Robot Planning in the Space of Feasible Actions: Two Examples

Sanjiv Singh and Alonzo Kelly

Field Robotics Center
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Several researchers in robotics and artificial intelligence have found that the commonly used method of planning in a state (configuration) space is intractable in certain domains. This may be because the C-space has very high dimensionality, the “C-space obstacles” are too difficult to compute, or, because a mapping between desired states and actions is not straightforward. Instead of using an inverse model that relates a desired state to an action to be executed by a robot, we have used a methodology that selects between the feasible actions that a robot might execute, in effect, circumventing many of the problems faced by configuration space planners. In this paper we discuss the implications of such a method and present two examples of working systems that employ this methodology. One system drives an autonomous cross-country vehicle while the other controls a robotic excavator performing a trenching operation.

1 Introduction

Commonly, robot planning has used an abstraction known as *configuration space* (C-space). The main idea is that the configuration of a robot and the objects that it manipulates can be represented as points in the space spanned by a set of parameters that uniquely describe the state of the robot and the world. Simplistically speaking, the job of a planner is to find a path in this space from an initial state to the goal state. The path in C-space prescribes a sequence of states that must be achieved to accomplish the goal and as a consequence dictates the actions that the robot must execute. There are, however, several cases in which configuration-space methods do not work well. In this paper we examine a different approach to posing these problems using the examples of autonomous cross-country navigation and robotic excavation.

The duality between actions and states has been well discussed in the AI literature [4]. In a state based representation, the world is viewed as a series of states that are altered by events. Events are modeled only in terms of their state-changing function. Alternatively, in an action based approach, the state of the world at a particular moment in time is a function of the set of events that have occurred up to that moment. As with most dualities, the choice of one representation over another does not affect any essential capability for expression since one dual representation can be converted into the other. However, the form of a representation can make certain kinds of properties more natural to express and reason about [3]. State based methods require a planner to find a mapping relating states, while action based methods require a mapping between actions and states. We will see that in some cases, it is easier to express the latter.

This paper suggests a methodology for robot planning using an abstraction known as *action* space. The process starts by encoding a task as opposed to the mechanism used to perform the task. At every step, the robot selects from the set of feasible actions available to it, one that optimizes some cost criterion. We show how the tasks of autonomous cross-country navigation and robotic excavation are posed as problems of constrained optimization. While at first sight these systems seem to have little in common, they both succeed in large part due to a formulation that makes the problem tractable.

2 Relation to Other Work

As mentioned above, several researchers have written about the duality between actions (events) and states. In fact, AI researchers now routinely use planners that search in a space of plans rather that in the space of states [4]. In the robotics literature, several researchers have adopted a similar view to planning for diverse tasks such as robot juggling and robot pool playing [6][10][11]. Similarly 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 state (configuration) space of the vehicle [2]. The proposed approach is similar in motivation to work in classical optimal control [8] in that it seeks to determine the control signals (plans) that will both satisfy some constraints as well as optimize some performance criterion. However, this paper is less about the methods of constrained optimization and more about the methodology from which the task representation is derived.

3 Action Spaces

In this section we describe some of the problems with planning in C-space. We propose an abstraction called an *action* space and discuss the implications.

3.1 Robot Planning Using C-space methods

Planning in C-space is a powerful paradigm. There are, however, at least three cases in which C-space methods do not work well:

- If C-space obstacles are complex, it may not be possible to build them quickly in response to sensor data.
- If the inverse model that prescribes action given a desired state is either not well defined or too difficult to compute, as in the case of non-holonomic systems, a C-space representation is ineffective.
- If a planning problem has very high dimensionality it is computationally intractable to represent the problem in C-space.

3.2 Action Space methods

Consider a different way of posing planning. Instead of abstracting robot and world state, the task that the robot is to perform is abstracted such that actions that the robots must perform can be described by a compact set of parameters. An action space planner identifies the set of feasible actions (a subset of the task parameters) and chooses one that is optimal. The chosen plan is guaranteed to satisfy constraints (e.g. avoid collision with obstacles) as well as optimize a cost criterion (e.g. minimize joint torques). What makes such a method distinct from C-space methods is that instead of inferring actions from states, it operates directly on the set of robot actions. Simplicity is the main advantage. Instead of building complicated C-space obstacles and then deducing a path in between the obstacles, a planner chooses from among the set of actions available to it, the one that best accomplishes the robot's goal.

