Parameterized Scripts for Motion Planning

Patrick Rowe and Anthony Stentz
Robotics Institute
Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

We present an approach for real time planning and execution of the motions of complicated robotic systems. The approach is motivated by the observation that a robot's task can be described as a series of simple steps, or a script. The script is a general template which encodes knowledge for a class of tasks and is fitted to a specific instance of a task. The script receives information about its environment in the form of parameters, which it uses to bind variables in the template and allows it to deal with the current task conditions. Changes or variations in the robot's environment can be easily handled with this parameterized script approach. New tasks for the robot to perform can be added in the form of subscripts, which could handle exceptional cases. We apply this approach to the task of autonomous excavation, and demonstrate its validity on an actual hydraulic excavator. We obtain good results, with the autonomous system approaching the performance of an expert human operator.

1.0 Introduction

As robotic machines become more advanced in order to solve more challenging problems, they themselves become increasingly more complicated. An example of this is a class of robots known as mobile manipulators, which consist of a multi-link manipulator that is mounted on a mobile base. Applications for such robots include autonomous exploration, inspection, mining, construction, and excavation [1]. These robots tend to have a high number of degrees of freedom. This makes optimal path planning in real time difficult because of the high dimensionality of the robot's configuration or state spaces. Trajectory generation can also present a problem for these types of robots because it requires knowledge of the dynamic characteristics of the robot, particularly the velocity and acceleration limits of each degree of freedom [2]. Changing payloads of the robot's end effector, for example loading large rocks into the bucket of an excavator, can drastically change the dynamic characteristics of these robots. Furthermore, in some cases the best thing to do is to simply move from point A to point B as fast as possible regardless of the dynamics instead of following a specified trajectory.

Control of these robots is also a difficult problem. While there does exist a wealth of information on robot manipulator control strategies [3], many of which require accurate dynamic models, control for some larger mobile manipulators, or any manipulator in which dynamics is a large factor such as a hydraulic excavator, is a difficult problem. Accurate modelling of the actuators of an excavator's hydraulic system is difficult as well.

Fortunately, several observations about these types of robots make the problem of planning and executing their motions much simpler. The first is that these robots typically have been designed for a specific purpose or task, so the entire flexibility of motion is not used and does not have to be reasoned about in the motion planning. Furthermore, the tasks that these robots perform can usually be described as a series of simple steps. It can be argued that this is how humans perform complicated tasks, such as driving to work or cooking a meal, and is the key observation motivating the approach presented here.

The idea of scripts as a way to store knowledge is not new to the fields of Artificial Intelligence and robotics. Scripts have been used to capture common-sense knowledge about everyday situations with the intent to reason about actions, predict future events, and answer questions [4]. These scripts are "parameterized" in the sense that script variables are bound to objects in the world, however their use of parameters is different from the approach presented here, where parameters are calculated for the purpose of controlling a robot.

The idea of using procedures to reason about and construct plans has been presented in planners such as the Procedural Reasoning System [5]. The approach here, however, does not exist at that level of planning which aims to synthesize entire plans for performing a complex task given a clever representation of the world and world states [6]. One level down are entities such as SAUSAGES [7] which is described as "existing between planning and action." Its role is to take plans generated from higher level planners and execute them on an actual robot by controlling its subsystems. The parameterized script approach to motion planning presented here can be considered to be at an even lower level, almost to the point of robot control. It uses the idea of scripts and procedures to control individual degrees of freedom, such as the joints of a manipulator, in order to perform a given task. This certainly doesn't limit the types of tasks that can be done with parameterized scripts, as they can become arbitrary complex and can be built upon by adding more scripts to do different tasks.

Parameterized scripts are a way of encapsulating high level expert human knowledge and heuristics about how to do a given task, while still being able to deal with environmental variables by modifying the details of the procedure. The structure of the script's steps encodes procedural

knowledge of how the task is to be performed. For example, a segment of a general script for a manipulator may look something like...*move joint 0 up, then move joint 1 down, etc.* In this example, knowledge of the joint sequence and the direction of joint motion is captured. This script structure is unchanging, and usually requires human expertise and study of the problem to construct it.

