Search in the Context of Complex Robot-World Interaction

Sanjiv Singh

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
ssingh@cmu.edu

From: AAAI Technical Report WS-97-10. Compilation copyright © 1997, AAAI (www.aaai.org). All rights reserved.

Abstract

In contrast to typical planning problems, automated earthmoving operations such as digging a trench or leveling a mound of soil pose some additional challenges. First, soil is diffuse and therefore a unique description of the world requires a very large number of variables. Second, the interaction between the robot and the world is very complex to the point that it is not possible to accurately predict the effect of a candidate action. In this paper, I pose planning for earthmoving operations as a problem of constraint optimization. I discuss why on-line search is necessary and present experimental results from a testbed where a robot is tasked to dig a trench.

Introduction

In our laboratory, we develop automation for machines that operate in industries such as mining and construction. Three characteristics distinguish such machines. They are *capable*, *efficient* and *safe*. Capability has to do with scale—forceful interaction with the world requires machines that can generate large forces. Efficiency boils down to performing a task as fast as possible without compromising output. Safety is a matter of ensuring collision free operation and in some cases avoiding tipover.

Used judiciously, models can greatly help in increasing efficiency and safety. By "models" I mean physical, numerical models of mechanisms and their interaction with the world. (Models of planning and sensing are also useful, but I do not address them here.) That is, given a candidate input, these models predict the corresponding output of a (sub) system. Of course we would like invertible models—models that compute the input necessary for a desired effect—but these are rarely available. Good forward models are often the best we can achieve. They provide the proaction that is necessary for successful goal directed activity, at least in the environments in which we must put our autonomous machines. When invertible models are not possible, search is necessary.

In this paper I will concentrate on one task—earthmoving—that highlights the approach espoused above. I have a robot manipulator that is equipped with a scoop ("bucket") and a sensor that allows it to measure the shape of the terrain around it. I would like the robot to dig a trench in a sandbox as per specification, or perhaps level a mound of soil. On the surface, such a problem is familiar to researchers in artificial intelligence. The world is in some initial state and must be modified to some other state, through the use of a reasonably competent agent. To be considered successful, the agent must independently cause the transition for a sufficiently large set of initial and goal states. There are two main difficulties with operating in this particular domain. First, the dynamics of the interaction between a robot earthmover and the world are very complex. This means that it is difficult to produce accurate forward models of actions that the robot might choose. Second, since soil is diffuse, the number of variables required to uniquely specify the state of the world is extremely large. In other words, the configuration space that the robot operates in is essentially infinite.

I have formulated the task of planning earthmoving operations as a problem of constraint optimization. The nature of the models in this domain dictates the use of search. Further, since the large configuration space affects the choice of the control parameters, there is no hope of doing the search off-line.

Several researchers have adopted a similar view to planning for diverse tasks such as robot juggling and robot pool playing (Jordan 90, Moore 92, Schaal 94). Feiten and Bauer have devised a planner for a mobile robot operating in a cluttered environments that plans in the space of actions rather than in the configuration space of the vehicle (Feiten 94). Kelly describes a methodology for navigation of a cross country robot based on like principles (Kelly 95). My approach is also similar in motivation to work in classical optimal control (Kirk 70) in that it seeks to determine the control signals (plans) that will both satisfy some constraints as well as optimize some performance criterion.

Below I discuss five principles developed from my experience with a planner for earthmoving.

- *Develop a tractable representation.* Since the configuration space for this domain is intractable, another framework is necessary. Search is conducted in the space of control parameters (action space) that abstracts the prototypical task that the robot must perform. It is also necessary to develop the notion of an atomic action and a forward model that predicts the effect of an action on the world. Finally, we need an evaluation function to measures the utility of an action.

- *Reduce the search space.* On-line search dictates compactness of the search space. In some cases an analysis of the mechanics will often show that not all the parameters are independent. In other cases, if it is found that the utility function is monotonic in relation to one of the parameters, then it needn't be searched for.

