Obstacle detection and safeguarding for a high-speed autonomous hydraulic excavator

Chris Leger[a], Patrick Rowe, John Bares,

Scott Boehmke, and Anthony Stentz

The Robotics Institute, Carnegie Mellon University

ABSTRACT

Hydraulic excavators are large, powerful machines which are often operated in high-production settings. Successful automation of excavators for mass excavation tasks require safeguarding algorithms which do not negatively impact productivity. We present a two-level sensor-based safeguarding approach which utilizes obstacle detection to prevent collisions and motion detection to halt operation when unanticipated vehicles or people approach the excavator.

Keywords: Obstacle detection, excavation, safeguarding, motion detection.

1. INTRODUCTION

In a typical mass excavation application, a loading machine digs material from a face and dumps the material into a truck, with a throughput of hundreds of trucks per day. The process is repetitive and continues day and night and in most weather conditions. The repetitive nature and potential cost benefit of even minor performance improvements make the automation of mass excavation attractive.

Mass excavation using hydraulic excavators is typically accomplished by eroding a strip of soil (known as the "bench") underneath the excavator. The excavator backs up as it erodes the face of the bench, and dump trucks arrive for loading next to the excavator. Depending on the relative sizes of the excavator and dump trucks, the excavator may load between 2 and 10 buckets of soil to fill each truck. During each loading cycle, the excavator digs a bucket of soil from the bench, swings over the truck, dumps the soil, and swings back to the dig face; each cycle takes between 15 and 30 seconds, varying with dig and dump locations and soil conditions.

The robot used in this work is a commercial hydraulic excavator that has been modified for autonomous operation[1] (Figure 1). The excavator can be considered to be a four degree of freedom manipulator mounted on a mobile base. The first degree of freedom is the swing joint, which rotates the entire cab about a vertical axis. The remaining degrees of freedom are the boom, stick, and bucket, respectively, all of which have parallel horizontal axes. (The boom, stick, and bucket links are collectively known as the *implements*.) All actuators on the machine are hydraulic. The excavator's conventional valve-based manual controls have been replaced by a microprocessor-based controller which may accept commands from a set of instrumented joysticks or from a separate onboard computer. The onboard computer executes all perception and control software and consists of several MIPS-based processor boards on a common backplane. Two scanning rangefinders provide perceptual information to the computer during autonomous operation. Each rangefinder's beam is directed by two actuators: the tilt motor spins a mirror at several tens of cycles per second, thus sweeping the beam through a vertical plane, and the pan motor changes the azimuth of the sensor head, tilt motor, and mirror (Figure 2).

One of the key challenges in automating mass excavation is ensuring safety for workers and equipment. When swinging between the dig and dump locations, the excavator's bucket velocity may approach 10 m/s and the swing axis may require 60 or more degrees to stop during an emergency. Additionally, rocks and soil may be dislodged from the bucket, posing a potential hazard to nearby equipment and personnel. It is also necessary to prevent the excavator's implements from swinging over workers or equipment: a ruptured hydraulic line may cause the implements to sag or drop uncontrollably. (The dump truck bed is the only exception to this rule, since the excavator must obviously swing

a. The first author may be contacted via email at blah@cmu.edu, and is located on the WWW at http://www.frc.ri.cmu.edu/~blah/.

Figure 1: The excavator testbed.

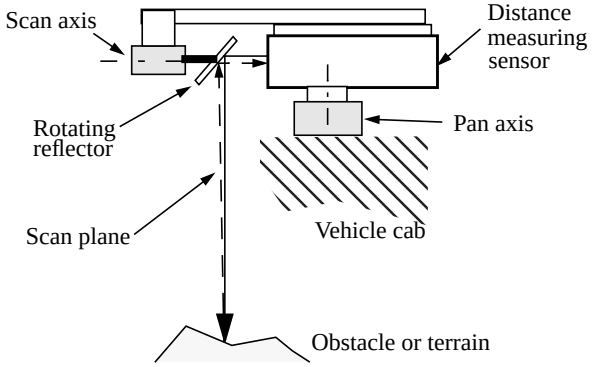


Figure 2: Range sensor configuration

the bucket over the truck bed in order to load the truck).

