Increasing Productivity for Autonomous Mass Excavation

A Thesis Proposal

Patrick Rowe

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15217

May 23, 1997

Abstract

This research focuses on the problem of increasing productivity for the task of autonomous mass excavation. Autonomous excavation has the benefits of higher productivity, lower labor costs, increased safety, and the ability to work in hazardous environments. Mass excavation involves rapidly loading trucks with soil/rock/ore using a mobile digging machine with a bucket on an arm-like appendage. It is desirable for this operation to proceed very quickly, load the trucks evenly, avoid excessive spillage, and perform safely while operating in a wide variety of possible digging conditions and worksite configurations.

Preliminary work has been done on planning the excavator's motions using a script-based approach, which takes advantage of the fact that the excavator's motions are very similar for each bucket load, but the kinematic details can change due to changes in digging, dumping, and truck locations. Currently these kinematic details, also known as the script parameters, are computed using inverse kinematics, simple machine models, heuristics, and "magic" numbers which need to be adjusted from time to time. To date, this technique has worked very well on our test site, with autonomous excavator productivity approaching that of a skilled human operator. It is unclear, however, if the current script parameter computation will achieve the same results on different materials and in different worksite configurations, or if the productivity could be increased even more.

This research proposes the idea of using information gathered on-line as the excavator performs its task in its current working conditions as a better way of computing the script parameters. Information about an excavator action, i.e. one set of script parameters, and the results of that action, such as the execution time, location of dumped soil, etc., would be recorded. Robot learning techniques, particularly memory-based learning, have been investigated as a method of using this information to compute the excavator's next action which will better achieve its performance goals. Preliminary results of a simple learning system show it is possible for the excavator to become faster and more accurate with manipulating the soil. It is believed this proposed approach will result in productivity that equals or exceeds human operators while maintaining the ability to be productive in a wide range of working conditions.

1.0 Introduction

The automation of large construction and earth-moving machines holds the promise of increased safety, higher productivity, and lower labor costs. Automation will allow these machines to work in areas that are hazardous or inaccessible to humans, such as toxic environments, and increased sensing and reasoning capabilities about the machine's own actions provides a way to prevent unsafe conditions. Higher productivity results from these machines performing their tasks at peak machine performance all of the time, thereby eliminating factors such as human fatigue. Lower labor costs can be the result of having one human operator supervise several autonomous machines simultaneously, as opposed to requiring an operator for each machine.

FIGURE 1. Hydraulic excavator (left) and wheel loader (right) loading trucks during a mass excavation task.

This research primarily focuses on automating a hydraulic excavator, shown on the left, although much of the proposed work could also be applied to other machines, like a front-end loader shown on the right in the figure above. The task to perform is mass excavation where large amounts of earth and rock are excavated from one location and loaded into trucks which haul it away. The machine motions are highly repetitive, and the task is performed very quickly with skilled human operators loading on the order several hundred trucks in a typical work shift.

There is a clear need for these machines to operate at peak levels of performance and to deal with a wide range of digging conditions. This research will explore ways for these machines to meet these goals and ultimately do their tasks better by gathering data on-line as they perform their tasks and learning from past experience. Better task performance means higher productivity and can be defined by two criteria: doing the task faster and doing the task more precisely.

The benefits of performing the task faster are obvious in terms of increasing productivity by decreasing the overall time to load a truck. For example, if merely 2 seconds were shaved off of the time for each bucket load, the resulting time savings would be enough to load an additional 26 trucks in a work shift. Doing the task faster involves an understanding

of the machine's own performance abilities, how fast it can move, etc.

The second criterion for improving overall productivity is doing the task more precisely in terms of manipulating the excavated material. It would be meaningless to have an autonomous excavation system which worked extremely fast but did so by throwing the soil everywhere. Here precision means minimizing spillage of the earth by placing as much as possible in the truck, placing the earth evenly in the truck to avoid an unbalanced load, and digging a desired amount of soil to avoid overloading the trucks. Doing the task more precisely involves an understanding of how the excavated material behaves, whether it is sticky mud or loose gravel.

This research proposes the idea of applying robot learning techniques to improve the autonomous performance of these machines. Simply put, robot learning involves using past experiences and actions of the robot to achieve current performance goals. It is proposed that the learning be done on-line, as the robot goes about its task, as opposed to off-line where data is collected ahead of time and processed, for example when collecting training data for a neural network. The idea of applying on-line learning to the task of mass excavation is a good one for a number of reasons.

- The general machine motions are highly repetitive, and the task is performed relatively quickly, so data of past actions and results are readily available as the machine goes about its job. This offers the opportunity of improving its performance quickly.
- The machine motions for each bucket load are very similar which offers the advantage that only small sections of the vast space of possible task parameters need to be explored. For example, assuming that the trucks are roughly the same size and park in approximately the same location relative to the digging machine, then there is only a small area of the machine's workspace where it is desirable to dump the soil.
- There are a large number of potential worksite configurations for these machines to work with. It would be desirable to place an autonomous excavator in a new worksite configuration and have it improve its performance as it performs its task, as opposed to performing some potentially tedious and time consuming off-line data collection procedure which might require a human operator. The off-line data collection procedure may also have undesirable side effects. For example, digging buckets of soil and dumping them where trucks will park in the future means additional work would have to be done to clean up this soil.
- These machines need to deal with a wide variety of natural materials such as dirt, sand, or rock whose behavior may be unknown ahead of time and can be difficult to model even with empirical information. Through its own interaction with the material as it performs its task in its current task conditions, the machine can learn a mapping from action to outcome, for example where the bulk of the soil ends up after a particular dumping maneuver, and achieve better precision with the material.
- Similarly, constructing models of the machine's own performance characteristics is also difficult. These models may not be able to capture all of the relevant dynamics of a particular digging machine, and the models could be different for different machines and different hydraulic actuation systems. These machines have been designed to

change links easily, so different links and link lengths may make a preexisting model useless. Changes in temperature may also affect the excavator's dynamics. Again, through the machine's own actions, it should be possible to learn enough about its own capabilities under the current working conditions to improve its task performance and overall productivity.

The next section will describe some related research on autonomous excavation, primarily focusing on what other sort of learning methods have been applied. Section 3.0 will give a problem statement and some predictions of what is hoped to be achieved with this research. Section 4.0 will describe the current work on planning the motions of an excavator for the task of autonomous mass excavation. Section 5.0 will present the technical approach and some initial ideas for the components of the learning system that this research proposes to build, along with a discussion of the challenges involved in such a system. Section 6.0 will describe our experimental test site and will present some preliminary results of an early system which learns to do better. Finally, Section 7.0 will give the expected contributions of this work and a schedule.

2.0 Related Research

Research on autonomous excavation has been done at many levels of the problem, from low level trajectory control to high level strategic planning. Singh [26] presents a survey of current state of the art autonomous excavation systems. This section will offer a brief description of several systems, focusing primarily on what other learning techniques have been applied to the problem of autonomous excavation.

Much of the research has concentrated on the subproblem of digging, or controlling the tip of the bucket along a predefined trajectory through soil. This is a challenging problem because of the variability of the soil conditions, the difficulty in modeling the interaction between the bucket and the soil, and dealing with problems like encountering large boulders which could mean having to adjust the current digging plan. Researchers at Purdue University [28] improved their controller by adding a feed-forward torque term to the control torques computed by a PD feedback controller. The feed-forward torque was found using a neural network, which was trained to compute the inverse dynamics of the excavator. The inputs to the net are the desired joint positions at times t, t-1, and t-2 and the outputs are the feed-forward torques for the different joints. Adding the feed-forward torque improved overall tracking and stability, and shows that a complete dynamic model of a machine like an excavator is difficult to construct analytically, so learning methods are an appropriate approach. The research that is proposed here, however, is not at this level of control, but rather at a higher level of generating the desired machine motions for an excavation task like loading a truck.

Other researchers have forgone explicit trajectory tracking and instead have used rule based systems for generating the motions of an autonomous excavator. A research group at the University of Lancaster, England [16, 17] uses a production system with a large number of rules (about 70) which can deal with the variability in digging. It is evident that the large rule base provides a great deal of functionality to the system, however it is unclear how the

parameters or thresholds in the rules themselves are generated, and there seems to be no mechanism to adapt the rules based on past experience. Therefore, if the production system rules prescribe a machine motion which is suboptimal under the current task conditions, then it will be suboptimal all of the time (unless a researcher adjusts the rules or adds new rules to the rule base).