- *Use force as well as geometry to constrain planning.* Geometry of the world and mechanism provides useful constraints on the action space. However, force constraints that arise from the mechanics of robot-world interaction can effectively represent the bounds on what the robot can actually do in a world where friction and torques are significant. No a priori models may be available in which case it is necessary to learn these models from experience.
- *Use multi-resolution search and heuristics.* Abstraction of the search into levels of varying granularity makes the search more efficient. In some domains the addition of common sense heuristics, can make the progression of the plans much more natural.
- *Take plans generated with a grain of salt.* If meaningful models of uncertainty can not be incorporated into the planning, the plans generated through the kind of process described above should be thought of a means of being proactive. The plans are best handed to lower levels controls that are tightly coupled to the world with feedback.

Developing A Tractable Representation

To develop a planner for a task such as digging a trench, we need three things: a search space, a forward model and an evaluation function.

The Search Space

Instead of abstracting robot and world state, we could abstract the task that the robot is to perform such that atomic actions are described by a compact set of parameters. The space spanned by these parameters is called an action space. At every step, the robot selects from the set of feasible actions, one that optimizes some cost criterion. The chosen plan is guaranteed to satisfy constraints (e.g. avoid collisions) as well as optimize a cost criterion (e.g. minimize joint torques). Fig. 1 provides a schematic view of this optimization.

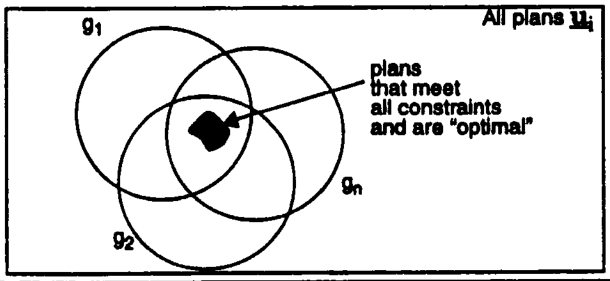

Fig. 1 A schematic view of constraint optimization. I

More formally, an action space is spanned by the range of parameters used to define an action that a robot is capable of executing. Each point in this space represents an atomic action. The space can be separated into two sets— the set of all feasible actions and the set of actions known to fail. An action might fail because it is impossible to achieve or it results in an undesirable effect. The task of an action space planner, then, is a familiar problem in optimal control:

Maximize/Minimize $h(\mathbf{u}_i)$ subject to $g(\mathbf{u}_i)$

where $h()$ is a utility function and $g()$ are constraints that delimit the set of feasible actions and $\mathbf{u}_i$ spans the set of actions that a robot can execute.

Note that the constraints change whenever the state of the world changes. Since we cannot predict the state of the world in response to robot actions and there is a large branching factor (many actions may be chosen at any step), it is not practically feasible to consider sequences of digging actions. Hence the searching will involve identification of only the action that is optimal over a single planning step.

An atomic action

Since an action space encodes actions, not mechanisms, our task representation will not include any details about the configuration of the mechanism. Instead, we will encode the trajectory followed by the excavator bucket (Fig. 2)

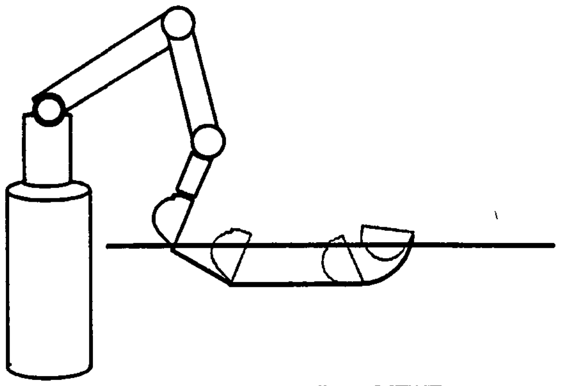

Fig. 2 Manipulator arm equipped with a bucket, creating a trench.

There are an infinite number of digging trajectories that could be used so it is necessary to consider a smaller set—those that are of a particular form. For example, we could base the notion of an atomic action on the patterns used by human operators. There are typically three phases to an operation— penetrate, drag and curl. Trajectories of this form can be parameterized by a smaller set (Fig. 3).