In a mass excavation setting, obstacles and unanticipated vehicles or workers are rare. Taken with the goal of high productivity, this implies that the motion planner should be concerned with optimizing productivity while treating unanticipated obstacles, vehicles, and personnel as exceptional conditions. Thus, a reactive approach to obstacle detection and safeguarding is more appropriate than deliberative planning to avoid obstacles. We have developed a two-tiered reactive safety system. The first tier utilizes perceptual data and a fast dynamic simulation of the excavator to predict future machine states and detect collisions with the environment. The second tier uses range data to detect unanticipated moving objects around the excavator, allowing the excavator to halt motion before a person or vehicle gets close enough to travel under the excavator's implements or become endangered by falling rocks or soil.

The problems addressed in this work differ from previous work in obstacle detection for manipulators in that our manipulator (the excavator) requires significant time and distance to come to a halt. Additionally, since productivity is a primary concern and unanticipated obstacles are infrequent, our planning algorithms must be concerned primarily with optimizing throughput rather than finding paths which give obstacles a wide berth. Prior work in manipulator obstacle avoidance has focused on planning paths around obstacles[2, 3] in known environments, rather than halting when an occasional obstacle is sensed. Work has been done in the area of safeguarding for excavators, but has focused on low-speed operation in which the stopping time and distance are negligible.[4, 5] We find that the needs of our system are closer to those found in obstacle detection for high-speed mobile robots[6] than to manipulator obstacle avoidance.

2. OBSTACLE DETECTION

The obstacle detection algorithm is tasked with ensuring that the excavator does not execute any motions which cause a collision with objects in the environment. Since the motion planner must know where the truck is in order to load it, the planner will not command a trajectory that causes the excavator to collide with the truck. However, the motion planner does not have knowledge of other obstacles in the environment and relies upon the obstacle detector to inform it of any potential collisions. Briefly, the obstacle detector algorithm works as follows:

After the motion planner has generated a set of commands (typically swinging from the dig location to the dump location or vice versa), it scans the range sensors through the region of anticipated motion. The obstacle detector then builds an elevation map from the range data. Every tenth of a second, the obstacle detector simulates the motion of the excavator over the next few seconds (starting with the most recent sensed vehicle state). At each simulation time step, the predicted pose of the machine is checked against the elevation map for collisions. An emergency stop command is immediately issued if any collision is predicted. The remainder of this section describes each component of the obstacle detector in detail.

2.1 Constructing elevation maps

Depending on which way the excavator will swing, the motion planner commands one of the sensors to pan ahead and gather data in the region through which the implements will move. The resulting range data is processed to create a grid-based elevation map. The map is centered on the excavator and is built before each swing motion. Each grid cell contains the height of the highest data point that falls within the cell; thus, the elevation map provides a conservative estimate of the height of the terrain at any point nearby the excavator. The grid resolution is not very critical, and is limited by two factors: the number of intersection checks required for collision detection increases with the number of grid cells, and the accuracy of the elevation map representation decreases as cell size increases.

It is necessary to ensure that the implements do not travel through a region of space which was occluded during scanning, since there may be objects in the occluded regions. For example, if there is a tall object close to the excavator and a shorter one beyond it, the boom may be able to clear the close object while the bucket may not clear the farther one. To prevent collisions with occluded obstacles, we interpolate between adjacent points within a scanline if the points' corresponding grid cells differ by more than one in either coordinate. The interpolated points are then processed just as the original points were: if a point's height is greater than the height of the grid cell in which it lies, then the cell's height is set to the point's height. Figure 3 shows how points are interpolated. The intersections of the line between the sensed points and each grid cell boundary are computed, and the height at these intersections is determined. The interpolated height is then used to update the height of the grid cells neighboring the point.

2.2 State prediction

During each planner cycle, the obstacle detector reads the excavator's joint positions and velocities and begins a simulation to predict the path of the implements in the next few seconds. The motion planner generates vehicle commands from vehicle states; thus, during each simulation step, the obstacle detector queries the motion planner for a set of commands based on the simulated vehicle state. A simplified dynamic model then predicts the next vehicle state. The obstacle detector must predict the vehicle state far enough in the future so that potential collisions can be detected far enough in advance to allow the machine to come to a halt. Since the stopping time of the excavator varies with velocity, the necessary feedforward time varies as well.