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 $h(\underline{\mathbf{u}}_i)$ *subject to* $g(\underline{\mathbf{u}}_i)$

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

We propose that planning in certain domains is more tractable if the task is represented in an action space. The method allows for generality since an action space allows abstraction from the robot mechanism. The mechanism is represented via the constraints. For two different robots performing the same task, only the function that determines feasibility due to the mechanism constraints need change. The downside of planning in an action space is that most solutions are *local*. That is, in practice, a higher level planner must specify intermediate goals.

4 Cross-Country Navigation

Consider the task of path planning for an autonomous vehicle travelling cross country over rough terrain at high speeds [1][7]. In general, the vehicle must achieve a useful goal while avoiding collision. The goal of the vehicle may be to move from its initial position to some other distant position, to map an entire area, or to search an area for objects of interest. It may also be useful to optimize fuel consumption, or distance travelled. In realistic terrain, the vehicle is challenged by regions that would cause tipover, trapped wheels, or loss of traction. Some regions are not traversable at all and others may cause disastrous system failures such as falling into an abyss.

An action space approach is attractive for the purposes of cross-country navigation for several reasons. First, a conventional automobile is underactuated (non-holonomic), so the mapping from C-space to action space is under-determined. It is not possible, in general, to compute the speed and steering commands which will cause a vehicle to follow an arbitrary C-space curve. The use of action space avoids entirely the problem of path generation for non-holonomic vehicles. Second, many constraints on vehicle motion are differential in nature and most naturally expressed in action space. In a purely geometric (kinematic) world, path curvature can be changed instantaneously. However, a vehicle traveling at even moderate speeds is constrained in the range of its steering actions. Thus, it is necessary to consider actuator response and the natural space in which to model actuators is action space because it permits expression of these constraints in terms of dynamic models. Third, action space planning is more computationally efficient. Continuous high speed motion implies that a vehicle has very little time to react to what it sees. A planner must decide what to do under stringent timing constraints or the entire system will fail. An action space formulation permits a fast, if coarse-grained, evaluation of all feasible alternatives.

The method starts with enumeration of alternatives expressed in action space (a discretized set of feasible actions that the vehicle might execute). Next, the corresponding Cartesian trajectories are computed using a forward model of the vehicle. No attempt is made to represent obstacles in C-space. Candidate vehicle trajectories are evaluated in order to assess vehicle safety. Finally, the planner integrates the safety assessments with its strategic goal to decide on the commands to be sent to the vehicle.

Below we present a forward model of our autonomous vehicle, show how this model is used to determine the trajectory that corresponds to every action space alternative, and finally discuss how, at every control cycle, a command is chosen that both satisfies the constraints and optimizes a cost function.

4.1 Representing Navigation in an Action Space

Our action space for this task is spanned by the variables of speed and path curvature. For conventional vehicles, these variables map directly to the controls of throttle and steering but even for other types of vehicles (such as skidsteered), these variables provide an intuitively understandable representation. This section shows how cross country navigation is stated as a problem of constrained optimization in the space spanned by the task parameters.

In this action space the constraints that separate the set of feasible actions from the infeasible are due to steering limits, acceleration limits and collision of the vehicle with objects in the world. Consider the vehicle in Fig. 1. Of the

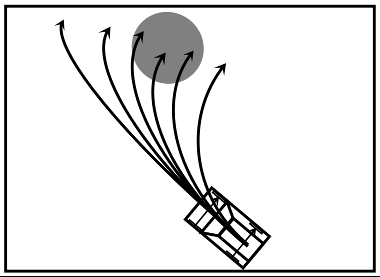

Fig. 1 Some of the trajectories that the vehicle might choose in the next control cycle will lead to collision with an object in the world.

throttle and steering commands that the vehicle might attempt at the next instant, some are not available because they exceed the turning radii possible (Fig. 2 a), exceed the lateral acceleration limits (Fig. 2 b), or, cause collision with an obstacle (Fig. 2 c). Additionally, in rough terrain some actions might cause tipover or collision of the undercarriage.

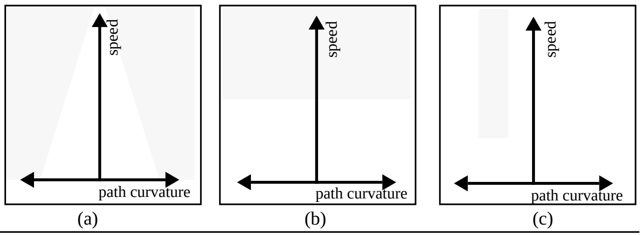

Fig. 2 Three of the constraints in the action space spanned by path curvature and speed. (a) steering constraint (b) acceleration constraint (c) collision constraint. Shaded areas indicate infeasible regions.