For these complex tasks, which typically involves interacting with and manipulating its environment, just blindly following a set sequence of steps, as with teach-playback systems, will not work. They require information about their environment in order to do something useful, which is provided in the form of *script parameters.*

Script parameters fill in the details in the script steps. They describe where objects are located in the environment, for example, or provide goals for the robot to achieve. The parameters can come from a variety of sources, from data files to other software modules. The script itself may do some other internal computations on the parameters it receives to convert them to a form that is more useful to the script. The script parameters can define the commands which are sent to the robot, the events which cause a transition from step to step, or a signal which informs the script it is time to handle an exceptional case. The example manipulator script may now read... *move joint 0 up to 20 deg., when joint 0 passes 10 deg., then move joint 1 down to 0 deg., etc.*

Apart from simplifying the motion planning, the parameterized script approach has the additional advantage that it does away with the need for complicated robot control algorithms for trajectory tracking because explicit trajectories are not constructed. Rather, the only control that is needed is low level control of the individual robot degrees of freedom.

2.0 Autonomous mass excavation

We apply the parameterized script approach to the real world task of autonomous mass excavation. Most of the research on autonomous excavation has focused on digging without much attention given to the completion of the rest of the task. The parameterized script algorithm implemented in this paper is concerned with the excavator's motions after digging a bucket a soil. A survey of autonomous excavation systems is given in [8].

One common trend is the use of human expertise in constructing the digging algorithms because it has been found in such an unpredictable domain traditional control algorithms become impractical. For example, in [9], human knowledge is used for constructing the finite state machines for various digging actions, and for defining fuzzy logic rules to compute the excavator's actions in each state. The parameterized script approach also uses expert human knowledge to construct the script's steps as well as to decide the relevant script parameters for transitioning between steps and computing the robot's commands.

Figure 1 shows a hydraulic excavator in a mass excavation setting. The excavator consists of a 4 DOF manipulator which is mounted on a tracked mobile base providing 2 additional degrees of freedom. Mass excavation involves removing large amounts of earth and loading it into trucks which haul it away. The excavation task proceeds rapidly with good excavator operators loading several hundred trucks per day.

FIGURE 1: A mass excavation scenario

The task of mass excavation is a good candidate for planning and executing motions using parameterized scripts. The excavator performs essentially the same series of moves each time to load a truck. Each truck is parked in the same general location. The earth to be removed is located in the same area, typically directly in front of the excavator's tracks. There are usually no obstacles in the workspace, and the appearance of an obstacle which could impede the excavator is a very rare occurrence.

Things do change slightly in the excavator's environment, however. For example, there could be slight variations in the pose and dimensions of each truck to be loaded. The terrain of the earth that is being removed is obviously changing. The earth cannot always be dumped in the same place in the truck, nor can it always be dug from the same location in the ground. While the overall sequence of steps of the task remains the same, the script's low level details, or script parameters, do change.

There may also be times when action must be taken which may not be considered part of a "normal" truck loading procedure. For example, if the earth is loaded too high in the truck bed it may need to be tamped down with the bucket to avoid spilling over the sides. This is a special event because it does not happen on each bucket load, rather only when the truck is full, and it may not be needed in all cases. Of course, safety is a large concern when dealing with such a machine, so the parameterized script approach must also be able to deal with an obstacle which could get in the way of the excavator.

3.0 Mass excavation parameterized script

The art of designing a parameterized script comes in determining how best to express a desired task as a series of steps, and in the proper choice of the parameters that fully describe the world in which the robot will perform its task. This information could come from interviews with human experts along with a thorough study of the task. For example, in this case of mass excavation, expert human excavator operators were observed and asked how they would load a truck. Their sequence of steps ultimately became the truck loading parameterized script.

The structure of the truck loading script was designed with a number of considerations in mind. These include:

- loading a truck as efficiently as possible in normal working conditions.
- taking hydraulic flow coupling and hydraulic system power limits into account. For example, the hydraulic pumps can only deliver a finite amount of power, and they also power more than one joint, so it is usually a bad idea to move both coupled joints at the same time.
- safely avoiding all known obstacles in the workspace. These include the truck and the unexcavated ground.
- having minimal spillage of earth around the truck during dumping.