The simulation step size should be chosen so that when simulating swing motions at the maximum angular velocity, the excavator's bucket will move no more than the width of one grid cell between simulation time steps. The maximum distance between the bucket and the swing axis is roughly 10m. During loading operations, swing velocity may reach 50 deg/s; this yields a linear velocity of 8.7 m/s at the bucket tip. Using our grid cell size of 0.6m, the time step must be no more than 68ms. This is quite reasonable from a computation standpoint, as simulation time step

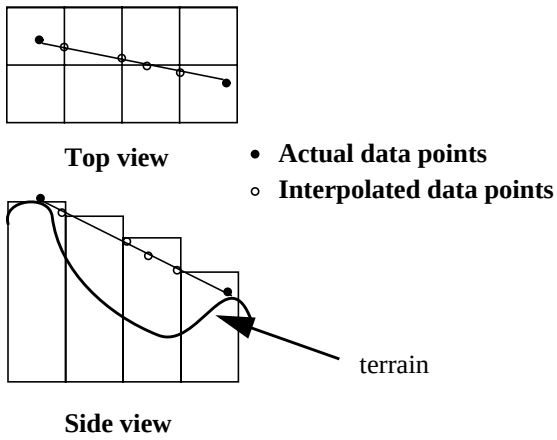


Figure 3: When neighboring points in a scanline do not fall in adjacent cells in the terrain map, interpolation is performed. Interpolated points are computed at the intersections of the line between real data points with the boundaries between grid cells.

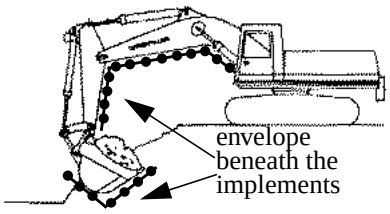


Figure 4: For collision checking, the excavator's implements are modeled by a set of points on the underside of each link. The height of each point is checked against an elevation map acquired by the range sensor.

requires less than a tenth of a millisecond; the total computation time required for simulation during each 100ms cycle is less than 2ms.

2.3 Detecting intersections

At each simulation time step, a simple geometric representation of the excavator's implements is computed based on the predicted joint angles. This geometric model consists of a series of points along the underside of the implements (Figure 4). Each point is checked for collision with any obstacles by computing the grid cell in which the point lies and then comparing the point's height to the height stored in the cell. If the point's height is less than the cell's height plus a safety buffer, the motion planner is informed of a potential collision and an emergency stop command is issued. The location of the points on each link of the implements depends on the swing direction and the implement joint angles. If the excavator is swinging to the left, an offset is added to the points so that they lie on the left edge of the bottom side of the implements; a similar adjustment is made if the excavator is swinging right. Additionally, the stick and bucket may have different surfaces towards the ground depending on the implement joint angles, so the appropriate surface must be used when computing the points.

2.4 Scanner lookahead

Predicting collisions requires that the elevation map contains data for the region through which the excavator will be moving. To ensure this, the scanner must begin panning far enough in advance so that at each time step, the scanner has panned farther than the lookahead angle (which is determined by the stopping distance and obstacle detection latency). There are two scanner pan angles involved in computing lookahead: the lookahead angle, which is the maximum angle that the scanner needs to be panned to during a given swing motion; and the trigger angle, which is the pan angle the scanner must reach before the swing motion begins. These angles are computed off-line for a range of swing step commands and are stored in a lookup table. After the motion planner determines the swing command, it determines the lookahead and trigger angles from the lookup table. The scanner begins panning to the lookahead angle during the dig (or dump) operation so that the trigger angle is reached before digging (or dumping) is complete.

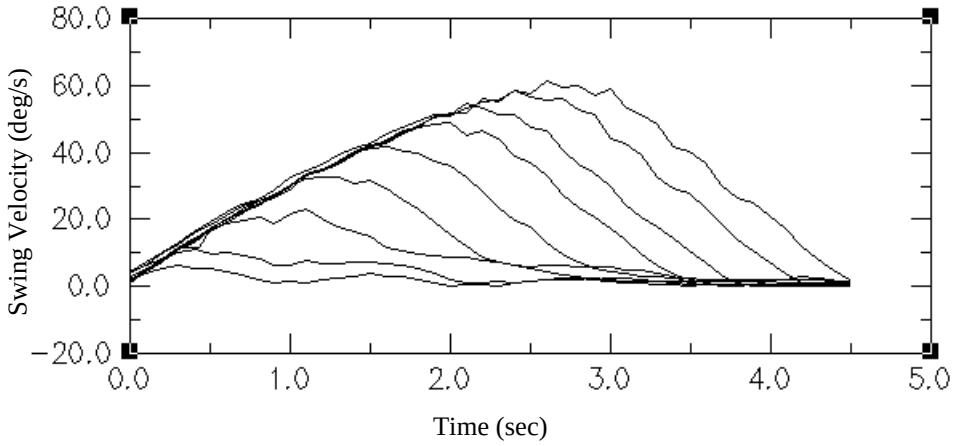
The lookup table is built from measured machine responses to varying swing step commands. First, the braking time and angle is measured for a range of swing velocities. The lookahead angle is given by the braking angle plus the computing latency multiplied by the swing velocity. (The braking time measured in this step is used to determine the number of simulation time steps, as described in Section 2.2.) Next, a range swing step commands are executed and the velocity profiles are recorded (Figure 5a). For each step command, the required lookahead angle at each point along the swing velocity profile is then computed. The scanner pan command for a given step command is the maximum lookahead angle over the velocity profile. Finally, the scanner's response curve to the commanded lookahead angle is computed, and then shifted leftward until the scanner pan angle is greater than or equal to the required lookahead angle at each point along the velocity profile (Figure 5b). The y-intercept of the shifted response curve is the trigger value, the value which the scanner's pan axis must reach before the swing motion can begin to ensure adequate lookahead.