Of the feasible set of actions, it is necessary to select one that optimizes a cost criterion. A simple utility function is to proximity to the goal. Maximizing this utility function implies that the command that drives the vehicle closest to the goal is preferred. Additionally we may consider the hazards that a vehicle might encounter in terms of a continuous space that encodes a degree of danger. That is, in addition to the hard constraints described above, those commands that minimize the degree of hazard to the vehicle are preferred. However, hazard avoidance is considered a constraint in the current formulation while goal-seeking performance is optimized.

4.1.1 Forward model

At high speeds and over uneven terrain it is essential to explicitly consider vehicle dynamics since the actual response varies significantly from the kinematic idealization. Our forward model produces an estimate of the trajectories resulting from vehicle commands allowing the planner to conducts its search over the space of feasible commands $\underline{u}(t) \in U$. We have modeled our vehicle as a multivariable state space system (Fig. 3).

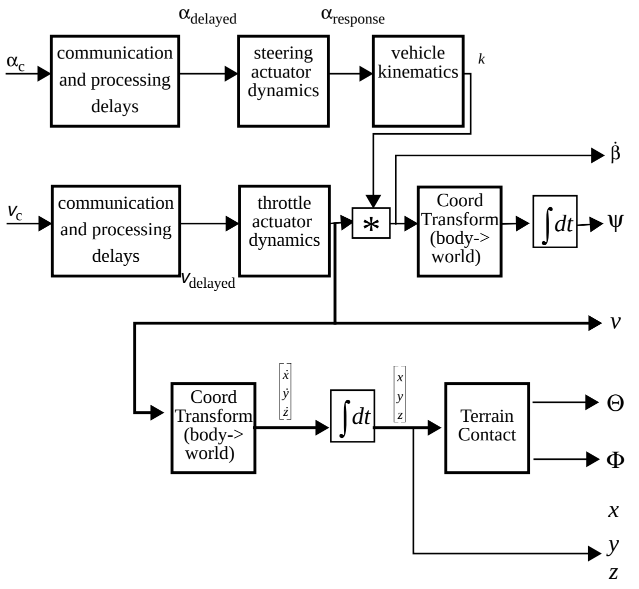
$\alpha_{delayed}$
$\alpha_{response}$
α_c
communication and processing delays
steering actuator dynamics
vehicle kinematics
k
$\dot{\beta}$
v_c
communication and processing delays
throttle actuator dynamics
$*$
Coord Transform (body-> world)
$\int dt$
Ψ
$v_{delayed}$
v
Coord Transform (body-> world)
$\dot{x}$ $\dot{y}$ $\dot{z}$
$\int dt$
x y z
Terrain Contact
Θ
Φ
x
y
z

Fig. 3 Forward model used to predict vehicle state given actuator commands (α_c, v_c). The output state contains rate of heading ($\dot{\beta}$), vehicle velocity(v), heading(ψ), pitch(θ), roll(ϕ) and position (x, y, z).

The inputs to the model are the steering angle, α_c, (corresponding to the desired path curvature), and throttle, v_c, (corresponding to desired speed) and an elevation map of the terrain ahead of the vehicle. The commands are first delayed to account for communications and processing and passed through a model of the actuator dynamics. In the case of the throttle (speed) the influence the gravitational load is so significant that it must be modelled. The predicted steer angle response is passed through a model of the steering column to predict the actual curvature (k) of the path traversed. The product of curvature and speed provides an estimate of angular velocity. The velocity is converted to world coordinates to generate the components of vehicle velocity along the world frame axes and then integrated to provide position. Pitch and roll are determined by placing the vehicle wheels over the terrain map and allowing the vehicle to settle onto the map to determine pitch and roll. Heading is computed by integrating the angular velocity after converting coordinates to the world frame.

Note that with the use of a forward model, computed trajectories *meet the mobility constraints of the vehicle by* *construction*, so there is never any question if an action (steering/throttle pair) can be executed by the vehicle.

4.2 Optimization

Apart from satisfaction of thresholds on the degree of predicted hazard, the system ranks actions based on proximity of the resultant trajectory to the ultimate (strategic) goal. At times, goal-seeking may cause collision with obstacles. The system incorporates an arbiter which permits obstacle avoidance and goal-seeking to coexist and to simultaneously influence the behavior of the vehicle. Arbitration between obstacle avoidance and goal seeking is accomplished by forming a vote vector for each action and choosing the action which is closest to the strategic vote maximum while not violating the obstacle (hazard) avoidance constraint. Thus, the strategic goal is used to bias obstacle avoidance when there are a number of alternatives, and obstacle avoidance wrests absolute control from the strategic goal seeker when it is necessary to do so.