This representation requires six variables and a functional to represent uniquely: k is the distance from a fixed reference frame to the point where the bucket enters in the soil, α is the angle at which the bucket enters the soil. It travels along this angle for a distance d_1, and then follows a horizontal path for a distance d_2. Finally the bucket rotates through the soil along an

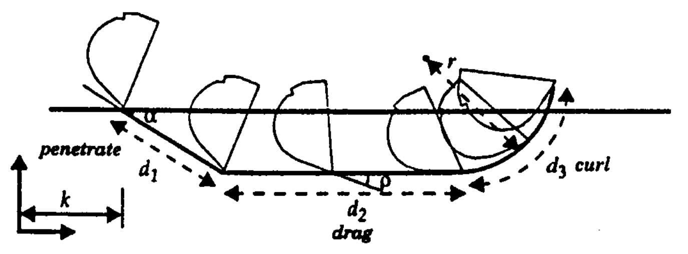

Fig. 3 A typical dig during a trenching operation. A digging trajectory of this form can be represented by (k, α, d_1, d_2, d_3, r, $\rho()$).

arc of length d_3 and radius r. We will also need to determine the value of $\rho()$ (the pitch angle of the bucket, relative to a fixed coordinate frame) throughout the dig.

The Forward Model

One interesting aspect of this problem is that no computationally tractable forward models are available. That is, given a shape of the terrain, and a candidate action, there is no hope of predicting the resultant terrain. Finite element models are available to perform such simulations but since a single run might take tens of minutes to simulate, it is not possible to use such models for the purposes of planning. We might satisfy ourselves with something less. For reasonable plans it is possible to approximate the resulting trajectories. Under this assumption, it is possible to calculate the forces experienced at the cutting edge (using a pseudo-static analysis) as well as approximate the amount soil that might be swept into the bucket.

The Evaluation Function

Typically, for earthmoving operations it is essential to optimize the volume of soil excavated. Since there is often more than one action that will meet this requirement, it is possible to add other criteria such as minimizing the time taken to complete the dig, or minimizing the torques on the joints of robot. Note that the evaluation is good for only single actions and must be recomputed across the range of the action space whenever the state of the world changes. I assume that the state of the world (topology of the terrain) can be measured using a sensor.

Reducing the Search Space

Taken naively, the search space as described above is too large to be searched on-line. However, some analysis of the space can make the search more tractable. For example, if we assume that the amount of soil excavated increases monotonically with d_2, its value can be found easily if the other variables are instantiated. Values of $\rho()$, r, d_3 can be found from a mechanical analysis of contact between the tool and the terrain (Singh 95a).

The result is that our action space can be spanned by a compact three-tuple (k, α, d_1).

Using Force & Geometry to Constrain Planning

Recall that each point in an action space represents a unique action. We can pose constraints in the regions of the space that correspond to plans that the planner should stay away from.

Geometric Constraints

A plan may be *geometrically* infeasible because it requires a robot to exceed its range of motions or because it violates some geometric criterion associated with successful execution of the task.

The reachability constraint. Fig. 4 shows a graphical depiction of the set of digging actions that are within the workspace of the robot for the very first dig. The surface that is formed by this three-dimensional set represents critical points that are on the edge of being reachable. Since the actions are parameterized with a larger set of variables than can be visualized in three dimensions, some of the critically reachable plans are in the interior of the set.

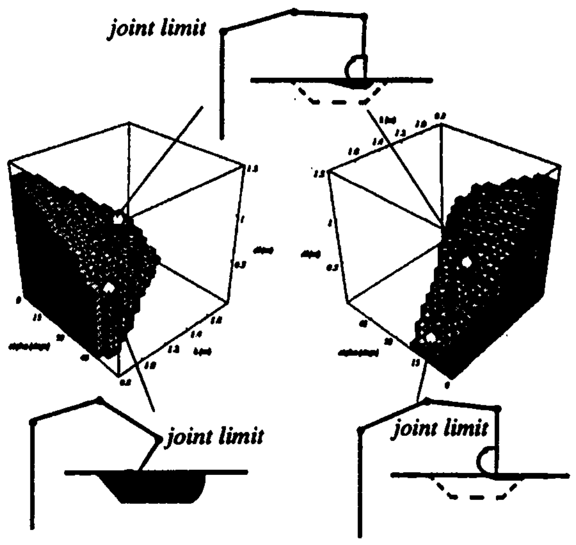

Fig. 4 Two views of set of feasible plans that meet the reachability constraint. This set is specific to the joint limits of the robot used and to the terrain before the action is commenced. The trajectories implied by three points are shown explicitly. The shaded region shows the amount of soil swept by each trajectory.