2.5 Validation

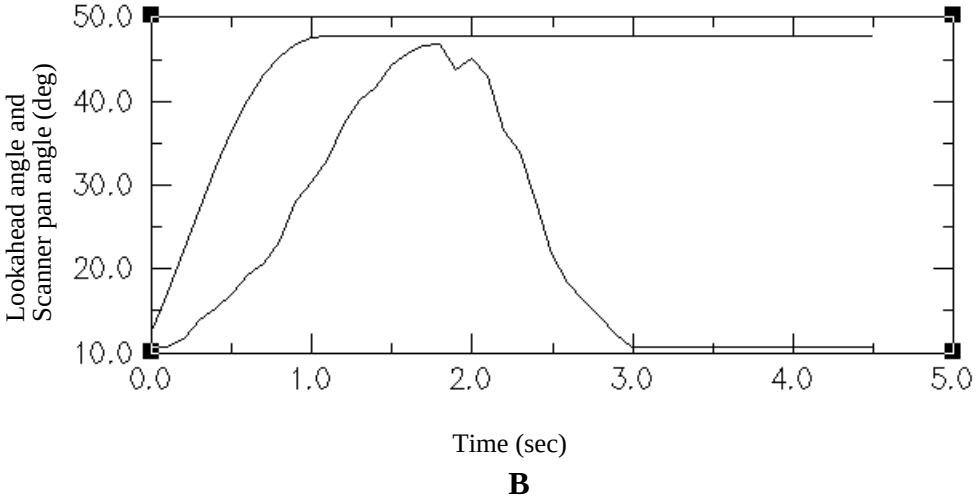
Before testing the obstacle detection algorithm on the excavator, we performed testing in simulation over a wide range of scenarios. Our simulator provides the control software with an interface identical to the real excavator's and allows the excavator's complete software system to be tested in simulation. In addition to simulating the excavator's kinematic and dynamics, the simulator models soil removal, interaction, and settling and can generate range data from simulated scanners with the same specifications as the real scanners.

We tested the obstacle detection algorithm with varying obstacle sizes and positions, and varying implement poses. These tests ensured robustness in three areas:

- Geometry - are the geometric models of the excavator and terrain correct?
- Prediction - does the algorithm predict vehicle states far enough into the future to stop before collisions? Are vehicle states predicted correctly?
- Perception - can obstacles be reliably sensed?



A



B

Figure 5: The swing velocity as a function of time is shown for a series of step commands to the swing joint in **A** above. The step commands ranged from 10 to 160 degrees. **B** shows the required lookahead angle (lower curve) and final pan response curve (upper curve) computed for a swing step command of 100 degrees. (These curves are computed for each of the velocity profiles in **A**, but only one set is shown for clarity.)

It is desirable to distribute tests throughout the space of operating conditions. However, given the simulation time required for each test (one to two minutes) and the large dimensionality of the operating conditions (obstacle height and width, obstacle position in the plane, and three implement joints angles), we decided to partition the tests to independently verify robustness in the three areas above. Varying obstacle range and size tests perceptual robustness; varying the implement angles and obstacle position verifies geometric robustness; and varying the swing angle at which the obstacle is located tests prediction. With the final version of the software there were no false positives or false negatives and no collisions. All potential collisions were detected soon enough to allow the excavator to stop before reaching the obstacle, and the excavator did not stop when a collision would not have occurred.