3.1 External parameters

Figure 2 shows the software architecture for the truck loading parameterized script. The script receives external parameters from a number of outside perceptual modules. These modules receive perceptual information, from a laser rangefinder for example, and perform some useful function involving analysis of the scene into symbolic or geometric parameters. The perceptual modules then output their specific parameters to the script, which provide it with the information that it needs about its world. For example, the Truck Recognizer module performs the task of locating and measuring the truck to be loaded. The external parameters that it sends to the script are the coordinates of the four corners of the truck bed and the truck's dimensions. Other perceptual modules decide where to dig based on the current shape of the terrain, and where to dump based on how the truck is currently loaded. These external parameters change as the environment changes.

3.2 Internal parameters

Once the script has received all the external information that it needs, it may perform other calculations to convert the information to a useful form for the script. This information is referred to as internal parameters. Both the external and internal parameters take the form of numerical values which are specific to the script steps. There are two ways internal parameters are computed, geometrically and dynamically.

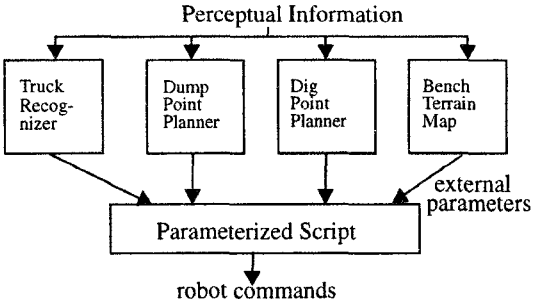


FIGURE 2: Motion planner architecture

3.2.1 Geometric parameters

Geometric calculations use the geometry of the environment and the kinematics of the robot to compute the necessary script parameters. For example, Figure 3 shows one of these parameters, the joint angle which is required to raise the bucket safely above the truck in order to avoid a collision. This computation requires knowledge about the dimensions and location of the truck, knowledge of its own link lengths, and perhaps other internal parameters that have already been computed such as the other joint angles at this point in the task. Other internal script parameters which are computed in this way are the angles which will keep the soil captured in the bucket and dump the earth in the truck.

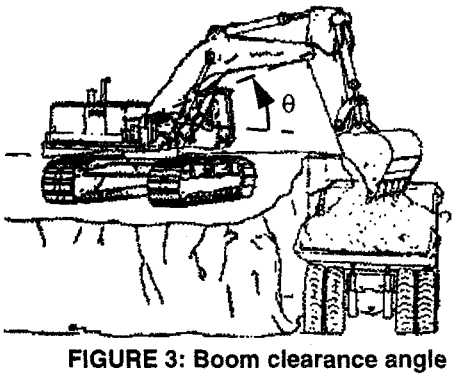


FIGURE 3: Boom clearance angle

3.2.2 Dynamic parameters

Dynamic parameter calculations use simple knowledge of the velocities and accelerations of the robot's joints to predict how long it will take to move a joint from a start angle to a goal angle. While the same information could be used for standard trajectory generation, it is not required with this approach. Using this knowledge, the joints may be better coordinated to achieve more optimal performance during task execution. For example, immediately after digging, the machine does not have to wait until the bucket has

raised to the clearance angle, computed in the previous section, before swinging to the truck. A better maneuver would move both joints at the same time taking advantage of the free space between the digging area and the truck, keeping in mind that there is the danger of swinging too fast before the bucket has fully raised. Knowledge of both joint's velocity and acceleration limits provides a way to safely couple the two. Dynamic knowledge can also help overcome system latencies. On an excavator, there are quite significant delays (on the order of 0.5 seconds) between when a joint command is issued and when the joint actually begins to move, primarily due to the time it takes to open the large hydraulic valves and allow hydraulic fluid to flow to the cylinders. By measuring these delays, the commands can be sent out ahead of time so that the joint moves at the right place in the task. For example, it is a good idea to begin opening the bucket while the bucket is moving over the truck. By anticipating the bucket joint delay, the command to open the bucket can be issued at the right point during execution so it begins to open just as it reaches the truck.