4.3 Implementation

Several times a second, the planner considers approximately ten steering angles to use during the next control cycle. The forward model simulates the effect of using these steering angles over a short period of time and evaluates each of the resultant paths. Any steering angles that result in paths that go near or through hazardous vehicle configurations are discarded. The steering angle that results in a path that is optimal based on several criteria is chosen (Fig. 4). The choice is necessarily based on predicting the state of the robot a few seconds later. The search space is too large to consider sequences of actions over a longer period because of the combinatorics involved. In Fig. 4 the system issues a left turn command to avoid a hill to its right. The histograms represent the votes for each candidate trajectory (higher values indicate safer trajectories). The hazards are excessive roll, excessive pitch, collision with the body, and collision with the wheels. The tactical vote is the overall vote of hazard avoidance. The strategic vote is the goal seeking vote.

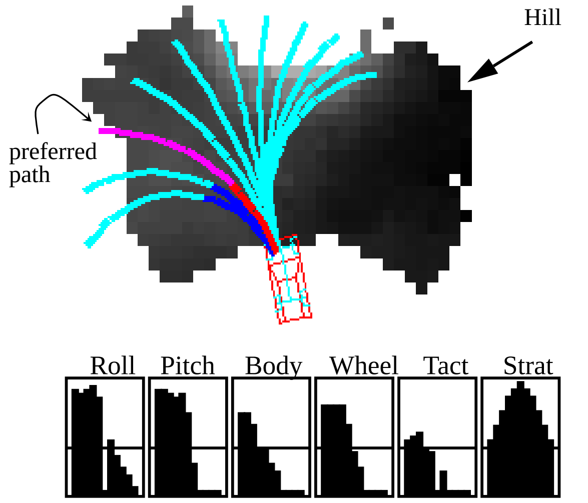

Fig. 4 The system chooses a steering angle from a set of candidate trajectories.The histograms below represent the votes for each candidate trajectory (higher values indicate preferred trajectories). The tactical vote is the overall vote of hazard avoidance. The strategic vote is highest for the action that is most directed towards the goal.

4.4 Results

Our navigation planner (RANGER) is being used on an autonomous vehicle shown in Fig. 5. We have used laser range data and stereo range data to build maps of the terrain over which the vehicle must travel. In the former case, excursions of 15 kilometers and instantaneous speeds of 15 km/hr have been achieved while tracking a coarsely specified path. Average speed was on the order of 7 km/hr.

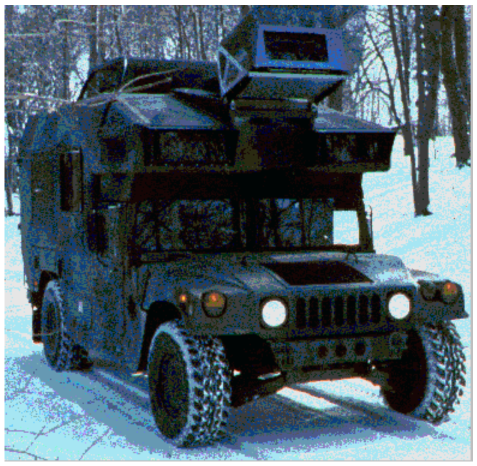

Fig. 5 Navigation testbed— a modified military HMMWV.

RANGER has also been integrated with a stereo vision system at the Jet Propulsion Laboratory[9] on another vehicle. Fig. 6 shows a short autonomous excursion along a dirt road bounded by trees and bushes on the right and a ravine on the left. The sequence of images to the left are the stereo range images. To the right are intensity images of the scene corresponding to the range images. The images are positioned in correspondence with their associated position in the terrain map. The terrain map, drawn in the center, is rendered with intensity proportional to elevation. The path followed is drawn leading to the position of the vehicle near the end of the run. The run terminates at the end of the road. Two distinct obstacle avoidance maneuvers occur. The first is a left turn to avoid a large tree and the second is a recovery right turn to prevent falling into the ravine.The system drives this road routinely at this point in its development.