While the simulated tests verified the integrity of the algorithm, the simulation did not account for several factors present on the real excavator: noise in sensor data, command and data latencies, processing time, and calibration. Most

of the noise in our range data is additive and on the order of several centimeters; this does not impact obstacle detection. More problematic is the occasional spurious data point, which can potentially cause false positives. Fortunately, these points are rare enough that they can be filtered out by discarding any data points with a large difference in range from its neighbors. At the time of testing, the implement and range sensor calibrations yielded discrepancies in bucket location of 20-30 cm. To account for this, we added a safety margin of 50cm above obstacles--the excavator was halted if any part of the implements was predicted to pass less than 50cm above an object. (Since the time of testing, the calibration has been improved to yield errors around 10cm.) We found that the computation required for the algorithm was fairly low, occupying only a small part of one CPU's processing time.

Tests on the real machine were performed with a cardboard, human-sized target as an obstacle. Due to the time-consuming nature of test site preparation when moving soil, we did not test obstacle detection while actually performing truck loading. Instead, we used a simplified motion planner which commanded the machine to swing back and forth between pre-selected dig and dump locations. The final version of the software was able to detect all potential obstacles and halt the excavator before a collision occurred. Due to the safety margin, the excavator stopped several times when it would have narrowly missed the obstacle. This is desirable from a safety standpoint, but we must ensure that the excavator is not halted unnecessarily when swinging the bucket over the truck bed before dumping. Since we now have a more accurate calibration, we anticipate reducing the safety margin to allow for better discrimination between potential collisions and near misses.

3. SAFEGUARDING

The obstacle detection algorithm presented in the previous section is sufficient to prevent collisions with stationary or slow-moving objects in the path of the excavator. However, as discussed in the introduction, it is desirable to halt the excavator's motion when a person or vehicle unexpectedly approaches the excavator. Like the obstacle detection algorithm, our safeguarding approach (called the Workspace Monitor or WSM) represents the environment with an elevation map. The WSM continually gathers range data, using the same data stream as other perception tasks. A temporary elevation map holds recent range data and is periodically compared with an accumulated elevation map. An increases in cell elevation can indicate the presence of a moving object; a decrease may indicate that the obstacle has changed position. This is the basis of the WSM algorithm; the details are described in the following sections.

3.1 Building elevation maps

The WSM builds and represents elevation maps in a slightly different manner than the obstacle detection algorithm. In addition to storing a height for each cell, a status flag indicates special properties of the cell. Table 1 shows the status flag values and their meanings. The temporary map is initialized so that each cell's status flag is EMPTY. As each scanline is received, the data points are checked against the temporary elevation map and interpolated points are computed if necessary. Each cell stores the heights of the highest two points to fall in the cell; the highest is discarded to minimize the impact of false sensor readings. Whenever a data point lies within a cell, the cell's status is set to SET (if the point was sensed) or INTERPOLATED (if the point was interpolated, and the cell was empty). Interpolation in the WSM serves a different than in obstacle detection: it allows moving objects to be detected in areas of the elevation map which were previously occluded. Interpolated points are used to set a cell's height only if the cell was previously marked as EMPTY, and a real data point will always override a cell height that is based on interpolated data.[b] Each time a temporary map cell changes status from EMPTY to SET or INTERPOLATED, it is added to a list of modified cells. This allows fast access to those cells that have been modified so that they can be merged in to the accumulated map.

3.2 Updating the accumulated elevation map

The accumulated elevation map is incrementally built from the modified cells of the temporary map. Each modified cell in the temporary map is compared with the corresponding cell in the accumulated map to detect elevation changes and to update the accumulated map height. While it would be possible to fill the entire temporary map before comparing it as a whole to the accumulated map, this would cause a significant latency in detecting moving objects. In-

b. This prevents occlusions from generating an interpolated cell height which is higher than a previous actual cell height, which could be mistaken for a height increase caused by a moving object.

stead, we frequently compare the accumulated map with the small portions of the temporary map that have recent data. This allows the latency to be significantly reduced: in our implementation, map updates occur every half-second while the period of the range sensor scanning motion is on the order of 15 seconds

The WSM updates the accumulated map using two procedures. `CheckCell` determines whether a cell in the temporary map should be considered a moving object, and `MergeCell` updates the height and status of an accumulated map cell based on the results of `CheckCell.` During each map update, `CheckCell` is called for those temporary map cells marked as NOT_VISITED. After all cells have been checked, `MergeCell` is called for all temporary map cells except those that are marked as SET or INTERPOLATED, which are changed to NOT_VISITED or I_NOT_VISITED respectively. This prevents cells that are in the middle of being scanned from being merged into the accumulated map, which is desirable since their height in uncertain. If no new range data falls within a cell before the next update, the cell will still be marked as NOT_VISITED or I_NOT_VISITED, and will thus be processed.