Another type of rule based system has been developed by a group at the University of Arizona [5, 22, 23]. In their system, excavation tasks, such as loading a truck, are broken down into constituent behaviors, which themselves are made up of lower level primitive actions. Sets of tasks and behaviors are organized into finite state machines. Transitions between states are decided using neural networks which have been trained from previously generated data of the current task at hand, although it is unclear exactly how this data is obtained. Again, these researchers appear to be primarily interested in functionality and the ability to handle difficult scenarios, such as hard or rocky digging conditions. They do not appear to consider optimality of the motions, however, nor do they have the ability to adjust the digging rule system or finite state machines based on past experience, since the neural nets are trained in batch mode off-line before the task even begins. Like the University of Lancaster's system, their solution to the problem involves having many rules which can do many different things, but comes at the cost of added system complexity.

At the higher level problem of tactical dig planning, Singh [24, 25] explored various learning techniques. His planning system selects a parameterized digging action which optimizes a cost criterion while not violating any geometric or force constraints. He investigated learning techniques which could predict the force needed for a particular dig to determine if the excavator could generate the required force and not violate the force constraint. He investigated global parametric regression using basis functions which were chosen from known equations of soil models, neural networks, and memory based learning using locally weighted linear regression. He concluded that no one method stood out as superior to the others, that predicting forces of large magnitude accurately was important, and that memory-based learning methods would be appropriate if there is a continuously growing data set, a characteristic which is similar to the mass excavation task described here.

To summarize, the differences that I have found between the work that I am proposing and other relevant research in the field of autonomous excavation is that many researchers have focused on functionality over performance, and many of the systems are content with training their learning systems off-line prior to performance. This work proposes to take advantage of the high repetition of the truck loading task and to use the knowledge gained from every bucket loading pass to improve the next bucket loading cycle. Thus it will use each bucket loading pass as an experiment to improve its own performance on-line.

3.0 Problem Statement

The problem that the proposed research is attempting to solve is that of increasing productivity for the application of autonomous mass excavation and truck loading. It is clear that increasing productivity is very important to potential users of this technology. As stated earlier, increased productivity means performing the task in the minimum time at maxi-

mum machine efficiency. It also includes getting the majority of the excavated material in the truck while avoiding unwanted side effects like excessive spillage and unbalanced loading. One metric which combines these two is the throughput of the loaded soil, or how many tons per hour of earth is loaded into the trucks.

It is estimated that the proposed on-line learning approach can increase autonomous productivity to the point of expert human operators. The current motion planning scheme, which will be described below in Section 4.0, already provides competent machine performance with productivity levels between 80% to 90% of skilled human operators. It is predicted that the method outlined in this research will be able to squeeze out the remaining percentage points and bring autonomous productivity to 100%, or perhaps beyond, that of an expert human.

The proposed on-line learning approach has the advantage of generality. Because the excavator will be gathering data on-line as it performs its task in its current working conditions, the approach holds the promise of working at highest productivity in a large number of different worksite configurations, in different digging conditions, and with different excavators. This is a desired characteristic that is also very important to potential users of this technology.

4.0 Preliminary Motion Planning

My current work on the autonomous mass excavation task has focussed on planning the machine motions for the dumping phase of each bucket loading pass. This is the part of the task when the bucket is not in contact with the ground. The digging algorithm was previously implemented and will be described later in this section.

The first attempt at planning the motions of the excavator was based on a traditional manipulator trajectory generation scheme, but that did not work very well for several reasons. For one, the four joints of a hydraulic excavator are not powered independently. Rather, one hydraulic pump powers two of the joints, and a second hydraulic pump powers the other two. When moving by themselves, each individual joint can achieve a certain peak velocity, however, when both coupled joints are moving simultaneously, neither joint can achieve its own independent peak velocity. Minimal time trajectories were found using these individual joint velocity limits. As a result, the trajectories could not be tracked because it assumed each joint could achieve its maximum velocity regardless of the other joint's motion. If machine models or sets of equations which captured this coupled motion were available, then it is possible they could be used to plan more realistic trajectories, but initial investigations revealed that these models would be very difficult to construct while still performing the computation in an acceptable time.

A second difficulty with the trajectory generation scheme was there really was no notion of a start or end point for the trajectory. Unlike a typical manipulator pick-and-place problem, where the objects being manipulated are solid polyhedra, the material that is being manipulated here is soil which is highly deformable, so the motions to dig and dump the material involves more complicated continuous motions of the joints. In the trajectory gen-

eration scheme, artificial via points were computed which the planned trajectory would pass through, however it was unclear exactly where these via points should be, and it was found that poorly placed via points forced the excavator to take a path that was not along its natural motion resulting in slower bucket loading times.

Another more practical reason for why the trajectory generation scheme failed was the low level PD controller that was supplied with the excavator testbed was not a true trajectory tracker, and it was found that it operated best when given a step command of a desired joint position rather than a particular joint velocity profile. In order to achieve our specified performance goals, it was clear that another type of motion planning algorithm was needed.

4.1 Parameterized Scripts

Careful consideration of the mass excavation task showed that the general trends of the machine motion for each loading pass were very similar, only a few types of machine motions were needed, but specific kinematic details of the motion frequently changed. For example, the trucks being loaded could be of different sizes and could be parked in slightly different locations or orientations. The exact spot of where to dump the earth in the truck is not always the same, nor is the location of the next dig.

The new algorithm that was devised and implemented for planning the excavator's motions is referred to as *parameterized scripts* [14]. A *script* can be thought of as a general template which describes how to do a certain task as a series of simple steps. The script defines the general directions of motions (e.g. move this joint up) as well as the coupling between different degrees of freedom (e.g. when joint 1 has passed a certain angle, start moving joint 2). The script steps for the mass excavation script were designed with the input of an expert human excavator operator, and that knowledge is reflected in the structure of the script itself. For example, if it was advised that moving two particular joints at the same time was a bad idea, then that joint coupling is not present in the script. The script implicitly constrains the possible motions of the machine to only those which are needed for a particular task. The script structure is unchanging during the execution of the task.

The *parameters* of the script are what do change during the execution of the task. They are the variables in the script steps which define the specific machine motion. The values of these parameters are computed based on information about the task environment. In the case of mass excavation, this information is the current truck location and dimensions, the desired dig location and the desired dump location for every bucket load. These values are sent as input to the parameter computation part of the motion planner which computes a value for each script variable. These script parameters are then used by the script during task execution to determine the commands which go to the excavator's low level controller. On each subsequent bucket load, as the inputs change, so do the parameters in the script and the motion of the machine itself. This is summarized in the figure below.

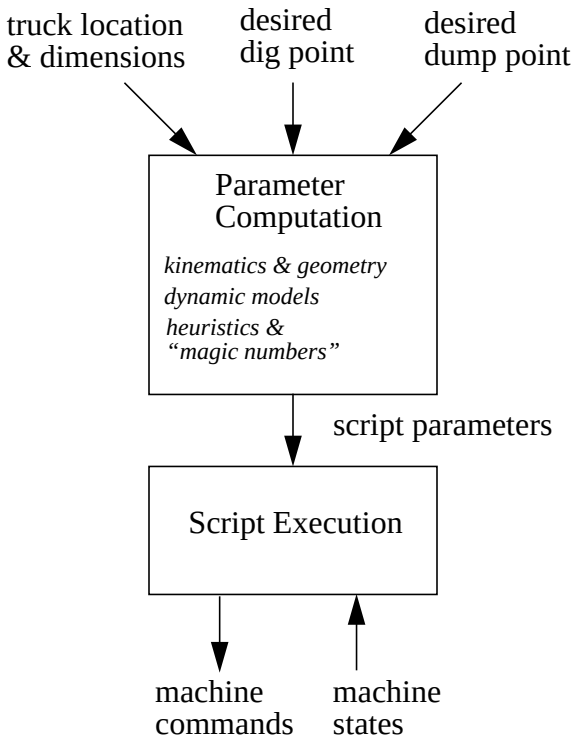

FIGURE 2. Flow chart showing current motion planning scheme. Script parameters are computed from external task information and are then filled into the script template which determines the machine commands.