The shaping constraint. The shaping constraint for trenching keeps the trajectory of the bucket from going past the shape of the desired trench (dashed line in Fig. 5). That is, all trajectories that extrude past these boundaries are excluded.

The volume constraint. Of the geometrically feasible actions, some will yield a partially filled bucket, some will result in a full bucket and yet others will sweep through the terrain to excavate more soil that can possibly be held in the bucket. A simple calculation of swept volume is used to predict the amount of soil excavated by a digging action. All actions that excavate a volume larger than a preset percentage of the bucket

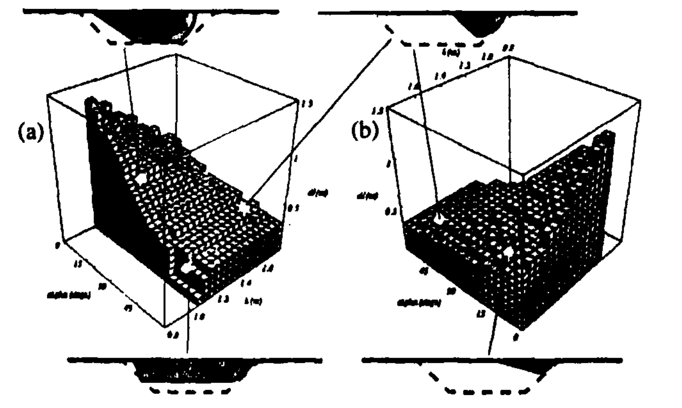
Fig. 5 Two views of set of feasible plans that meet the shaping constraint. Note that all of the digs here stay within the bounds of the trench that has been specified (dashed line)

capacity, are excluded (Fig. 6).

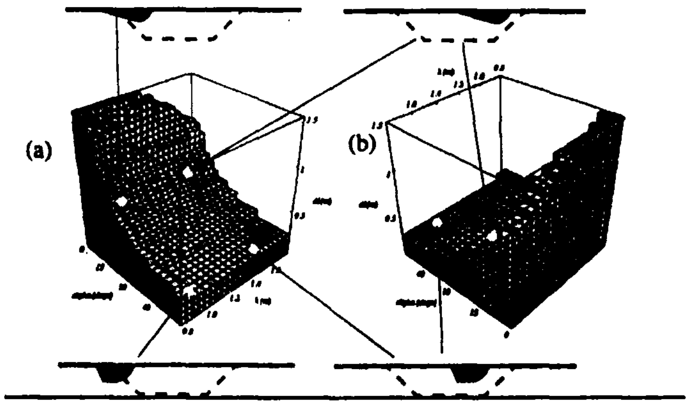
Fig. 6 Two views of set of feasible plans that meet the volume constraint. All trajectories sweep only sweep up to as much soil that the bucket can hold

The set of actions that meet all the geometric constraints is shown in Fig. 7.

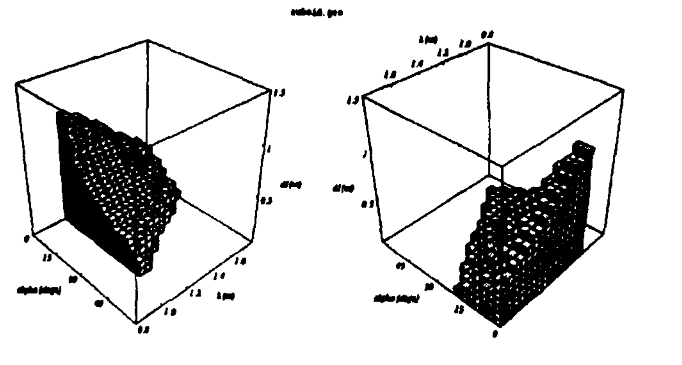
Fig. 7 Two views of set of feasible plans that meet the shaping constraint.