Pseudocode for `CheckCell` is shown in Figure 6. Due to sensing errors, it is not sufficient to simply check the change in height of a single map cell. For example, if the edge of an object lies close to a grid cell boundary, the sampled edge of the object may lie in either cell due to noise in range measurements. To avoid this sort of problem, neighboring SET cells in the accumulated map are checked to determine if there is a large height discontinuity. If a large discontinuity exists or if the temporary cell's height is close to a neighboring cell's height, then the height change is assumed to be caused by sampling errors. A similar problem may occur if there is a large height discontinuity in cells at the edge of the scanned area. In one scan, a cell at the border of the scanned area may only contain data from the low part of the cell, but in the next scan the higher part may be detected. We avoid false detections in this case by ignoring height changes if there are 2 or more empty neighboring cells[c]. Whenever an OBSTACLE cell is detected within the specified safety region around the machine, the motion planner is notified and halts motion. In our system, the safety region extends approximately 15 meters from the center of the excavator (5m beyond the excavator's maximum reach)

We tried two versions of MergeCell in our system. The first version, shown in Figure 7, always updated the height of cells in the accumulated map with height information from the temporary map. This version had problems with sampling errors at the long ranges (15-20m). Due to the sparseness of the range data, narrow objects could not be sensed reliably and would thus cause apparent elevation changes. One solution to this problem is to periodically take a slow, dense scan of the environment to build a reliable elevation map, and then use data collected during normal operation only for comparison, rather than updating the accumulated map. The version of MergeCell used for this is given in Figure 8. The accumulated map can be cleared and rebuilt during breaks in loading operations, so that a relatively recent accumulated map is available at all times.

Table 1: status flags

Flag	Meaning
EMPTY	No range data lies within the cell.
SET	One or more data points lie within the cell
INTERPOLATED	Only interpolated data points are in the cell
NOT_VISITED	No further data has been received since the last map update
I_NOT_VISITED	Same as NOT_VISITED, but for cells with interpolated data
CHANGED	An accumulated map cell's height has changed significantly
OBSTACLE	An accumulated map cell contains a moving object

c. This occurs only at the edge of the scanned region, which lies outside the operating area of the machine.

```
procedure checkCell(cell, tempCell):

if (tempCell.height - cell.height > heightThreshold) then
   numEmpty = 0;
   foreach neighbor n of cell
      if (n.status = SET) then
         if (n.height > tempCell.height - neighborHeightThreshold) or
            (n.height - cell.height > slopeThreshold) then
               tempCell.status = CHANGED;
               return;
         endif
      else if (n.status = EMPTY) then
         tempCell.status = CHANGED;
         numEmpty = numEmpty+1;
         if (numEmpty >= 2) then return;
      endif
   endfor

   cell.status = OBSTACLE;

end
```

Figure 6: Pseudocode for `CheckCell`.

In addition to detecting when a moving object approaches the machine, it is also important to detect when a moving object has moved away. This is accomplished by monitoring height decreases as well as increases. If a cell marked as CHANGED or OBSTACLE decreases significantly in height (in the case of the first MergeCell algorithm), or returns to a height close to its original height (for the second MergeCell algorithm), it is marked SET. When there are no longer any obstacles in the safety region, vehicle motion can safely resume.

3.3 Experiments

As with the obstacle detection algorithm, we first evaluated the WSM in simulation. The simulation scenario is shown in Figure 9. While the excavator was performing regular truck loading cycles, a human-sized object was interactively moved in the simulator. When the object was moved into the safety region from the area at the base of the dig face, it was reliably detected. When the object was moved along the side of the dump truck, it was not detected until it was approximately one meter away from the dump truck. This is due to the heuristic which ignores height changes if there is a large height discontinuity between neighboring grid cells.