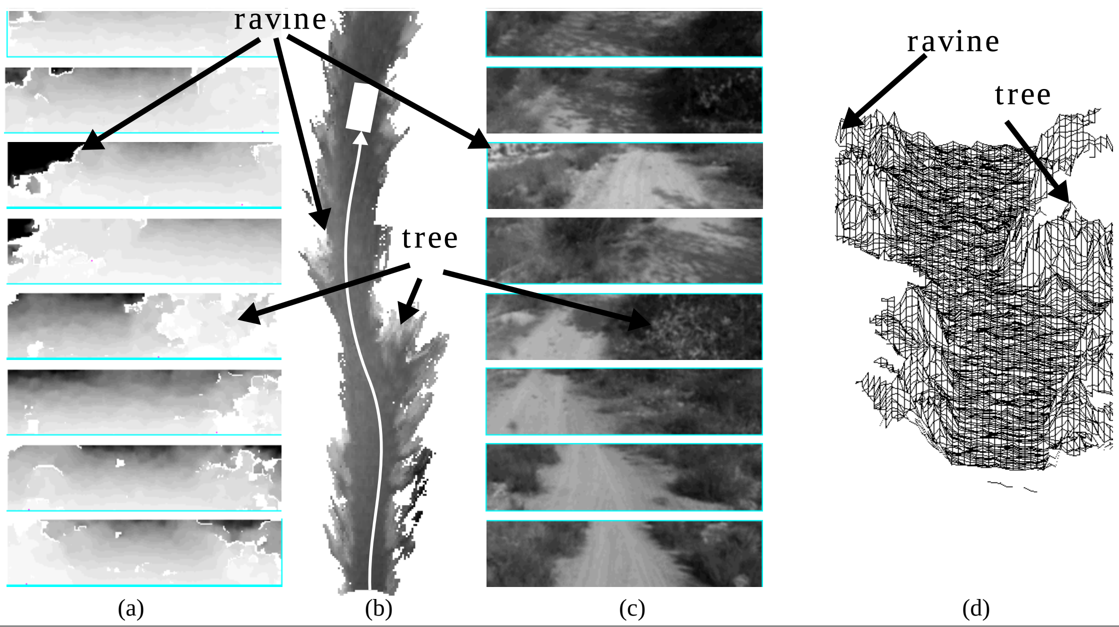

Fig. 6A short cross country excursion. (a) shows a sequence of range images from a stereo vision system mounted on the vehicle. (c) shows a sequence of reflectance images from one of the cameras. (b) and (d) show an overhead view of an elevation map that was generated.

5 Excavation

The task of robotic excavation can be stated in familiar terms. The world is in some initial state and must be transformed into some other state. For example, we would like a robot to dig a trench of specified size, or to level a pile of soil. This domain is distinguished in two important ways. First, since soil is not rigid, a C-space representation of natural terrain has very high dimensionality. Second, the inverse model, the mapping from a desired state to the next action is not straightforward. Unfortunately, a full forward model of the type presented in 4.1.1 is not readily available because of the complexity of the interaction between a tool and terrain. In most cases, however, it is possible to approximate the utility of an action. For example it is possible to estimate the amount of soil that would be excavated by a particular action and the resistive joint torques that would be experienced during the excavation.

This section presents a compact set of task parameters for the task of trenching. Parameterization of this task is not as simple as in the case of the autonomous navigation because a mapping between the controls of an excavator and task variables is not obvious. Hence, we have modeled the patterns used by human operators. In this action space we pose geometric and force constraints. Next we show how a single plan is selected for execution.

5.1 Developing a Representation

The first task is to encode a prototypical trenching 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 encode the trajectory followed by the excavator bucket. Fig. 7 shows a robot manipulator equipped with a "bucket" performing a trenching operation.

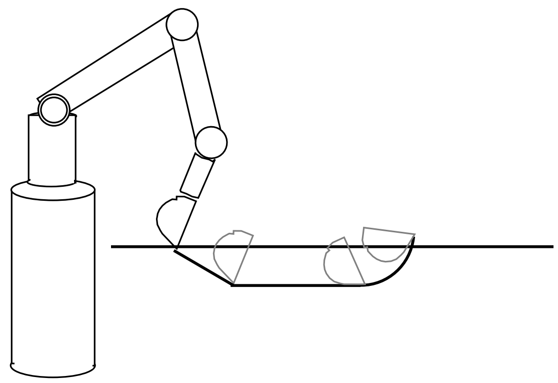

Fig. 7 Manipulator arm equipped with a bucket, creating a trench.