Force Constraints

If a robot earthmover is infinitely strong, that is, it can muster any torque required, then it is sufficient to consider only geometric constraints. More realistically, for robots with torque limits, it is necessary to consider the required forces. If we could estimate the forces required for candidate digs, we would have a good criterion by which to further restrict the set of digs that are geometrically feasible. Unfortunately, the interaction between an excavating tool and terrain is complex enough that no simple physics-based models are available to predict the resistive forces for arbitrary actions and terrain shapes. In earlier work I have shown a method that learns to predict resistive forces based on observation of the resistive forces, tool trajectories and the shape of the terrain from previously conducted digging experiments (Singh 95b). Fig. 8 compares the geometric and force constraints for the very first digging action— for some values of the parameters, the robot is not limited geometrically but the corresponding actions can not be achieved because the required forces are too great.

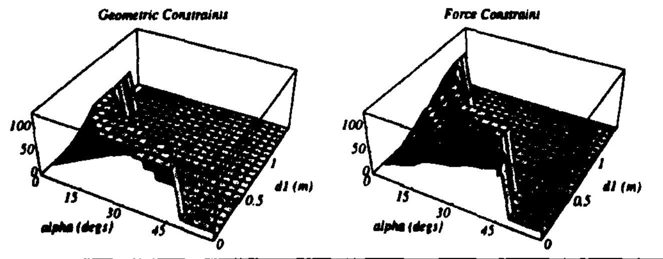

Fig. 8 Comparison of geometric constraints vs. force constraints for one value of *k*.

To determine if a candidate dig passes the force constraint, the force prediction function is called. The predicted resistive force and the robot's trajectory are used to calculate the effective joint torques required to overcome the resistive forces. A candidate dig is force-feasible if the joint torques due to resistive forces do not exceed the torque capability of the robot.

Using Multi-resolution Search and Heuristics

So far I have examined only a planar problem, that of digging a trench as wide as the excavator bucket. For more complex problems such as the excavation of a foundation footing (Fig. 9), we could abstract the problem such that at a coarser level the ultimate goal is broken down into a sequence of trenches. The alternative is to make the planner global by searching for additional variables that dictate base position. This has two problems. First, the search space becomes very large and second, it is difficult to develop an efficient sequence of actions.

Sometimes, there are common sense procedures that can be used at a coarser level of planning. For example, when digging into a cavity we might always start near the top, before digging further down into the hole for the same reason as one starts

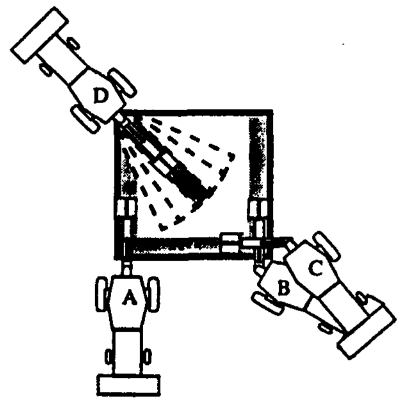

Fig. 9 Plan view of an excavator backhoe digging a footing. The machine starts at A and digs one side of the footing to a desired depth. Second, the machine moves to B from which a second side can be excavated. A third setup (C) is achieved by rotation of the base. Excavation of the center is achieved by a fan-like pattern of digs from (D).

sweeping stairs from the top and not the bottom. If relevant, such procedures can be incorporated into the coarse planning alleviating the need for search to generate such sequencing.

Taking Plans Generated With a Grain of Salt

For real world applications, it is never possible to model all the effects of sensors and actuators. At best we might hope to capture first order or second order effects. It may be possible to fold uncertainty models into the planning directly and limit the feasible set by the margin of cumulative uncertainty. However, this is not always possible and after all the modeling, plans are best thought of as a means of proaction. Final execution of the plan is best left to a lower level control scheme that has a tight coupling with the world. Control schemes typically are applicable in a narrow range of operating conditions and in this light planning can be thought of as a means to specify actions that fall within the operating conditions that where properties of convergence hold.