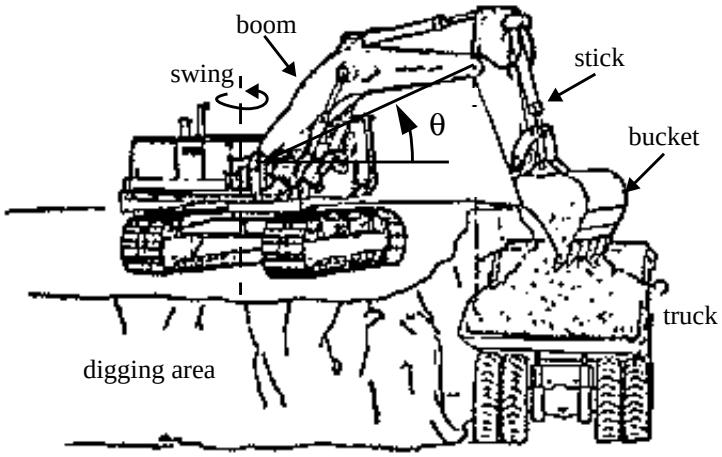

FIGURE 3. Boom angle θ which allows the bucket to safely clear the truck

4.2 Parameter Computation

Most of the script parameters are computed using geometric and kinematic information about the machine and its workspace. For example, Figure 3 shows one parameter value, the angle of the boom for the bucket to safely clear the truck. This computation requires

knowledge of the excavator's link lengths as well as the location and height of the truck. Script parameters which are of this class are typically those which define the desired goal positions for each joint during the different steps of the scripted motion.

Other types of script parameters dictate the coupling between the joints, or how one joint should move with respect to another. These are computed using a simplified machine model which contains knowledge about the excavator's dynamic characteristics, such as joint velocities and latencies. The model is used to answer questions about how long it will take for a joint to move from point A to point B, for instance. One example of this type of script parameter is the boom angle which triggers when to begin swinging towards the truck with the bucket of soil. In order to be more efficient, the excavator does not have to wait for the boom to reach the safe clearance angle θ shown in Figure 3 before it begins to swing. Rather, it can take advantage of the free space between the digging area and the truck, but in order to safely couple the motions of the two joints, it must know the times it will take to reach the desired boom angle and to swing to the truck. It uses the machine model to get this information, and then computes the appropriate boom trigger angle.

The model, however, cannot be used to compute all of these trigger parameters, so heuristics and "magic" numbers are also employed. These are primarily used for dealing with parameters related to soil, such as the bucket angle when all the soil has fallen out of the bucket. They are currently tweaked by hand based on the observed soil conditions, and there is currently no way to automatically select them. Poorly chosen values can have drastic effects. For example, too shallow a bucket dumping angle and the soil does not fall out of the bucket, where too large an angle means the excavator is wasting precious time as it waits for the bucket to reach the desired angle.

4.3 Mass excavation parameterized script

To make the entire algorithm more concrete, below is an example of a mass excavation script for an excavator. The script was designed with a number of goals in mind:

- safely avoid collisions with all known obstacles in the workspace such as the truck and the soil,
- minimize the time for each bucket load by intelligently coupling the joint motions, and
- minimize the spillage of soil during dumping.

The numbers in the bold font are the script parameters that are computed for every bucket load. Every time the excavator digs a new bucket of soil, these numbers are recomputed for the current bucket loading pass. For this script, all of the parameters are joint angles. The parameters which are defined as commands on the right hand side, after the word "Command," are primarily computed geometrically. They are associated with each script step and are sent to the low level joint controller. The parameters that appear in the "rule" part of each script step are the trigger parameters that are primarily computed using machine models and heuristics. Refer to Figure 3 for the names for the different joints.

```
Swing:
1) When digging finishes, wait                    Command = 5°
2) When boom angle > 14°, swing to truck          Command = 101°
3) When bucket angle > 10°, swing to dig face     Command = -5°
4) When swing angle = 5°, stop to dig             Command = -5°

Boom:
1) When digging finishes, raise boom              Command = 18°
2) When swing angle > 60° and stick angle > -89°,Command = 16°
   begin dump procedure
3) When bucket angle > -30°, continue to dump     Command = 18°
4) When swing angle < 60°, lower boom to dig      Command = 14°

Stick:
1) When digging finishes, wait                    Command = -100°
2) When swing angle > 31°, begin dump procedure   Command = -76°
3) When bucket angle > -30°, continue to dump     Command = -101°
4) When swing angle < 65°, move to dig            Command = -85°

Bucket:
1) When digging finishes, capture soil            Command = -90°
2) When swing angle > 60° and stick angle > -89°,Command = 30°
   open bucket
3) When swing angle < 60°, move to dig            Command = 7°
```

FIGURE 4. Mass excavation script for an excavator with one set of parameters

As an example of how to read the script, consider the first two steps of the swing joint's sub-script. The script begins at step 1, when digging has completed, and the swing command associated with step 1 is 5°, which must be the current swing angle. During this time, the boom angle is monitored as the boom is raising up out of the ground. When the boom passes a certain angle, in this case 14°, the script step switches from step 1 to step 2, and the swing joint command switches from 5° to 101°, which causes the excavator to swing toward the designated dump location. The script continues to send this swing command until the conditions are satisfied to switch to step 3, and change the swing command again.

The parameterized script approach to motion planning brought about a dramatic improvement in the excavator's productivity and speed over the previous trajectory generation based approach. Average bucket loading and truck loading cycle times neared, and in some cases exceeded, those of an expert human operator.

There is still room for improvement in this approach, however, particularly in the way the script parameters are computed. Many of the heuristics and magic numbers in the parameter computation are ad hoc and need to be adjusted based on the current soil conditions, which even at our test site are highly variable with weather and moisture changes. Script parameters that are computed using the simplified machine model can only be as accurate as the model itself, which may not capture all relevant dynamic characteristics such as the full hydraulic coupling between the joints. The idea of taking advantage of the repetitiveness of the task motion by gathering and storing some information on every bucket load and using that information as a way to improve performance is a powerful one. It is this

idea that makes up the core of the proposed research and will be further discussed in Section 5.0.

4.4 AutoDig

The digging algorithm, known as AutoDig [13], is similar in nature to the parameterized script approach, but with a few differences. It is similar in that the digging process is described as a series of steps shown below, with transitions that take place when certain events occur. Like parameterized scripts, these events are when angles are reached, as well as when a hydraulic cylinder pressures exceed preset thresholds.

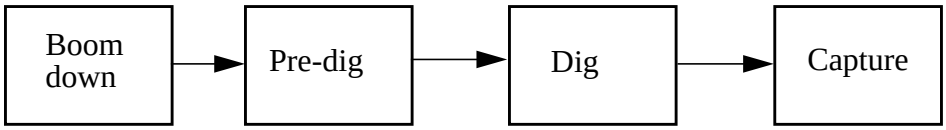

FIGURE 5. Simplified AutoDig transition flow chart

The difference between AutoDig and parameterized scripts is how motion commands are generated. The parameterized scripts send out desired angle positions, which themselves are parameters computed for every different bucket loading pass. The commands that AutoDig computes are a percentage of the maximum cylinder velocity for each of the cylinders. These cylinder velocity commands are a function of the measured cylinder pressures and are found using look-up tables that were found empirically for a number of different soil conditions ranging from easy to hard digging.

In the current implementation of AutoDig, the angle and pressure thresholds, as well as which look-up table to use, must all be given to the algorithm, usually from a human operator. The same ideas of using past experience to select these values and improve the digging performance could be applied to the AutoDig algorithm as well in its current form.

Both the parameterized scripts and AutoDig are similar in that they define the machine's motions as a finite set of values which, when filled into the appropriate script, results in a particular motion of the machine. The union of the AutoDig parameters and the mass excavation script parameters are one complete set of numbers for one bucket loading pass. It is unclear at this point whether or not to continue to separate the entire motion into a digging and dumping phase or to combine the two into a unified motion, and what the implications on the learning algorithms will be. The answer to this question will become clearer as the research proceeds.

5.0 Technical Approach

This research will apply robot learning techniques to the task of mass excavation in order to achieve better performance from the autonomous excavation system, and, in doing so, tackle the issues involved in what it takes to implement a practical learning system for a real world robotics problem. The ultimate goal of the system is to have the autonomous excavator improve its performance as defined by the criteria in Section 1.0: doing the task faster in terms of average time it takes to load a truck, and doing the task more accurately in terms of the amount of soil captured in the bucket and the placement of soil in the truck. It will do so by using past experiences of its own actions and the results of those actions to generate new actions which will result in better task performance. The figure below illustrates how this new approach differs from the current motion planning approach.