3.3 Truck Loading Script

Figure 4 shows the joint enumeration of an excavator used in the truck loading script. Figure 5 shows the full script that was developed for the truck loading task. Each circle represents the script step. Directly below each circle is the position step command, which are script parameters either provided externally or computed internally, that is sent to the machine. Above each link is the event, also a script parameter, which triggers a transition to the next step. A brief description of each parameter is shown at the bottom of the figure. For this particular task, each joint follows its own script, thus the trajectory of the end-effector emerges from the combined motions of the joints. Therefore, each truck loading pass could be slightly different from the last one. Of course, scripts could explicitly encode joint coupling as well. For example, a step in the script may compute simultaneous stick and bucket commands as a function of some partial trajectory.

All script step transitions are event based rather than time based. This offers a notion of closing a larger loop around the entire planning process and providing a layer of protection. If something should go wrong, such as a joint should fail, then the script will fail because a certain event has not triggered the transition to a new state, and the excavator will come to a stop.

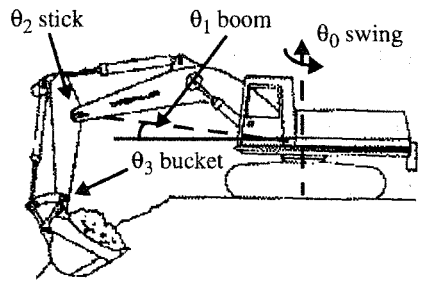
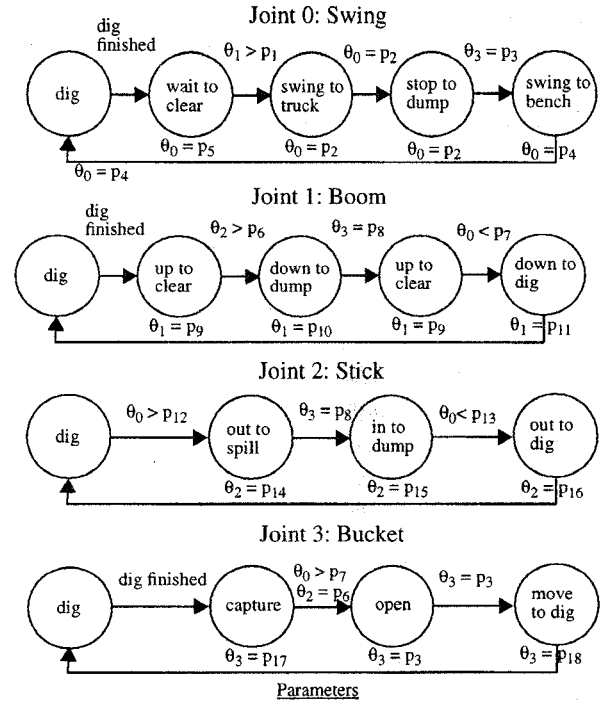


FIGURE 4: The excavator's manipulator



Parameters

p1: boom angle to begin swing
p2: swing dump angle
p3: bucket dump angle
p4: swing dig angle
p5: current swing angle
p6: stick angle which places bucket over the truck
p7: swing angle at truck rear
p8: bucket angle when mouth is vertical
p9: boom clearance angle
p10:boom dump angle
p11: boom dig angle
p12: swing angle to begin sticking out
p13: swing angle to begin sticking in
p14: stick start dump angle
p15: stick end dump angle
p16: stick dig angle
p17: bucket capture angle
p18: bucket dig angle

FIGURE 5: Truck loading parameterized script

4.0 Adding additional capabilities

New robot capabilities and behaviors can be easily added with the parameterized script approach as shown in Figure 6. Examples of new capabilities in a mass excavation scenario could include tamping the soil in the truck, cleaning up soil that has spilled around the loading area, and moving around obstacles that are in the workspace. This last capability, collision detection and avoidance, is described here.

New capabilities are added onto the main parameterized script in the form of parameterized subscripts, which con-

tain their own sequence of steps, required inputs, and internal parameter calculations to compute transition events and robot commands. Each subscript consists of two parts, a perceptual piece and a procedural piece. Like the external modules in Figure 2 which send external parameters to the main motion script, the perceptual part of the subscript transforms perceptual information into a signal which informs the main script that it is necessary to take another course of action. This procedural part of the subscript is then invoked, which directly controls the robot until it has finished its subtask, and then returns control to the main script. This part of the subscript may require information from its perceptual half, as well as certain parameters from the main script.

