin
  $\hat{q}_1 = \hat{q}_o$;
  $i := 0$;
  do
    i = i+1;
    $\hat{\varphi}_i = \min_{\hat{\gamma}^k \in \Omega_{free}} f_a(\hat{\gamma}^k, \hat{q}_{\mathcal{M}\bullet})$;
    $distance = f_a(\hat{\varphi}_i, \hat{q}_{\mathcal{M}\bullet})$;
    $\hat{q}_i = \hat{\varphi}_i(1)$;
  while(($distance \neq 0$) and ($\hat{q}_i \neq \hat{q}_{i-1}$));
  if ($distance = 0$)
    then return($\hat{\varphi}_1 * \hat{\varphi}_2 * \ldots * \hat{\varphi}_i$);
    else return(*null_path*);
end

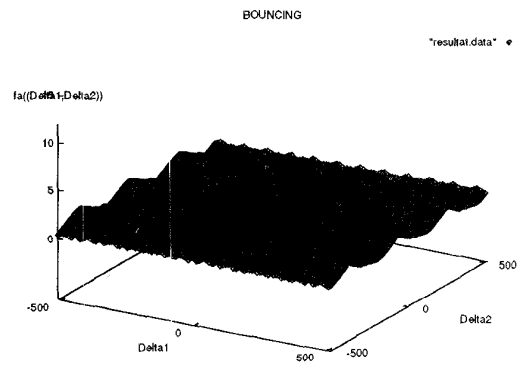

Figure 5: Potential space of f_a obtained with the bouncing technique.

Note that *SEARCH_TRANSIT* returns the null path if no path has been found for the grasp ${}^G\hat{q}_{\mathcal{M}_j}$, i.e., a local minima has been reached. Moreover, using bouncing techniques, we have an explicit coding of any manhattan path $\hat{\gamma}_k \in I\!R^{n*k}$ to a path in Ω_{free}. We will not go into details of this optimization here, however, we have used genetic algorithms to carry out this optimization as in [1].

SEARCH_TRANSIT is now used to build *EXPLORE* as follows. A single manipulation path $\hat{\sigma}$ starting at $L_i \in \mathcal{L}_n$ is generated by a concatenation of the transit path $\hat{\varphi}$ obtained by *SEARCH_TRANSIT*$(L_i, {}^G\hat{q}_{\mathcal{M}_j})$, and a transfer path $\hat{\tau}$ coded by a manhattan path $\hat{\gamma}^k \in \Omega_{free}$. This transfer path is obtained as follows. *EXPLORE* randomly generates m_2 transfer paths with grasp configuration in $\mathcal{G}$, yielding m_2 transfer paths. The submanifold where $\hat{\tau}$ is executed depends in $\hat{\varphi}(1)$. Remember that the algorithm *SEARCH_TRANSIT* returns the null path if no path has been found for the grasp ${}^G\hat{q}_{\mathcal{M}_j}$. In this case, the obtained manipulation path will keep the same grasp that corresponds to L_i (i.e. there is no change of grasp). Furthermore, for each landmark L_i, m_1 transit paths $\hat{\varphi}_j$, $j = 1, m_1$ are generated. Therefore, in all, $m_1 \times m_2$ manipulation paths are generated from each landmark. The manipulation path $\hat{\sigma}$ that such that $\hat{\sigma}(1)$ is farthest away from $\mathcal{L}_n$ is the chosen manipulation path and $\hat{\sigma}(1)$ gives the $n+1^{th}$ landmark.

The algorithm *EXPLORE* can be written as follows :

EXPLORE(n)
begin
  for i = 1 to n
    for j = 1 to m_1
      choose a grasp ${}^G\hat{q}_{\mathcal{M}_j} \in \mathcal{G}$;
      $\hat{\varphi}_{i,j} = SEARCH\_TRANSIT(Li, {}^G\hat{q}_{\mathcal{M}_j})$;

Proceedings of the 1995 IEEE International Symposium on Assembly and Task Planning (ISATP '95)
0-8186-6995-0/95 $10.00 © 1995 IEEE

for k = 1 to m_2
$\hat{\tau}_{i,j,k}$ = random_manhattan_path_with($\hat{\varphi}_{i,j}(1)$);
$\hat{\sigma}_{i,j,k} = \hat{\varphi}_{i,j} * \hat{\tau}_{i,j,k}$;
$\hat{\sigma} = \max_{\hat{\sigma}_{i,j,k}} ||\hat{\sigma}_{i,j,k}(1) - \mathcal{L}_t||$;
return($\hat{\sigma}$);
end

In reality, we use genetic algorithms to solve the above optimization problems in *EXPLORE* as in [1]. The above explanation is a high-level description of this process. In particular, constants m_1 and m_2 are related to the number of generations used in the genetic optimization process. The search space Σ_n used by *EXPLORE* can be represented by : $[1, 2, \ldots, n] \times [1, 2, \ldots, m] \times I\!R^{k*n}$, where $[1, 2, \ldots, n]$ is the id of the starting landmark, $[1, 2, \ldots, m]$ the id of grasp in $\mathcal{G}$ and $I\!R^{k*n}$ represents the transfer paths.