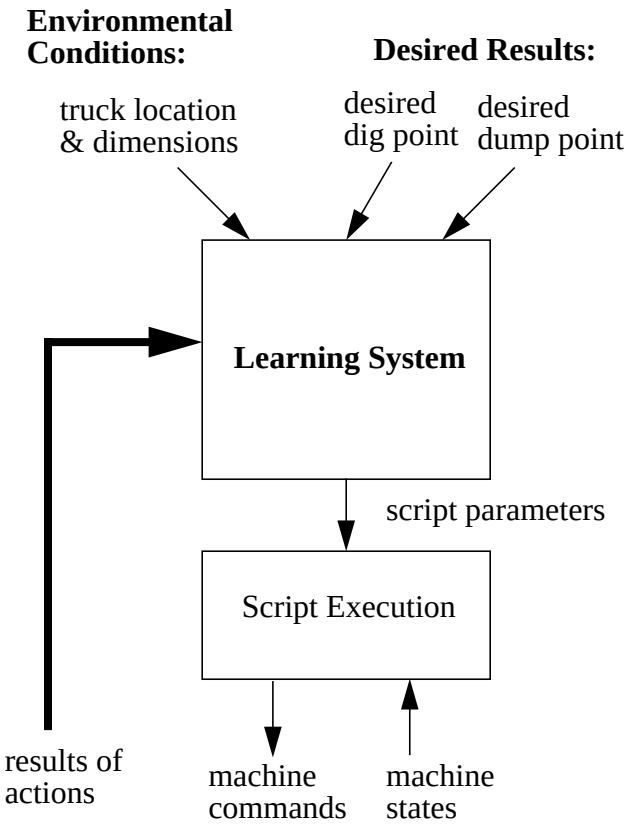

FIGURE 6. Flow chart showing the new approach to motion planning where motion script parameters are computed using past experience, which it is hoped will improve current performance.

There are several components of the learning system which will be developed in the course of this research, and are discussed in the following sections. These are:

- an encoding of what makes up a past experience which is to remembered and learned from. This includes:
  - relevant initial and environmental conditions about the world,

    - the machine action which was taken, and
    - the resulting outcomes of the action.
- software modules which will use perceptual sensors to measure and calculate information about the results of the excavator's actions,
- the learning algorithm itself, which returns an action for the excavator to attempt which it believes will achieve the desired result and increase productivity. This component consists of a number of pieces including:
    - a way of predicting the results of a candidate action (function approximator),
    - a method for generating new actions for the excavator which will minimize a cost criterion (optimization).

This section will also discuss some issues and open problems related to the proposed research. It is not intended that a solution will be found to all of these items in the course of this research, and as the work continues some may present themselves as being more important than others. Some initial ideas about how to deal with these issues are presented. A quick summary of these issues are:

- learning safely,
- learning without a "practice" or "experimentation" phase,
- dealing with noisy, inaccurate, or missing measurements of the outcomes of an action,
- beginning execution when there is no past experience,
- and ways domain knowledge, human knowledge, and the use of models may help.

5.1 Past Experiences

The central idea of this proposed research is gathering some information from each of the hundreds, or even thousands, of bucket loading passes that the excavator performs in order to increase its productivity. The first question, then, is what sort of information makes up a past experience.

5.1.1 Initial Conditions

One portion of each recorded data point is the initial or environmental conditions at the beginning of the motion. For the digging portion of the task, the shape of the terrain in the local area which has been selected to dig next would probably be the most appropriate information about the state of the world. The challenge comes in how to express this succinctly to be able to easily compare one type of terrain shape to another. A local grid of elevations is one possibility, but that consists of many numbers and may prove too unwieldy unless it is made very coarse in resolution. Another idea is locally approximating the terrain in the desired digging area as a plane and using the normal to the plane. A third idea might be to take a cross-section of the local terrain and define the surface as a 1-D curve.

For the dumping portion of the task, the initial conditions and state information about the world that would most likely be needed are the extent or height of the captured soil in the bucket after digging, information about the truck including its height and location, and perhaps some knowledge of the current soil distribution in the truck. The positions and velocities of the joints after the digging process may also be required since a dumping action may depend on the starting configuration of the excavator.

5.1.2 Machine Action

The second piece of information that needs to be recorded with each data point is, of course, the action that the excavator took. The parameterized script and AutoDig algorithms described in Section 4.0 provide a nice framework for expressing the machine's motion for an entire bucket loading pass as a small set of parameters. This set of numbers would be recorded for each action.

5.1.3 Results

The final piece of information that is contained in each data point are the results of the machine action. For the digging portion of the task, this would include the time it took to dig a bucket of soil and the volume of soil captured in the bucket.

The results of the dumping maneuver would also be execution time as well as the location of the bulk of the dumped material, which would be expressed as Cartesian or polar coordinates relative to the digging machine. The shape of the dumped pile of soil may also be another variable of interest which would be recorded with each data point.

5.2 Measuring the Results

Now that the type of information to be recorded has been specified, the next components of the learning system that will be developed in this research are the software modules which use perceptual and other sensor information to measure the results of a machine action. Some information, such as execution times, are trivial to measure with internal software clocks. The use of high fidelity machine models, which do exist but run far too slow for planning purposes, may be a way to measure any variables of interest on the excavator, such as pump flows or pressures for calculating the total work done.

For the variables which involve the soil, it is anticipated that these would be measured perceptually. For example, for the digging task, the volume of captured soil will need to be measured and recorded. This could be done by scanning the soil in the bucket immediately after digging with a range sensor, and constructing a high resolution terrain map which will be used for a volumetric estimation.

The result that will need to be measured for the dumping portion of the bucket loading cycle is the location of the soil after it is dumped from the bucket. This could be a simple calculation, such as computing the "center of mass" of the soil pile using a terrain grid, or it may need to be more complex involving the shape of the deposited soil as well.

5.3 Action Generation

The major component of the learning system is the algorithm which computes the next action for the excavator given a set of initial conditions and a desired result. It does so by using experience of past actions and results, from which it can learn a mapping from action to outcome and use this to better achieve the desired performance goals.

The desired results which are given to the learning system are the same as those mentioned in Section 5.1.3. In this context, these are the desired behaviors for the excavator, such as dump the next bucket of soil at these coordinates, or capture a specified volume of soil in the bucket. Ideally, the action that the excavator takes would achieve these desired results perfectly. This information would come from higher level tactical planners which decide where to dig and dump for each bucket loading pass.

5.3.1 Inverse Models

One approach to compute the recommended action is to build an inverse model. That is, given a data base of points, where the inputs are the machine actions and the outputs are the results of those actions, the inverse model map outcomes to machine actions. One way this could be accomplished is by fitting a function to the points in the data base where the inputs are desired outcomes, and the output is the machine action. When a new desired outcome is presented, the approximated function is used to compute the machine action. Although this seems to be an intuitively simple idea, there are a number of problems with such an approach. For one, if there are a different number of inputs and outputs, then the model that maps actions to outcomes may not be invertible, or it may be inverted incorrectly and compute the wrong action for a desired outcome. Incorrectly inverted models could lead to situations where the robot repeats the same mistake over and over again and does not learn or improve [7, 21]. A simple 1-D example is shown in the figure below[1]. The true inverse, which is not a function, is shown in the middle graph. The function that would be fitted to the data points is shown in the far right plot. There are certain areas where it is vastly different from the true inverse, and would return an incorrect answer.

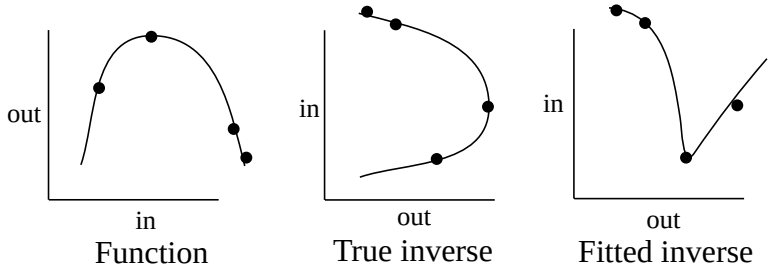

FIGURE 7. One problem with inverting a forward model. The middle plot is the true inverse. The right plot is the fitted inverse.

With these problems in mind, another approach is to use the more natural representation of data points and build a forward model. This approach requires two pieces, a way of predict-

1. The idea for these plots is taken from the Vizier tutorial [21]

ing the outcomes of candidate actions, and a way of selecting an optimal action to take in what could be a high dimensional action space.