4.1 Collision Detection

The perceptual part of the collision avoidance subscript uses information about the current state of the robot, (joint positions and velocities) to predict where it will move so many seconds into the future. It acquires sensor data in the area that the excavator's manipulator will travel and determines if there will be an intersection between the excavator and anything that it is in its way. The perceptual information must be sensed far enough ahead of the robot so it has the distance to stop if a collision is detected. If a collision is detected, it immediately informs the truck loading script. This portion of the subscript runs concurrently with the main truck loading script, which continuously monitors for any inputs from the subscripts during execution

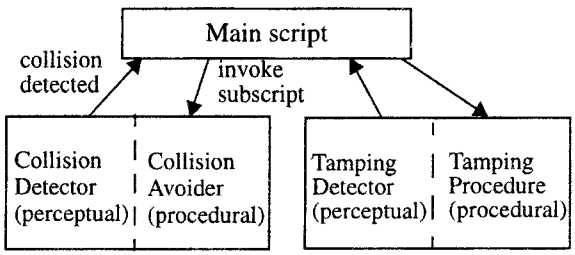


FIGURE 6: Adding capabilities with subscripts

4.2 Collision Avoidance

The procedural part of the collision avoidance subscript contains a specific algorithm which is designed to maneuver the excavator around an obstacle. When the collision detector has sent a signal to the truck loading script, the first course of action is to bring the excavator to a stop. More sensor data is then acquired about its environment before planning what to do next.

There are only a few courses of action that the excavator can take to get around an obstacle. This usually involves first moving the stick, for example to tuck in the bucket so there is less extent to the manipulator, then raising the boom to a height so that the implements clear the top of the obstacle, swinging the implements past the obstacle, and finally lowering the boom and moving the stick to their original positions. This is the sequence of actions that the collision avoider subscript takes to safely maneuver around the obstacle.

5.0 Results

The parameterized script approach described above was implemented and tested on a real 25 ton hydraulic excavator testbed shown in Figure 7. The excavator was retrofitted with resolvers and PD position controllers for each joint. The tracks are not actuated. The excavator's test environment is set up to emulate a mass excavation scenario. It sits atop a high berm with an adequate amount of soil to dig in front of it. A 15 ton dump truck is provided for loading.

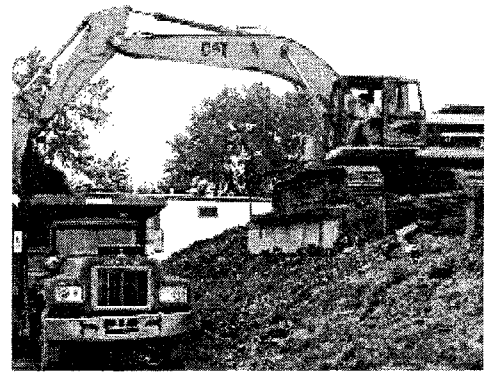


FIGURE 7: Mass excavation test site

The excavator came equipped with an algorithm, known as AutoDig, which uses pressure feedback to load the bucket with soil [10]. The rest of the excavator's motions were controlled using the parameterized script approach.

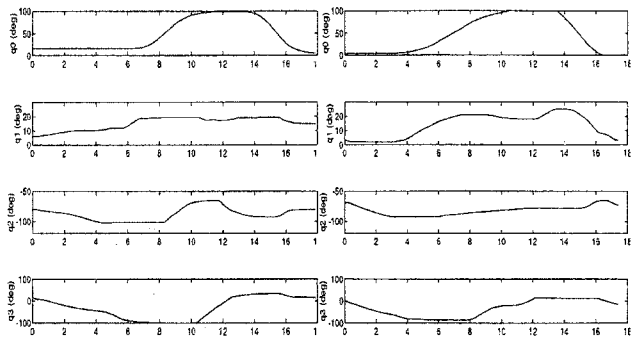
The external parameters of the truck's location and dimensions, as well as where to dig, were provided through a laser designator as used in surveying which is mounted on the excavator's cab. In future systems, these parameters will be provided through perceptual software modules which will use sensor data to calculate these parameters.