We would like as much feedback as is possible— state feedback before planning so as to plan properly, during execution so that unmodeled effects can be compensated for, and after execution so as to set the stage for the next plan.

Implementation

We have developed a testbed to conduct experiments in subsurface sensing and excavation. The testbed consists of a sandbox (2.5m x 2.5m x 1m), a hydraulic robot outfitted with an excavator bucket and force sensor, and a laser range finder. The setup of our testbed is shown in Fig. 10. We use a large industrial manipulator with an end effector payload of approximately 125 lbs. A small excavator bucket with a volume of $0.01m^3$ serves as an end effector. The laser range scanner pro-

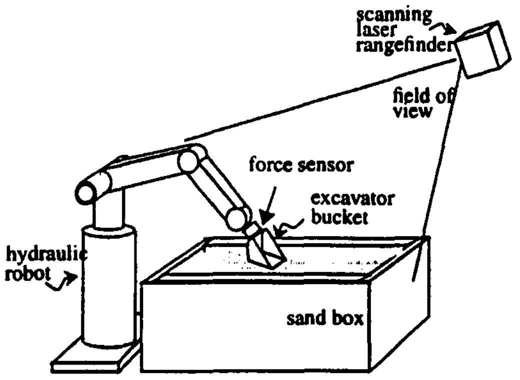

Fig. 10 Testbed

duces an image of the terrain such that the value of each pixel in the image corresponds to the distance from the scanner to the world. This image is used to build a topographical map of the terrain in the sandbox.

The excavation cycle consists of three phases. First, a terrain map is produced from range images taken by the laser scanner. Next, the planner chooses a digging action to execute that satisfies geometric and force constraints and optimizes utility criteria. Last, the action is executed by the robot and the cycle repeats until the terrain is sufficiently close to the specified goal.

When the search space is extremely large and answers are needed quickly, at the expense of optimality, a numerical method such as simulated annealing can be used. In fact I have used an augmented version of simulated annealing to do the optimization in the past (Singh 91). However, the stochastic nature of the search means that the solutions found are not repeatable and it is very hard to perform controlled tests where one of the constraints is modified. Further, exhaustive search (test and generate) with intelligent ordering of constraints works well enough for our purposes. While in the worst case this means that each constraint has to be evaluated for each candidate point in the action space, in reality testing early for conditions such as reachability prunes the search space effectively. It is possible to further select from this set based on other criteria. such as time or joint torques required. For example, Fig. 11 shows two different trajectories selected for the same initial state of the terrain, given different evaluation functions. Note that both plans sweep the same amount of soil.

Experimental Results

We have conducted over 2000 digging experiments with our testbed. Fig. 12 shows a progression of one sequence in which the goal is to produce a trench. In this experiment, the digging actions chosen sweep a volume of soil very close to the bucket capacity. However, the actual yield in the bucket is a function of how cohesive the soil is—digging in loose, granular soil results is some of the soil spilling out of the bucket. Experimen-

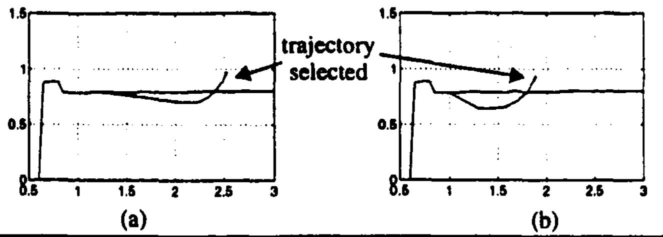

Fig. 11 Two different trajectories selected for the same state of the terrain (a) evaluation function minimizes the largest torques (b) evaluation function minimizes the total joint torques.