5.3.2 Function Approximation

Predicting the outcome of a candidate action, can also be done by means of approximating a function to the data points. There are many ways of doing this ranging from simple to sophisticated and complex. One simple method is nearest neighbor, a technique often used for pattern classification [4]. In this approach, the distance from the candidate action to all other actions stored in the data base is computed, and the output of the closest stored action to the candidate action is returned. While the nearest neighbor approach appears very simple, there are problems if the data is noisy because nearest neighbor will fit the noise. One improvement is to take the average of the outputs of the k nearest neighbors, which could help in smoothing the noise, but still suffers if the neighbors are very far away from the candidate input action. Several robot learning systems have been implemented based on nearest neighbor [6].

A slightly more complicated means of approximating a function is by global parametric regression. This assumes an underlying basis function, such as a polynomial, whose coefficients are computed in such a way to minimize some error metric between predicted and actual data. One problem with this approach, of course, is selecting the right basis functions. It also assumes there is one global set of coefficients which fit all of the data well, which may prove inadequate for complex and highly nonlinear systems.

Neural networks [15] are another means of approximating a highly nonlinear function. They are similar to global parametric regression in the sense that the weights in the network are the parameters which are computed, typically through gradient descent, but the underlying basis function is determined by the network's architecture. Neural networks typically require much training data and a relatively long time to train and compute the weights.

Another function approximation technique which takes ideas from global regression and nearest neighbor is locally weighted regression [3]. In this approach, a simple local model is used to fit the data. Points in the data base are assigned a weight which is proportional to their distance from the input action. Typically an exponential weighting term is used. The coefficients for the local model of the data, such as a linear, or quadratic model, are then computed using the weighted data. Weighting the data gives the effect of having data points which are closer to the input action count more in the final output than those which are farther away, and the locality assumes that there are different coefficients for different areas of the data. This approach can approximate complex, nonlinear functions.

Locally weighted learning and nearest neighbor fall into the class of the memory-based learning paradigm, where all data points are explicitly remembered over the course of a robot's lifetime. This is in contrast to algorithms such as global parametric regression and neural networks where the parameters of the fitted function only need to be remembered, not all of the data. Both approaches have been used to successfully learn and control robots [7, 9, 10]. The approaches have their advantages and disadvantages which will be discussed

below in the context of the mass excavation task.

- Neural networks work well when there are a large number of inputs, such as the pixel values in an image, while memory based learning, because of the matrix computations involved, are best suited for a "moderate" number of inputs. In the case of the parameterized scripts, there are typically less than 15 script variables, which is acceptable for a memory based learner.
- Neural networks need considerable computational time to train the network, but once trained, predictions are computed very quickly, while memory based learning defers all calculation to when it is queried for a prediction. When there is a large amount of training data, neural networks have the advantage, but in the mass excavation scenario, there would be typically on the order of only several hundred data points gathered per day. Also, a typical bucket loading cycle is about 15 to 20 seconds, and the script parameters only need to be computed once per cycle, so there is adequate time for the computation.
- As the mass excavation task progresses, there is a constant incoming stream of data points. With memory based learning, the new data are simply added to the data base, while with neural nets, it is unclear what to do with new data when it arrives. For example, the network could be retrained, but that may take more computational time than is available.
- Neural network training suffers from problems such as local minima and data interference. With memory based learning, the answer to a query is computed analytically, and all data is explicitly remembered which helps with the interference problem.
- Finally, memory based learning has some other nice features like cheap cross validation to update parameters used in the weighting and regression routines and "free" gradient estimates. With neural networks, considerable computation is required to produce these items. Other discussions of memory based learning vs. neural networks can be found in [3, 21].

There are many other function approximators that can be used such as radial basis functions, CMACs [1], decision trees [13], and so forth. This research is not concerned with surveying and comparing the different methods but rather selecting one method and building a working learning system with the chosen learning paradigm. For this research, it appears that memory based learning has advantages over other techniques, therefore it will be used in this work.

One distinction that should be made clear is precisely what function the system is learning. The system is not learning explicit models of the excavator's dynamic characteristics, nor a model of how the excavated soil behaves. Rather, it is a combination of several things including the rules of motion as defined by the truck loading parameterized script, the low level closed loop controller, the open loop excavator dynamics, and the soil behavior shown in the figure below. The input to this function is a set of script parameters defining a machine action. The outputs of the functions are the quantities of interest, such as total execution time, soil location, etc., which are measured using sensors. Assuming one had accurate

models of all of the separate components mentioned above, then one could analytically build such as function, but, since deriving models of both the machine's dynamic characteristics and soil behavior is very difficult, it appears that learning the combined function is a good approach for this application.

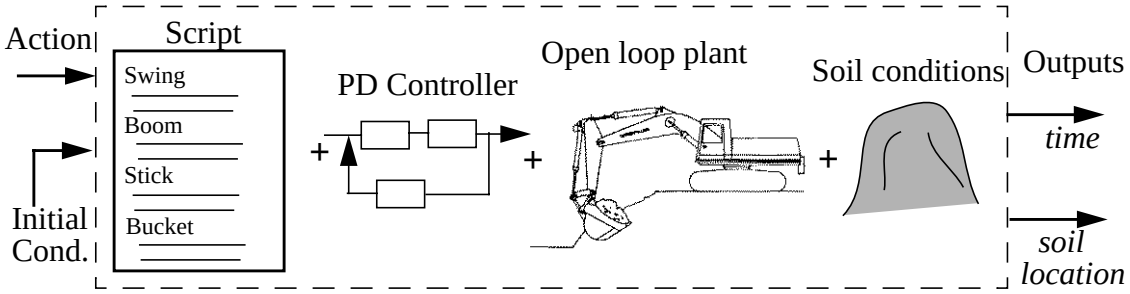

FIGURE 8. Schematic illustrating all of the components of the functions which are being learned.

5.3.3 Action Selection

The other major piece of the learning system is the routine which selects the next machine action. This has also been referred to as experiment selection or experiment design in the literature. For our purposes, we are concerned with selecting an action which attempts to achieve certain performance goals.

There are a number of ways to choose an action. If the dimensionality of the action space is small enough, then a brute force search over all actions at some finite resolution may be acceptable [27]. In other instances, where the dimensionality is high (say above 3 or 4), some possibilities may be choosing actions at random, or perhaps making random alterations to the best actions so far, much in the way genetic algorithms operate [15]. Schneider [18] used an interpolation approach which creates a new action from a linear combination of previous actions in the data base. Moore's SAB action chooser [6] generates actions at random, but uses a notion of reliability of the predicted accuracy as a way to prioritize the actions. For example, an action which predicts an outcome that is close to the desired outcome and is reliable is rated higher than an action which also predicts a desired outcome, but the prediction is unreliable.

If the desired outcome can be expressed as a cost function which is to be minimized or maximized, then an algorithm such as gradient descent can be used to select an action. One example of a cost function that may be appropriate for the dumping phase of the mass excavation task using the performance criteria defined earlier is

$$c = w_1 t + w_2 (loc_{des} - loc_{act})^2$$

where t is execution time, w_1 and w_2 are weight values on the different terms, and loc_{des} - loc_{act} is the error between desired and actual location of the soil pile during the dumping phase of motion. Other criteria that could go in other cost functions are

- error between desired and actual volume of soil captured in the bucket during digging.

- machine efficiency in terms of power available versus power used. It is desirable to operate these machines at peak power levels.
- perhaps a measure of smoothness of the motion, particularly if there are perception sensors mounted on the excavator.

With this technique, selecting the next action has been stated as an optimization problem. There are many techniques for optimization of a function [2, 29]. This approach looks promising because the learning system will select what it believes to be the best action to take given what it knows so far, as opposed to a random action which could be quite dangerous for this application. Moore and Schneider have explored function optimization using the memory based learning paradigm [8]. A variety of different optimization techniques were tried, with variations on how confidence measures are used, to optimize noisy nonlinear functions. Although the work is preliminary, it was found that memory based learning methods can help to accelerate the convergence of the optimization routines.

5.4 Related Issues and Open Problems

This section will discuss some of the issues and challenges that will be addressed at one level or another in this work. It is not intended that a complete and in-depth solution to all of these issues will be found in this research, although with time permitting, some may be explored a little deeper. Some initial ideas are presented to deal with some of the issues to allow the main area of this research to proceed.