We asked a human expert excavator operator to load the truck at out test site. The times to dig a bucket of dirt, dump it in the truck, and return to the soil face were recorded. Typical dig-dump cycle times were between 15 to 16 seconds, and the human expert could fully load the truck in approximately 2 minutes.

The parameterized script approach was then put to the test. The results were very good. It consistently averaged 17 to 19 seconds for a dig-dump cycle. The AutoDig algorithm dug slightly more heaped buckets of soil than the human expert did, therefore the throughput, or how many tons of earth could be moved in a given time, equalled and

sometimes exceeded that of the human expert operator. Figure 8 shows a comparison of the joint motions between the human operator and the autonomous system for one dig-dump cycle. Differences in the chosen dig location, digging styles, soil conditions, and location and orientation of the truck can be seen in the slight differences in the joint motions between the two, however the general motion trends are similar.

The script also gave good results in other less quantitative aspects. First and foremost, the excavator safely avoided colliding with anything in its workspace. Although soil conditions did play some part, there was in general little or no spillage of the soil on the ground when loading the truck. Different truck positions were tried all with good results.



a) Autonomous system b) Human expert

FIGURE 8: Joint motions for human and robot

The collision avoidance subscript was also tested. For these tests, the truck loading script received a pre-programmed signal from the collision detector at some point during its motion to the truck, along with the dimensions of an imaginary obstacle that was in its way. The truck loading script took the correct actions by stopping the machine before it collided with the obstacle, and safely moving around the obstacle. Once the collision avoidance subscript believed that it was safely past the obstacle, the truck loading script took control again and continued on its way to dump the soil in the truck.

6.0 Conclusions and future work

We have presented an approach to motion planning and execution for complicated robotic systems that is based on the observation that many tasks can be described as a series of simple steps. This approach can be applied to any robotic system whose task can be expressed as a series of steps. For other tasks, such as robotic welding, the parameterized script approach may not be the best method because the robot's motions are a function of the measured error. We have applied this technique to the real world application of autonomous mass excavation with extremely promising results. The approach is very simple conceptually and easy to implement. It greatly simplifies the difficult problems of planning and control for these complex machines. This approach also provides a way to add more advanced capabilities and behaviors to the robot by simply adding on other subscripts which fully describes what the robot is to do.

The immediate future goals of the autonomous mass excavation system is to add visual sensors to the excavator and to integrate the different external perceptual modules as we strive for greater autonomy.

This work on autonomous excavation using the parameterized script paradigm has advanced by adding an optimization routine to the internal script parameter calculation. Data about each set of script parameters (one for each bucket load), and the results of the excavator's actions, such as overall cycle time, are recorded. This information is used to adjust the calculated script parameters for faster loading times.

Acknowledgments

The authors would like to thank all the members of the Autonomous Loading System team for their help and support of this project.

7.0 References

[1] Hootsmans N., 1992. *The Motion Control of Manipulators on Mobile Vehicles.* Ph.D. Thesis, Department of Mechanical Engineering, MIT, Cambridge, MA.

[2] Craig J., 1986. *Introduction to Robotics Mechanics and Control.* Addison-Wesley.

[3] Lewis F., Abdallah C., Dawson D., 1993. *Control of Robot Manipulators.* Macmillan.

[4] Rich E., Knight K., 1983. *Artificial Intelligence.* McGraw-Hill.

[5] Georgeff M., Lansky A., 1986. Procedural Knowledge. in *Proc. IEEE Special Issue on Knowledge Representation,* pp. 1383-1398.

[6] Georgeff M., 1987. Planning. Technical Report. SRI International.

[7] Gowdy J., 1994. SAUSAGES: Between planning and action. Technical Report CMU-RI-TR-94-32, Robotics Institute, Carnegie Mellon.

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

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

[10] Rocke, D., 1995. Automatic Excavation Control System and Method. U.S. Patent No. 05446980.