When testing with the real excavator, we used a simplified motion planner instead of the full system (as was used for testing of the WSM in simulation). The simplified motion planner continuously swung the excavator between dig and dump locations, commanding scanner motions similar to those used during truck loading. Using the second version of the MergeCell procedure, the WSM was able to reliably stop the excavator when a person entered the safety area. We designated a 24m x 24m square centered on the excavator as the safety area. As in the simulated tests, a person moving next to a taller object (such as a dump truck) would not be detected until they moved far enough from the object (typically 1m) so that they would not be in an adjacent grid cell. While it is certainly possible for a determined individual to enter the swing zone of the excavator before the excavator halts, we found the WSM to provide a very useful level of safeguarding for the machine.

```
procedure mergeCell(tempCell, cell):

if (tempCell.status = NOT_VISITED) or
    (tempCell.status = I_NOT_VISITED) then
   if ((cell.status = OBSTACLE) or (cell.status = CHANGED)) and
      (tempCell.height < cell.height - heightThreshold) then
         if (cell.status = OBSTACLE) then notifyPlannerOfChange(cell);
         cell.status = SET;
   endif
else if (tempCell.status = CHANGED) then
   cell.status = CHANGED;
else if (tempCell.status = OBSTACLE) then
   cell.status = OBSTACLE;
   notifyPlannerOfChange(cell);
endif

cell.height = tempCell.height;

end
```

Figure 7. MergeCell pseudocode for continuous map updating.

---

4. DISCUSSION

4.1 Sensor requirements

In our system, obstacle detection imposes the strongest requirements on range sensors. Other perceptual tasks such as truck recognition and dig planning do not demand the spatial resolution and range accuracy that obstacle detection does. To robustly detect a child-sized obstacle at the maximum reach of the excavator, the range sensors should have an inter-scanline spacing of roughly 20cm[d] at 10m, or 1.15 degrees. To enable discrimination between obstacles and noise, 6 data points on an obstacle appear to be sufficient. Assuming a 1 meter high obstacle at 10m that is sensed only in a single scanline, this translates to an intra-scanline point spacing of 0.95 degrees. This equates to 376 samples per scanline. To achieve the desired inter-scanline spacing, we must account for the speed at which the scanner will be panned. If we can ensure that the scanner will not be panned at more than 30 degrees per second *including* the swing velocity of the excavator, then we can use a scanline rate of 26 scanlines/sec. This yields a point sampling rate 9.8KHz. In practice, our scanners have slightly higher point sampling and mirror rotation rates and are able to meet these requirements.

As noted in the description of the Workspace Monitor, our current sensors do not allow us to continually update the map due to the limited sampling density at long ranges. We feel that in order to reliably sense moving objects the size of small child at 15 to 20m, a point sampling of approximately 30 KHz and a mirror rotation rate of 50Hz will be necessary (assuming a total pan velocity of 30 deg/s and desired point spacing of 15cm). While point (unscanned) rangefinders with 30KHz and higher sample rates and sufficient accuracy are commercially available, they may not have adequate robustness to dust and fog, two conditions which are frequently encountered during mass excavation operations.

4.2 Geometric representations

The geometric representations we use for obstacle detection (points along the bottom of the implements, and an elevation map of the environment) are limited, but are appropriate for mass excavation tasks. Mass excavation takes place

---

d. This assumes a narrow rangefinder beam. A greater inter-scanline spacing is tolerable with larger beamwidth.

```
procedure mergeCell(cell, tempCell):

if (tempCell.status = NOT_VISITED) or
   (tempCell.status = I_NOT_VISITED) then
   if (tempCell.status = OBSTACLE) or (tempCell.status = CHANGED) then
      dHeight = tempCell.height - cell.ObstacleHeight;
      if (dHeight < heightThreshold/2) or
         ((tempCell.status = I_NOT_VISITED) and
          (dHeight < 0))) then
         if (cell.status = OBSTACLE) then notifyPlannerOfChange(cell);
         cell.status = SET;
      endif
   else if (cell.status = EMPTY) then
      cell.height = tempCell.height;
      cell.status = SET;
   endif
else if (tempCell.status = CHANGED) or (tempCell.status = OBSTACLE) then
   cell.status = tempCell.status;
   cell.obstacleHeight = tempCell.height;
   if (cell.status = OBSTACLE) then notifyPlannerOfChange(cell);
endif
```

Figure 8: MergeCell pseudocode for initial map building with continuous comparison.

---

outdoors and away from overhead structures. Since obstacles are part of or are resting on the ground, the grid-based elevation map is equivalent a set of bounding boxes around objects in the environment. Checking only the lower surface of the implements for collisions is also reasonable since the excavator will not pass *under* any objects in the environment. Other tasks may involve overhead obstacles; if this is the case, then a different intersection algorithm (such as determining whether any range points lie within a polyhedral model of the excavator) may be used. Given the low computational cost of our current system, we do not expect that a more complex geometric representation would render our general approach infeasible.