5.4.1 Learning Safely

Implicit in all of the performance goals that have been mentioned so far is the goal of safety, such as not colliding with things in the environment like the truck or ground. A compromise must be found between task speed and efficiency, and the safety of the machine motion. The ability to learn and optimize safely is of importance to researchers and have been discussed [20].

One way of dealing with this issue is to use the notion of the confidence or reliability of the predictions. These confidence measures can be incorporated into the optimization algorithms, for instance by selecting the action which best minimizes the cost criteria and is within some acceptable bounds on the confidence of the prediction.

Another idea is to place bounds on the space of possible excavator actions. With the parameterized script approach, many of the dimensions of the action space are excavator joint angles. Knowledge about the current geometric constraints of the worksite could be used to compute the joint angles which would definitely cause a collision, and reduce the size of the input space as well.

The use of the simple machine models, which are currently used in the parameter calculation may also find a place in dealing with this problem. These models can serve as an additional check on the candidate action by simulating the machine's motion before it is

executed on the real robot to check for any gross errors or potential collisions, and reject the action before it happens. Of course, it is hoped the learning system, once it has received enough data, will be more accurate than the simple predictive model, yet if the model is serving as a watchdog, then the actions which are accepted can only be as good as the model. It may be the case that the model is used less and less as the excavator gains more and more data through its actions. Furthermore, if some sort of external collision detection module is used in the system, then that could also abort a potential collision, and the data point is stored with a very high cost.

5.4.2 Experimentation vs. Performance

Several robot learning systems require a period of experimentation or "practice" in order to explore areas of the action space which may ultimately lead to a more optimal result [18]. Then, during performance mode, no exploration is performed and only the best actions of the currently learned model are used. Unfortunately, this may not be possible in the truck loading application because of the issues of safety and soil spillage. For example, it may not be acceptable to practice a few bucket loads without the truck in the way because the dumped dirt will end up where the truck is supposed to park and will have to be cleaned up. Therefore the system must be able to learn and improve performance without being able to randomly explore.

This problem is not as crucial in terms of hindering this research, however there is one idea that could be tried. Some initial "calibration" exercises could be performed before the truck loading task begins. There may be a few simple actions that the excavator can perform without the danger of collision or spilling dirt in undesirable locations. For example, experimenting with just the digging motion parameters and then dumping the dirt back in place may be acceptable as a means of experimentation and exploration.

5.4.3 Noisy or Missing Outcome Information

Another problem that will be encountered is what to do if the result of an action can't be measured completely. For example, the walls of the truck may occlude the pile of soil so its full extent cannot be completely scanned by a sensor. Or if the excavator and truck are at the same elevation on the ground, the sensors on the excavator may not be able to see inside the truck at all. Again the idea of pre-operational calibration may come to the rescue here in terms of getting some initial idea of how the soil will behave when it is dumped out of the bucket, and doing this in an area, such as over the digging area, where the result of the action can be measured without obstructions.

Another idea could be to incorporate confidence intervals in the measurements of the results as well, much like they would be incorporated in the machine action recommendations mentioned earlier. The data in the data base may then be weighted with such confidence measures, where highly confident scans and measurements of the location of the soil pile are counted more than less confident measurements.

This problem may turn out to be important, since the proposed system plans to use visual

information and feedback to improve task performance. It is unclear at this point how large a problem this may be. One saving factor may be the fact that loading trucks with dirt is not an exact science, and there may be a broad range of acceptable performance in terms of soil placement. In other words, criteria such as execution time, which can be measured accurately, may prove to be more important than absolute soil placement accuracy in terms of improving overall productivity.

5.4.4 Starting with no data

Another problem is how to begin when there is no data in the data base. One idea is to use data from past excavations or even different excavators assuming the soil conditions and machines are relatively the same. Another idea is to have a human operator initially perform some bucket loading cycles and have the system extract the relevant parameters and measured outcomes and record them in the data base. This gets into the difficult problem of programming by demonstration and may be beyond the scope of this work. Perhaps the best idea is to use the method parameters are computed now. This technique already provides a competent starting point which can then be improved upon once the learning system takes over.

5.4.5 Using domain knowledge

Another key piece of a working learning system is using domain knowledge in an intelligent way. For a general learning system, it is assumed that the relationship between an action and a result is unknown, and it is up the learning system to provide that mapping. But if it is known how some of the action parameters could directly affect the output, then that is a powerful piece of information that could be exploited in the learning system. For example, it may be known that not raising the boom as high after digging (assuming it is still high enough to clear the truck) results in a lower overall loading time independent of other machine actions. Or, that moving the bucket farther away from the machine as it swings to the truck results in the dirt landing at a further distance from the excavator, again independent of other machine actions. This kind of knowledge could provide guidance during the optimization and action selection.

6.0 Experimentation and Preliminary Results

This section presents a description of the tools and testbeds that are available to test the learning system. Some initial results of a learning system that was implemented and tested for the mass excavation task are also presented.

6.1 Experimental Testbed

The ideas presented in the previous section will be implemented and tested in both simulation and on a real digging machine. We currently have a 25 ton hydraulic excavator that has been retrofitted with joint resolvers, cylinder pressure sensors, inclinometers to measure pitch and roll, and a PD position controller for each of the joints. Furthermore, a highly ac-

curate laser range finder has been mounted on the excavator's cab to provide sensor data for the planning system. The excavator sits atop a concrete berm and has a large amount of soil in front of it for digging. A standard on-highway dump truck is available for loading with soil. A picture of our test site are shown below. In order to test the ability of the learning system to handle different soil and digging conditions, other materials, such as concrete or gravel, may be brought in for digging, or the excavator could be taken to other sites with different digging materials.

FIGURE 9. Autonomous Loading System project testsite at the National Robotics Engineering Consortium in Lawrenceville.

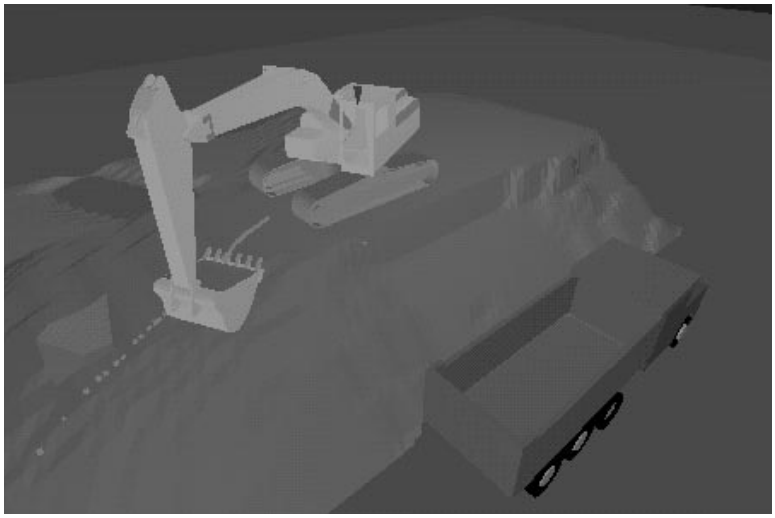

FIGURE 10. Simulated excavator, truck, dig face, and laser scanner.

In addition to the actual machine, we also have a sophisticated simulator which has a range of machine dynamic models, from low fidelity for speed to higher fidelity for increased accuracy. The simulator has a graphical display for visualization of the machine's motions. It can simulate soil being scooped out of the ground and dumped in the truck or on the ground.

The simulated excavator also comes equipped with two simulated laser range finders. The use of the simulator makes it very easy to generate new worksite geometries, different truck locations and orientations, and perhaps different soil conditions which would again test the system's ability to learn. It also provides the ability to run many bucket loading cycles in a short period of time. The figure below shows the simulated excavator with the one of the laser scanners in operation.

The software which makes up the mass excavation planning system is the result of over two years of development and maturation. Many of the software tools and support necessary for the task of mass excavation have already been implemented. These include software modules which use perceptual data for recognizing, measuring and locating the truck, determining where to dig next based on the current terrain conditions, determining where to dump next based on how the soil is resting in the truck, and a collision detection module which monitors for potential collisions of the machine with anything in the environment. There are also many software tools used throughout the system which allow a software module to acquire sensor data, build a terrain map from the data, and so forth.

6.2 Preliminary Learning Results

A preliminary learning system was implemented and tested using the simulated excavator. Simulation was used first in order to better control some of the aspects of the excavation scenario, such as providing a uniform soil behavior, as well as to easily run many trials in a short period of time. The desired task for the excavator to learn was to dump a bucket of soil at certain given coordinates in its workspace in the minimum possible time. The excavator was always given the same desired dump coordinates for each trial.