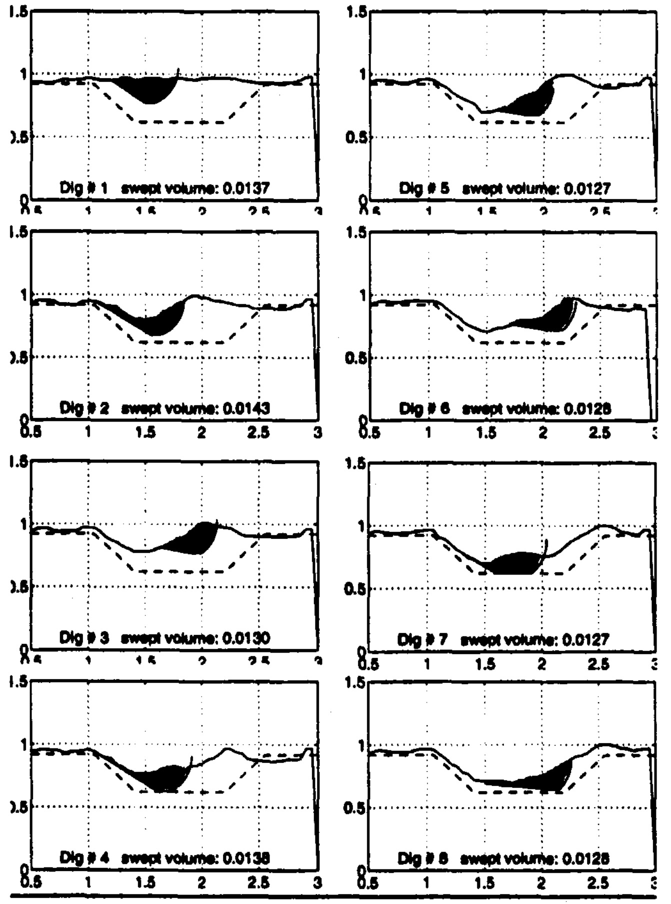

Fig. 12 The first eight steps in the creation of a trench. The desired trench is shown by the dashed line. The shaded region is the volume of soil (in m^3) estimated to be swept by the excavator bucket.

tal results are encouraging. The planner examines approximately 2000 plans in a few seconds and almost always find one that fills the bucket. The planner becomes inefficient as the profile of the terrain approaches the desired goal. This is because the form of the prototypical plans does not produce efficient trajectories. In this case, it is possible to execute open loop trajectories that follow the outline to the desired trench to scoop out the remaining soil.

Conclusions

Earthmoving is an example of an application where the interaction between a robot and the world is significant. To perform such a task efficiently, we will want to use every trick in the book. This includes the use of good forward models, mechanical analysis and multi-resolution search.

I have shown how the task of digging a trench can be stated as a problem of constraint optimization. Since the optimization happens one action at a time (necessary due to the uncertain forward models and large branching factor at every planning step), the planner is greedy and thus requires a higher level planner that ensures that proper subgoals are specified.

References

(Feiten 94) Feiten, W. and Bauer, R., "Robust Obstacle Avoidance in Unknown & Cramped Environments," In *Proc. IEEE International. Conference on Robotics and Automation*, May 1994. San Diego.

(Jordan 90) Jordan, M. I, and Jacobs, R. A, "Learning to Control an Unstable System with Forward Modeling," In *Advances in Neural Information Sciences* 2, Morgan Kaufmann, 1990.

(Kelly 95) Kelly, A. J., "An Intelligent Predictive Control Approach to the High-Speed, Cross Country Autonomous Navigation Problem", Ph. D. thesis, Robotics Institute, Carnegie Mellon University, June 1995.

(Kirk 70) Kirk, D. E., *Optimal Control Theory*, Prentice Hall, 1970.

(Moore 92) Moore, A. and Atkeson, C., "An Investigation of Memory-base Function Approximators for learning Control," Technical Report, MIT AI Lab, 1992.

(Schaal 94) Schaal, S., Atkeson C, "Robot Juggling: Implementation of Memory Based Learning," IEEE Control Systems, Vol 14 (1), February,1994.

(Singh 91) Singh, S. "An Operation Space Approach to Robotic Excavation," In Proc. *IEEE Symposium on Intelligent Control*. Alexandria, August, 1991.

(Singh 95a) Singh, S., "Synthesis of Tactical Plans for Robotic Excavation," Ph.D Thesis, Robotics Institute, Carnegie Mellon Univ, January 1995.

(Singh 95b) Singh, S., "Learning to Predict Resistive Forces During Robotic Excavation," In *Proc. International Conference on Robotics and Automation*, Nagoya, May 1995.