It may also be desirable to use a more complex representation for the WSM. Instead of storing a single height in a grid cell, a histogram of observed heights could be maintained. The points in a temporary grid cell could be compared to the histogram in the corresponding cell of the accumulated map, and significant departures from previous observations could be used to detect motion. This method may require an initialization phase, which could consist of taking a slow, dense scan (as we do now), or not detecting moving objects in a cell until a predetermined number of data points have been observed in the cell. In general, we have tried to avoid statistically-based approaches, since it is highly desirable to detect obstacles and moving objects with only 4 or 5 data points from the object. The excavator moves quickly enough and presents significant enough hazards that is unacceptable to require that an object be scanned several times in order to have sufficient evidence of its presences. The histogram-based approach may be feasible and yet sill allow some resistance to sampling errors, but requires further investigation.

4.3 Motion Estimation

One limitation of our current system is that it does not account for movement of the excavator's base. When the excavator drives a few feet to a new digging location, a new map must be built for workspace monitoring. Additionally, slippage of the tracks may occur during sudden starts and stops of swing motion. Inertial sensors and terrain map registration may be useful in both of these cases to ensure that range data is consistently converted to the same world coordinate system.

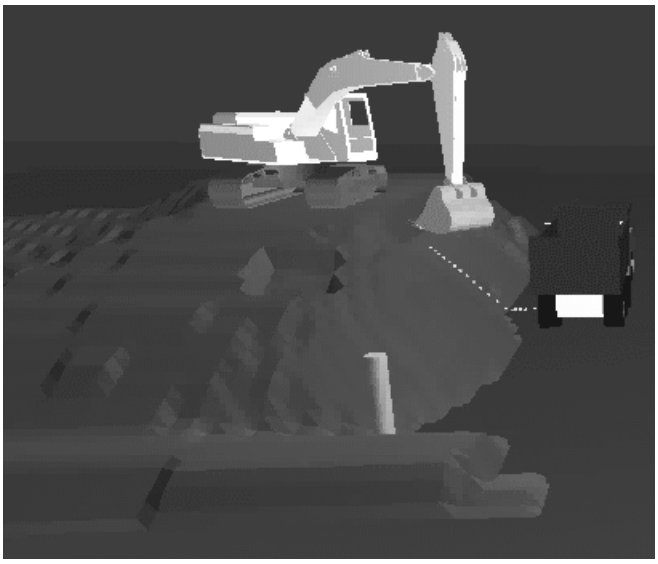


Figure 9: Simulation scenario used for testing the Workspace Monitor. The person-sized obstacle is the light object in the foreground.

5. CONCLUSION

We have presented a two-layer safety system for an autonomous excavator. A close-range obstacle detection system halts motion if collision with an obstacle is predicted, while a longer-range motion detection system gives the excavator time to halt before human workers or unanticipated vehicles enter the machine's workspace. This safeguarding system is an improvement over other similar systems in that it allows full-speed operation of the excavator while maintaining a high standard of accident prevention. While the obstacle detection approach is fairly specific to an excavator, the workspace monitor has broader application to large autonomous machines which must cease operation before people or vehicles approach.

6. REFERENCES

1. A. Stentz, J. Bares, S. Singh, and P. Rowe, "A Robotic Excavator for Autonomous Truck Loading", to appear in the 1998 Proceedings of the International Conference on Intelligent Robots and Systems, Victoria, Canada.
2. O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", The International Journal of Robotics Research, Spring 1986, Volume 5, Number 1, pp. 90-98.
3. M. Greenspan and N. Burtnyk, "Obstacle Count Independent Real-Time Collision Avoidance", Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, April 1996, pp. 1073-1080.
4. M. Greenspan, J. Ballantyne, and M. Lipsett, "Sticky and Slippery Collision Avoidance for Tele-Excavation", Proceedings of the 1997 IEEE/RSJ International Conference in Intelligent Robots and Systems, Grenoble, France, September 1997, pp. 1666-1671.
5. D. Seward, F. Margrave, I. Sommerville, and R. Morrey, "LUCIE the Robot Excavator - Design for System Safety", Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, April 1996, pp. 963-968.
6. L. Matthies, A. Kelly, T. Litwin, G. Tharp, "Obstacle Detection for Unmanned Ground Vehicles: A Progress Report", 7th International Symposium on Robotics Research, October 21-24, 1995, Munich, Germany.