Several simplifications were made to make this problem a little easier. The excavator always started in the same configuration and always went through exactly the same digging motions, so the initial conditions from trial to trial remained the same and would not need to be included in the data base. The simulation also ensured that the soil would behave the same from trial to trial. Only 7 of the most relevant script parameters for this type of motion were chosen as variables, while the others remained fixed from trial to trial. The script parameters that were chosen are shown below in bold font (these are example numbers).

```
Swing:
1) When digging finishes, wait                    Command = 0°
2) When boom angle > 14°, swing to truck          Command = 100°
3) When bucket angle > 10°, swing to dig face     Command = -5°
4) When swing angle = 0°, stop to dig             Command = -5°

Boom:
1) When digging finishes, raise boom              Command = 18°
2) When swing angle > 60° and stick angle > -89°,Command = 17°
   begin dump procedure
3) When bucket angle > -30°, continue to dump     Command = 18°
4) When swing angle < 45°, lower boom to dig      Command = 8°

Stick:
```

```
1) When digging finishes, wait                    Command = -80°
2) When swing angle > 31°, begin dump procedure Command = -76°
3) When bucket angle > -30°, continue to dump   Command = -101°
4) When swing angle < 45°, move to dig           Command = -60°

Bucket:
1) When digging finishes, capture soil           Command = -90°
2) When swing angle > 60° and stick angle > -89°,Command = 35°
   open bucket
3) When swing angle < 45°, move to dig           Command = 0°
```

The output variables which were recorded for each trial were the total execution time and the location of the dumped soil. The execution time began immediately after digging was finished and ended when the excavator had dumped the dirt and had swung back to where it had dug. The location of the mound of dirt was measured in polar coordinates with the origin located at the excavator's swing pivot shown in the figure below. The simulated truck was moved out of the workspace, and the soil was dumped on the ground. This was done in order to get an unobstructed scan of the dirt pile. Therefore, the output space is 3-dimensional, giving a total of 10 numbers which are stored in the data base for each trial.

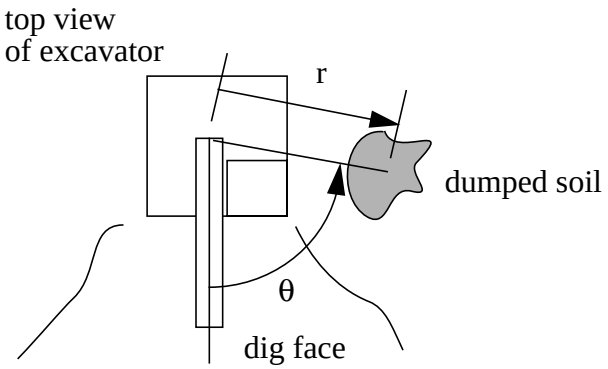

FIGURE 11. Schematic showing polar coordinate frame in which the location of the soil pile is measured.

An initial perceptual software module for measuring the location of the mound of dirt with the simulated laser sensor was implemented. Because the simulated soil is modeled as non-cohesive particles, there is a definite peak to the mound. After the motion is complete, the dirt pile is scanned, and the coordinates of the highest point of the pile of dirt is returned as the location of the dirt pile.

The selected cost function is made up of a linear combination of terms incorporating execution time, radial position error, and tangential position error of the form

$$c = w_1 t + w_2 \left((r_d - r)^2 + r_d^2 (\theta_d - \theta)^2 \right)$$

where t is execution time, r and θ are the measured radius and angle of the soil pile, and r_d and θ_d are the desired coordinates of the soil pile. The r_d^2 in front of the tangential error

term is used to equalize the radial and tangential error using the equation $s = r\theta$ where s is the arc length in the tangential direction at radius r.

An interesting research issue is how to determine the weights for the temporal and spatial terms in the cost function. This can also be asked as how much 1 second of time is worth in units of distance. For example, if $w_1 = 2$ and $w_2 = 400$, then 0.5 seconds of time would be equivalent to 5 cm of spatial error. The ability to adjust these weights may be an added advantage to the system and gives a human supervisor the ability to favor one criterion over the other depending on the task.

In order to get started with no previous experience in the data base, the script parameters for the first 6 trials were computed as they are in the current motion planning system, using geometry, inverse kinematics, simplified machine models, and heuristics. After the parameters were computed, a small random number was added to each of them to ensure they would be different for each initial trial (otherwise they would all be the same since the same desired dump target is given for each trial). The first 6 trials were chosen because that is the number of buckets needed to fill one truck. The simulations were then run using these script parameters, and the outcomes recorded and entered into the data base.

For the function approximation part of the system, a locally weighted linear regression algorithm was used with a weighting function of the form

$$w_i = e^{\frac{-(D(x_i - input))^2}{W}}$$

where x_i is the ith data point in the data base, *input* is the candidate action whose output is being predicted, and *D()* is the Euclidean distance between points in the 7-dimensional input space. All inputs were normalized between 0 and 1 before computing the weight. A leave-one-out cross validation scheme was implemented which selected the best kernel width W from 8 potential widths ranging from 2^{-1} to 2^{-8}.

Both the 7 input and 3 output terms of each data point, get multiplied by their corresponding weight. One way to express this in matrix form is

$$Z = WX$$

$$v = Wy$$

Assuming n points in the data base, and m terms for the input of each data point, 7 in our case, then W is a n x n diagonal matrix of the weights, X is an n x m matrix of the input terms of each data point, and y is an n x 1 vector of the output associated with each data point. In the case of multiple outputs, such as time and spatial accuracy, there would be several different y and v vectors, but the same weights would be used since they are just a function of the distance between input terms. An additional column of 1's was appended to the matrix X, resulting in an n x m+1 matrix so that the linear model need not pass through the origin.

After the Z and v matrices are computed, the m+1 coefficients β of the linear model are

found by solving

$$Z\beta = v$$

In order to provide a numerically stable algorithm in the face of singular, or nearly singular matrices, a singular value decomposition algorithm was implemented to solve the above matrix equation for β. Three different β vectors were computed for each prediction for the 3 different outputs.

Once β has been computed, the predicted output is simply

$$\hat{y} = \beta^T input$$

The action selection algorithm that was chosen was the downhill simplex method [11]. This was fairly easy to implement, runs very fast, and does not require gradient information, but can have problems falling into local minima which are not at acceptable performance levels. Therefore, the downhill simplex routine was started at 4 different starting points which were selected based on the best points so far with the lowest cost, lowest time, lowest radial error and lowest tangential error. Steps were taken to ensure that these four points were different. Other starting points could be at the boundaries of the input space assuming the global minima may lie there, although there would be a lot of boundary points for a high dimensional input space. The linearity of the input space also offers the possibility of using algorithms such as linear programming which are guaranteed to find the global minimum if one exists [2].

The figures below show the results of running with a cost function of

$$c = 5t + 100\left((7.0 - r)^2 + 7.0^2\left(81.4^\circ - \theta \right)^2 \right)$$

where $r_d = 7.0$ and $\theta_d = 81.4^\circ$. The upper plot shows the overall cost, the middle plot shows the time, and the lower plot shows the combined radial and tangential squared error. After the 6 initial trials, there is a drop in the cost which continues to decrease, with some rises here and there. In only a small number of trials, the time and soil placement accuracy have fallen to well within acceptable levels based on human expert performance.