The example shown in figure 1 was solved using the *EXPLORE* and *SEARCH-TRANSIT* as explained in this section. The approximate run time was about 4 minutes on an IPX Sparcstation. About seven landmarks were needed for this example. Admittedly our experiments are somewhat preliminary, however, they show the promise of our approach.

1.4 Conclusions

One of the emerging paradigms in solving motion planning problems is to capture the connectivity of the configuration space by using a finite (but possibly large) set of landmarks (or nodes) in it. In this paper, we extend this paradigm to manipulation planning problem. Our approach offers the following advantages over the previous methods for manipulation planning (i) it does not assume that an inverse kinematics solution for the manipulator is available, and (ii) it does not assume a finite number of robot configurations for each grasp, i.e., it is directly applicable to redundant manipulators.

Our initial experiments are with a simple 2-dof planar arm among polygonal obstacles. Another assumption is that the movable object is stable in any configuration of the free space. This simplified domain allows to better understand the approach.

We are now in the process of extending the approach to realistic 3-D environments and manipulators with many degrees of freedom. We believe that for enviroments in 3-D, the search for a stable placement (or a pre-shaping configuration) can be incorporated in the optimization function. Furthermore, in the current implementation, a local planner is used to to compute the transit paths. This could be augmented by using EXPLORE function for transit paths also.

References

[1] J.M. Ahuactzin : *Le Fil d'Ariane : Une methode de planification générale. Application a la planification automatique des trajectoires*, Ph.D. Tesis of the Institut National Polytechnique de Grenoble. Grenoble, France 1994.

[2] R. Alami, T. Siméon and J.P. Laumond : *A Geometrical Approach to Planning Manipulation Tasks : The Case of Discrete Placements and Grasps*, in Robotics Research 5, The MIT press, Cambridge, 1990.

[3] Jérôme Barraquand, Jean-Claude Latombe: *Robot Motion Planning A Distributed Representation Approach*, Robotics Laboratory, Computer Science Department, Stanford University.

[4] Christian BARD : *Interaction sensori-motrice en robotique : Application a la prehension automatisée pour une main articulée a plusieurs doigts*, Ph.D. Tesis of the Institut National Polytechnique de Grenoble. Grenoble, France 1994.

[5] C. Chen, Yong K. Hwang: *SANDROS: A Motion Planner with Performance Proportional to Task Difficulty*,Proceedings of the 1992 IEEE International Conference on Robotic and Automation, Nice, France - May 1992.

[6] Kamal Kant Gupta, Zhenping Guo: *Motion Planning for Many Degrees of Freedom Sequential Search With Backtracking*,Proceedings of the 1992 IEEE International Conference on Robotic and Automation, Nice, France - May 1992.

[7] Lydia Kavraki and Jean-Claude Latombe : *Randomized Preprocessing of Configuration Space for Path Planning : Articulated Robots*, IROS, 1994.

[8] Yoshihito Koga and Jean-Claude Latombe : *Experiments in Dual-Arm Manipulation Planning*, Proceedings of the 1992 IEEE International Conference on Robotic and Automation,Nice, France - May 1992.

[9] Yoshihito Koga : *On Computing Multi-Arm Manipulation Trajectories*, PhD. Tesis, Department of Mechanical Engineering, Stanford University, 1994.

[10] Jean-Claude Latombe : *Robot Motion Planning*, Kluwer Academic Publishers, Boston 1991.

[11] Jean Paul Laumond and Rachid Alami : A Geometrical Approach to Planning Manipulation Tasks : The Case of Circular Robot and Movable Circular Object Admist Polygonal Obstacles. Tech. Rep. No. 88314, LAAS, Toulouse, France, 1988.

[12] Jean Paul Laumond and Rachid Alami : *A Geometrical Approach to Planning Manipulation Tasks in Robotics*, Tech. Rep. No. 89261, LAAS, Toulouse, France, 1989.

[13] Tomás Lozano-Pérez: *A simple motion-planning algorithm for general robot manipulators*, IEEE Robotics and Automation, June 1987.

[14] E. Mazer, J.M. Ahuactzin, G. Talbi, P. Bessière : "The ariadne's Clew Algorithm", SAB92 Honolulu 1992.

[15] Mark H. Overmars: *A random approach to motion planning*, Utrecht University, Department of Computer Science, The Netherlands 1992.

[16] Nancy S. Pollard and Tomás Lozano-Pérez : *Grasp Stability and Feasibility for an Arm with Articulated Hand*, Proceedings of the 1991 IEEE International Conference on Robotics and Automation.

[17] Pierre Tournassoud, Tomás Lozano-Pérez and Emmanuel Mazer : *REGRASPING*, Proceedings of the 1987 IEEE International Conference on Robotic and Automation, Raleigh, North Carolina USA, March-April 1987.

Proceedings of the 1995 IEEE International Symposium on Assembly and Task Planning (ISATP '95)
0-8186-6995-0/95 $10.00 © 1995 IEEE