Fig. 8 shows a compact representation of a prototypical trenching action based on an observation of digging patterns used by human operators. This representation

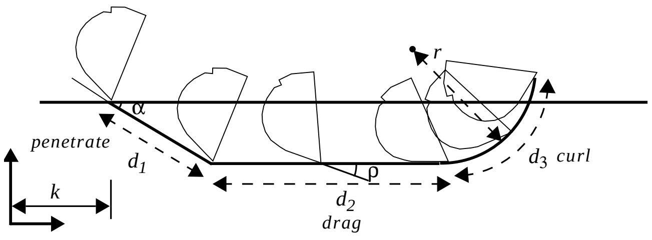

Fig. 8 A typical dig during a trenching operation. There are typically three distinct types of motion— *penetrate, drag and curl.* A digging trajectory of this form can be represented by a seven-tuple (k, α, d_1, d_2, d_3, r, ρ).

requires seven variables 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 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. If we assume that the amount of soil excavated increases monotonically with d_2, d_2 can be found easily if the other variables are instantiated. Values of $\rho()$, r, d_3 can be found from an analysis of contact between the tool and the terrain[12]. Our action space, then, is spanned by a compact three-tuple (k, α, d_1).

5.2 Geometric Constraints

Recall that each point in an action space represents a unique action and that some regions in the action space represent plans that are geometrically infeasible. 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. Starting with flat terrain, the robot is to create a trench of specified size as in Fig. 9. Below we examine the geometric constraints that can be posed on the space spanned by the independent parameters of prototypical trenching actions.

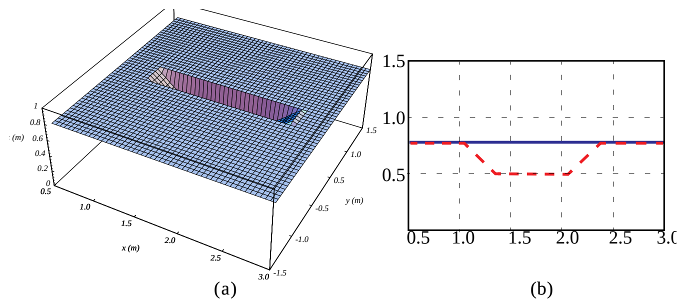

Fig. 9 (a) 3D view of the desired shape of the terrain. (b) 2 D section along trench (dashed line) to be excavated. The solid line represents the state of the existing terrain.

- **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. 9). 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 capacity, are excluded.
- **The reachability constraint**. A standard inverse kinematic method is used to determine whether or not the trajectory corresponding to a candidate action lies within the workspace of the manipulator.

The set of actions that meet all the geometric constraints is shown in Fig. 10.

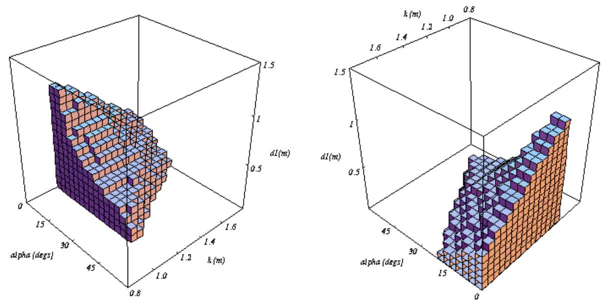

Fig. 10 Two views of set of feasible plans that meet all the geometric constrains for the scenario in Fig. 9

5.3 Force Constraints

If a robot excavator 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 forces required to accomplish digging. 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 a phenomenon that no simple physics-based models are available to predict the resistive forces for a given action and terrain shape. [13] discusses a method that learns to predict resistive forces based on experience. The basic idea is that for a given terrain and tool combination it is possible to build an expectation of the resistive force based on observation of the resistive forces, tool trajectories and the shape of the terrain. Fig. 11 shows the set of actions that satisfy the force constraint given the excavator and terrain in Fig. 7. Approximately 4000 force readings from approximately 100 digging actions were used to build the force model.

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.

5.4 Optimization

Given a mechanism and the shape of the terrain, it is the job of the planner to identify a single plan from the feasible set of plans to be executed. Picking a point in the feasible set guarantees that the plan will work (modulo correctness

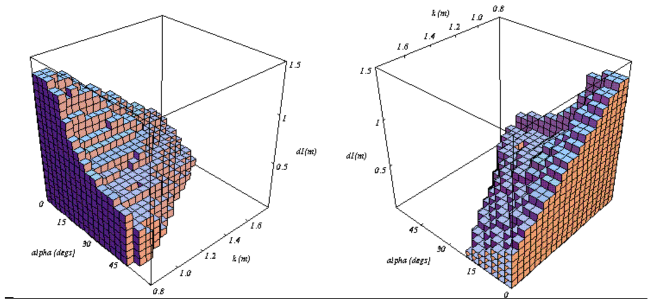

Fig. 11 Two views of set of feasible plans that meet the force constraints given scenario in Fig. 9.

of the model), but not all feasible plans are equivalent in their utility. Other criteria (maximize excavated volume, minimize time/joint torques, etc.) can be used to choose between the set of feasible plans.