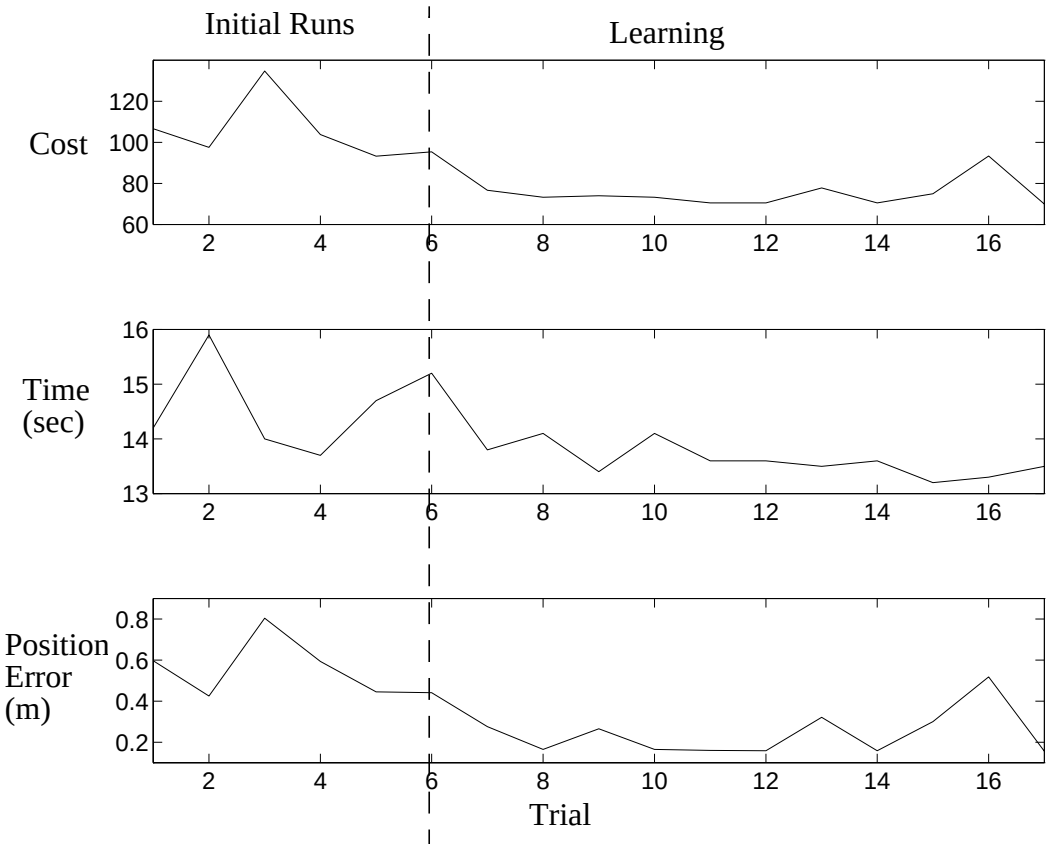

FIGURE 12. Results of applying learning system to soil placement task.

Although several simplifications were made, and the system run in simulation, these results do hold the promise that learning from past experience could result in improved autonomous excavation performance. I feel that there exists a set of steps that will take this initial work to the point of a full autonomous excavation system which can improve its performance on-line. Seven script parameters were used in the above experiments, with the others computed as they are now in the current motion planning system. One by one, more parameters can move from the old parameter computation method to the new one of selecting them by using the learning algorithm

The criteria to be optimized can be broken apart as well and incorporated one by one into the full system. For example, we currently have the ability to easily measure the total execution time. This could be optimized first, without considering the accuracy of soil placement. The different components of the soil location could also be broken apart. We currently have the ability to roughly measure the tangential soil location in the truck (θ in the above cost equations), which is used in the dump point planning algorithm. This information could also be used for the learning system.

7.0 Contributions

The expected contributions of this research to the field are:

- The first autonomous mass excavation and truck loading system which focuses on increasing task productivity by improving its performance on-line as it interacts with its current environment.
- A system to optimize the task of autonomous truck loading which can be used on a wide variety of different excavators working in different soil conditions and different worksite geometries.
- The initial application of memory-based learning techniques to the task of autonomous mass excavation, and further exploration of the general problem of optimization using a memory-based learning paradigm.
- A better motion planning technique, the parameterized scripts approach, for optimizing the motions of any high degree of freedom robot or system.

7.1 Schedule

Spring 1997	Implement preliminary learning system and test in simulation.
	Propose.
Summer 1997	Implement perceptual modules to measure results of actions. Test in simulation and on real excavator.
	Further improve preliminary learning system adding confidence measures and other safety features. Primarily focus on dumping procedure. Test in simulation.
Fall 1997	Increase number of parameters that are computed with learning algorithm.
	Refine and improve perceptual measurement software.
	Continue testing learning system on real excavator.
Winter 1997/1998	Get first mature version of learning system for free motion working.
	Begin work on extending methods to digging.
	Demo partial truck loading system.
Spring 1998	Continue improving and extending system, testing on different soil conditions and work site geometries.

Summer 1998	Finalize learning system and continue to extend digging system.
	Write thesis.
Fall 1998	Defend.
	Graduate.

8.0 References

[1] Albus, J., 1981. *Brains, Behavior, and Robotics*. BYTE Press.

[2] Arora, J., 1989. *Introduction to Optimum Design*. McGraw-Hill, Inc.

[3] Atkeson, C., Moore, A., and Schaal, S., 1997. Locally Weighted Learning. to appear in *AI Review*.

[4] Duda, R., and Hart, P., 1973. *Pattern Classification and Scene Analysis.* John Wiley & Sons.

[5] Lever, P., Wang, F., and Chen, D., 1994. A Fuzzy Control System for an Automated Mining Excavator. *in Proceedings of International Conference on Robotics and Automation*, pp. 3284-3289.

[6] Moore, A., 1990. Efficient Memory-based Learning for Robot Control. Ph.D. Thesis. University of Cambridge. *Computer Science Technical Report 209.*

[7] Moore, A., Atkeson, C., and Schaal, S., 1995. Memory-Based Learning for Control. *CMU Technical Report CMU-RI-TR-95-18.*

[8] Moore, A., and Schneider, J., 1995. Memory-based Stochastic Optimization. *Neural Information Processing Systems- 8.*

[9] Moore, A., Atkeson, C., and Schaal, S., 1997. Locally Weighted Learning for Control. to appear in *AI Review*.

[10] Pomerleau, D., 1993. *Neural Network Perception for Mobile Robot Guidance.* Kluwer, Dordrecht, Netherlands.

[11] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B., 1992. *Numerical Recipes in C.* Cambridge University Press.

[12] Quinlan, J., 1993. *C4.5: Programs for Machine Learning*. Morgan-Kaufman, San Mateo, California.

[13] Rocke, D. 1995. *Automatic Excavation Control System and Method.* US Patent No. 05446980.

[14] Rowe, P., and Stentz, A., 1996. Parameterized Scripts for Motion Planning. To appear *in Proceedings of International Conference on Intelligent Robots and Systems.*

[15] Russell, S., and Norvig, P., 1995. *Artificial Intelligence A Modern Approach*. Prentice-Hall Inc., NJ.

[16] Seward, D., 1992. LUCIE - The Autonomous Robot Excavator. *Industrial Robot International Quarterly.* Vol. 19., No. 1. pp. 14-18.

[17] Seward, D., Margrave, F., Sommerville, I., and Morrey, R. 1996. LUCIE the Robot Excavator - Design for System Safety. *in Proceedings of International Conference on Robotics and Automation.* pp. 963-968.

[18] Schneider, J., 1995. Robot Skill Learning Through Intelligent Experimentation. Ph.D. Thesis. University of Rochester. *Computer Science Technical Report 567.*

[19] Schneider, J., and Brown, C., 1995. Cooperative Coaching in Robot Learning. *in Proceedings of International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 332-337.

[20] Schneider, J., 1996. Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning. *Neural Information Processing Systems 9.*

[21] Schneider, J., and Moore, A., 1997. A Locally Weighted Learning Tutorial using Vizier 1.0.

[22] Shi, X., Wang, F., and Lever, P., 1995. Task and Behavior Formulations for Robotic Rock Excavation. *10th IEEE International Symposium on Intelligent Control*, pp. 248-253.

[23] Shi, X., Lever, P., and Wang F., 1996. Experimental Robotic Excavation with Fuzzy Logic and Neural Networks. *in Proceedings of International Conference on Robotics and Automation*. pp. 957-962.

[24] Singh, S., 1995. Synthesis of Tactical Plans for Robotic Excavation. Ph.D. Thesis. Carnegie Mellon University.

[25] Singh, S., 1995. Learning to Predict Resistive Forces During Robotic Excavation. in *Proceedings of International Conference on Robotics and Automation*, pp. 2102-2107.

[26] Singh, S., 1996. A Survey of Automation in Excavation. *Journal of Mining and Material Processing Institute of Japan*.

[27] Singh, S., and Kelly, A., 1996. Robot Planning in the Space of Feasible Actions: Two Examples. in *Proceedings of International Conference on Robotics and Automation*, pp. 3309-3316.

[28] Song, B., and Koivo, A., 1995. Neural Adaptive Control of Excavators. *in Proceedings of International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 162-167.

[29] Taha, H., 1987. *Operations Research: An Introduction*. Collier Macmillan, New York.