Identification of an “optimal” point within the feasible set does not require explicit computation of the constraint surfaces. It is possible to use a numerical method that only requires a boolean function that decides whether a candidate point passes or fails a particular constraint. We currently use an exhaustive search in a discretized action space to identify the set of feasible actions that have a minimum utility. For example, we might stipulate that initially all feasible plans sweep a volume of soil that is at least 90% of the volume of the bucket (Fig. 12). It is possible to further

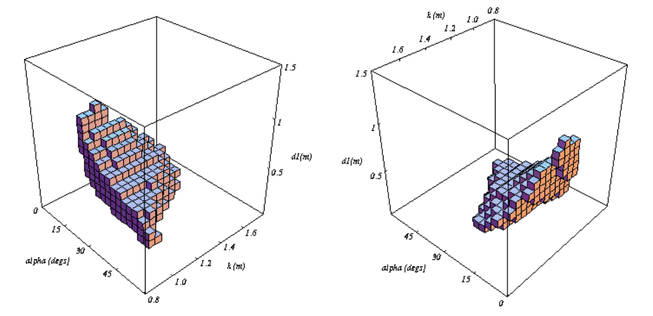

Fig. 12 Two views of set of feasible plans that meet all constraints given scenario in Fig. 9 and also sweep a volume of soil that is greater than 90% of the volume of the bucket.

select from this set based on other criteria. For example, it is possible to sort this set based on the magnitude of the joint torques expected to perform the action.

5.5 Results

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 Cincinnati Milacron T3 hydraulic robot outfitted with an excavator bucket and force sensor, and a laser range finder. The setup of our testbed is shown in Fig. 13. 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 produces an image of the terrain such that the value of each pixel in the image is based on the distance from the scanner to the world along a ray that sweeps in a raster fashion. The scanner has a 60 degree horizontal field of view and a 45 degree vertical field of view. We have adapted a perception and mapping system developed at CMU [5] to produce terrain elevation maps of the terrain in the sandbox. An example terrain map of the testbed is shown in Fig. 14. Each cell in the terrain map is 5 cm square.

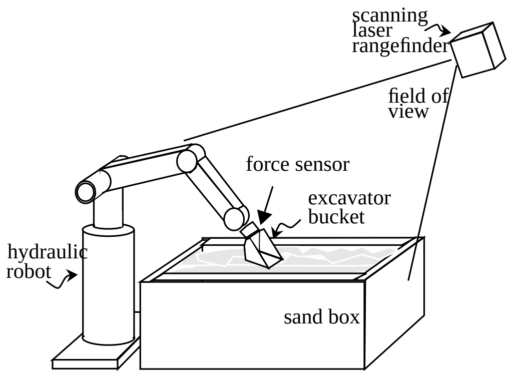

Fig. 13 Testbed

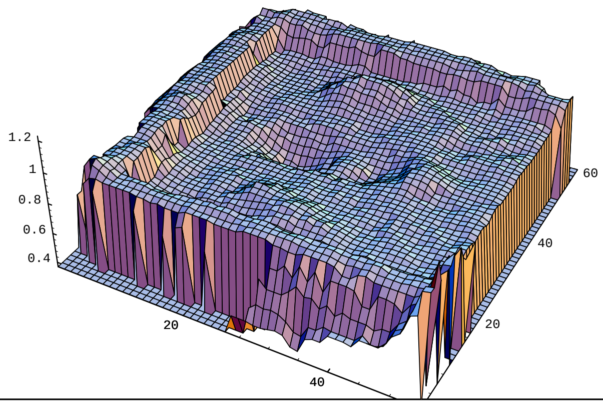

Fig. 14 Elevation map of testbed.

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. Fig. 15 shows a progression of a trench being excavated using this method. 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 leav-

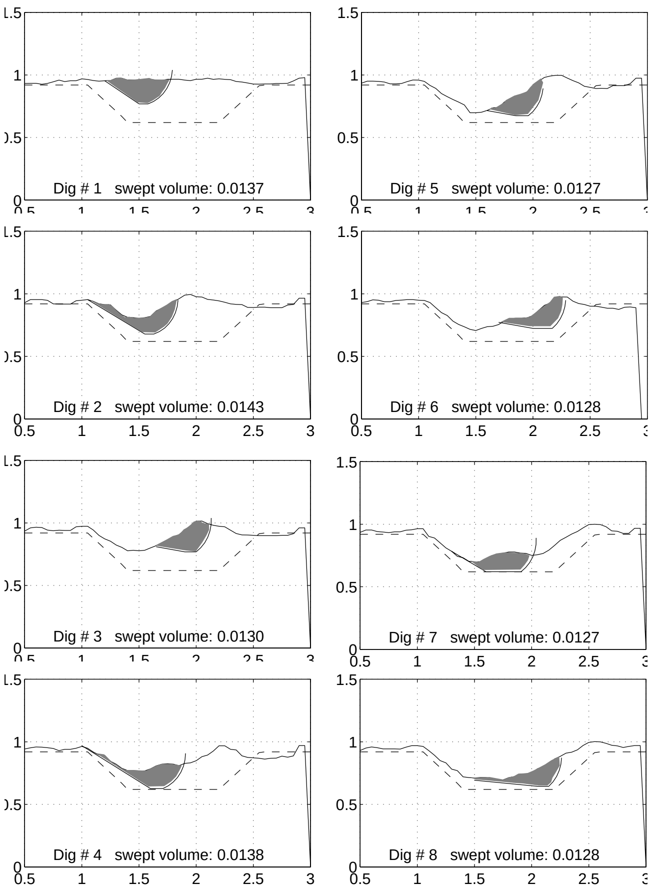

Fig. 15 The first eight steps in the creation of a trench. The desired trench is shown by the dashed line. The shaded region is the volume of soil (in m3) estimated to be swept by the excavator bucket.

ing the bucket. In practice we have found that the volume constraint is best relaxed to a volume approximately twice the bucket capacity. In this case, most digging actions are constrained by reachability, the shape of the desired trench and resistive force.

6 Summary

It has often been suggested that a significant part of solving problems in robotics and artificial intelligence is to find a suitable representation. We have started with the dictum "encode the task, not the mechanism." We find that for certain applications, the commonly used framework of configuration space is insufficient. Instead we have presented two robot planning problems that use a different representation called action space.

For cross-country navigation it is very difficult to build configuration space obstacles in response to real-time sensor data. In addition, for non-holonomic vehicles, such as our automated HMMWV, there is no straightforward mapping from desired vehicle state to the first of a series of actions that must be executed to attain the desired state. In the case of excavation, it is even more difficult to deduce the next action a robot excavator must execute given a desired shape of the terrain. This difficulty is further aggravated by the fact that the configuration space corresponding to natural terrain is extremely high dimensional. Both applications succeed in large part because of the action space representation. The autonomous vehicle is able to account for nonlinear dynamics and is able to react to range data in real-time. The robotic excavator is able to circumvent consideration of a computationally intractable configuration space.

The downside of the method we have used is that the plans produced are short-sighted and require guidance in the form of subgoals to ensure that the robot doesn't get stuck; sometimes it is necessary to be sub-optimal in the short run for over all optimality. In practice, we find that coarse grained supervisory planners are effective in providing such subgoals.

References

[1] M. Daily, "Autonomous Cross Country Navigation with the ALV", In *Proc. of the 1988 IEEE International. Conference on Robotics and Automation*, pp. 718-726.

[2] 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.

[3] Georgeff, M. and Lansky A., Reasoning about actions and plans: Proc. of the 1986 workshop,1986, AAAI.

[4] Hendler, J. and Tate, A. and Drummond, M., "AI Planning: Systems and Techniques," AI Magazine, Summer, 1990, Vol 11 (2).

[5] Hoffman, R., and Krotkov, E., "Terrain Mapping for Long Duration Autonomous Walking", In *Proc. IEEE/RSJ International. Conference on Intelligent Robots and Systems*, Raleigh, 1992.

[6] 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.

[7] 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.

[8] Kirk, D. E., *Optimal Control Theory*, Prentice Hall, 1970.

[9] L. Mathies, "Stereo Vision for Planetary Rovers", International Journal of Computer Vision, 8:1, 71-91, 1992.

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

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

[12] Singh, S., *Synthesis of Tactical Plans for Robotic Excavation*, Ph.D Thesis, Robotics Institute, Carnegie Mellon Univ, January 1995.

[13] Singh, S., "Learning to Predict Resistive Forces During Robotic Excavation," In P*roc. International Conference on Robotics and Automation*, Nagoya, May 1995.