Hierarchical terrain representations for off-road navigation

Jay Gowdy, Anthony Stentz, Martial Hebert

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

For most autonomous land vehicle tasks, creating the terrain representation is the greatest part of the problem. For example, once a road following system represents the terrain presented to it as road and non-road it is relatively easy to plan a path through the terrain. However, off road navigation does not have the luxury of such a compact representation. An off-road planner needs a detailed map of the terrain, and needs an efficient way of querying that terrain map.

We have implemented a system that satisfies these two constraints for off-road navigation. We first build a Cartesian elevation map from a series of laser range finder images. This map is a complete, but intractable, representation of the terrain. We use the map to build a hierarchical representation of the terrain that we call a "terrain pyramid." Each cell at a level of the terrain pyramid holds the maximum and minimum elevation of the four cells in the level below it. We also build pyramids for various features in the Cartesian map such as terrain discontinuity and slope. The terrain pyramids are shipped to a planner module. We provide the planner module with calls to find the minimum and maximum values of a feature over any rectangle in the terrain. With these calls taking advantage of the hierarchical representation of the terrain. the planner *can* efficiently determine a safe path through the terrain.

1. INTRODUCTION

One difficult problem in the field of autonomous land vehicles is off-road navigation. The basic task for an off-road navigation system is to build a representation of the terrain in front of and under the vehicle, and to plan a path through the terrain. Before planning can even start, there needs to be an efficient way of representing the terrain and getting information about it. In most other autonomous land vehicle tasks there are simplifying assumptions that can be made about the terrain, and most of the computational work is involved in generating a compact representation of the terrain.

For example, in a road following system, the model of the terrain is simple. There is road, and there is non-road. The traversable terrain, i.e., the road, *can* be modeled with simple mathematical entities such as lines' or arcs.' Much processing must be done to generate this terrain representation, but once it is done, deciding whether a patch of terrain is traversable is simply the question, "Is the patch of terrain on the road?" The planning step can make the assumption that everything on the road is traversable.

Another example is indoor navigation? An indoor navigation system can make the assumption that the floor is flat and navigable. Then it can represent the terrain as a set of discrete objects, such as chairs, walls, and tables. When determining the navigability of an *area,* the system just has to determine if there are any objects in the area.

A cross country system can make no such simplifying assumptions about the terrain it needs to traverse. Typical terrain *can* include such things as rocks that the vehicle can't climb, ditches that will cause the vehicle to bottom out, and slopes that will tip the vehicle over. The system has to deal with arbitrarily shaped areas of admissible and inadmissible terrain. To determine if an area is admissible for the vehicle, the planning step has to ask questions like, "Will the vehicle tip over here?", or "Is there a step here the vehicle cannot climb?", or "Is there an obstacle here that the vehicle cannot roll over?" Obviously, there needs to be a way to represent this unstructured terrain so that the planner can find the answers to these questions efficiently.

Only after we developed an efficient method of representing terrain could we implement a real-time generalized cross country planner. The planner that we are developing searches through a three dimensional constraint space, where the three dimensions are the 2D position of the center of the vehicle, and the orientation of the vehicle. The planner classifies areas in this three dimensional space as traversable, or nontraversable, and plans a path through it taking into account the turning radius of the

vehicle. The planner represents the areas of admissibility and inadmissibility in an oct-tree.[4] Using an inefficient terrain representation for such a planner would be disastrously slow.

2. BUILDING A SINGLE ELEVATION MAP

The sensor that we have available for cross-country navigation is an Erim imaging laser range scanner. The Erim scanning laser rangefinder is designed for applications in the Geld of outdoor autonomous navigation. Several versions of the scanner exist, we refer to the version that has been used at Martin Marietta[5] and at CMU[6] in the ALV and Navlab projects respectively. This version is the successor of the earlier ASV sensor used for legged locomotion? It is currently being used for research on navigation and terrain modeling on the Navlab.[8]

The Scanner produces a range image of 255 pixels by 64 pixels with a depth resolution of 7cm for each pixel. A sample image is shown in Figure 1a on the next page. The range images that the scanner produces contain all of the information necessary to navigate through the terrain, but in a fonnat that is difficult for a cross counuy planner to process. A much more natural representation of the terrain is as a Cartesian elcvation map9 In such a map the coordinates correspond to the x and y coordinates of the terrain, and the value at each point in the map corresponds to the height of the terrain.

The conversion from a range image to an elevation map is straightforward. Each pixel in the range image records the distance, ρ, from the sensor to a piece of terrain. The row and column value of the pixel correspond to the vertical angle, ϕ, and to the horizontal angle, θ, that describe the orientation of the laser beam when it scanned that pixel. These parameters, (θ,ϕ,ρ), give the spherical coordinatesof a piece of terrain. The terrain is thcn recorded in the elevation map by transforming its position into Cartesian coordinates.

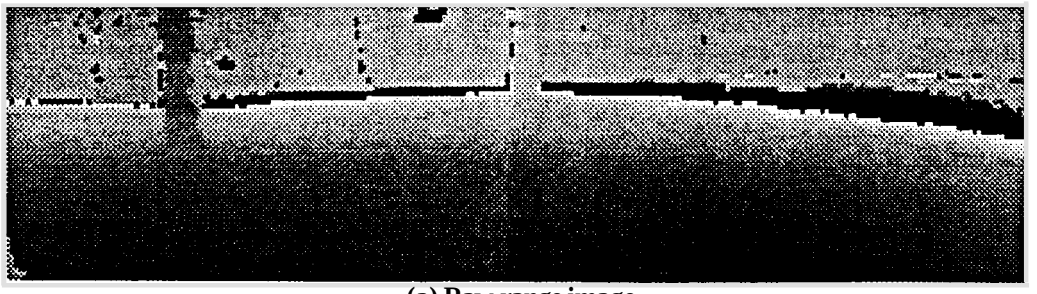
Not all of the range image *can* be used. The Erim measures range data with an ambiguity interval of 20m. In typical outdoor environments, the scanner may be looking at surfaces that are further than the first ambiguity interval. As a result, sharp edges appear in the range image at the junction between data collected within the first and second ambiguity interval where the pixel values wrap around from 255 to 0. It is desirable to retain only the points that are measured in the first ambiguity interval, 0 to 20m from the scanner, because measurements are unreliable past that point and because the artificial edges complicate the image processing.

There is no satisfactory method of removing distant points based on edges because it is not possible to distinguish between occluding edges of objects and edges between regions in different ambiguity intervals. Instead, we chose an approach in which regions that are enclosed by sharp discontinuities arc rcmovcd. This approach does not remove any real object because any object is smoothly connected to other objects, usually the ground. The only case in which this approach would remove an object is when an object floating far above the ground. Such an objcct would cause problems anyway, since there is no way to distinguish between a floating objcct and a tall objcct from the elevation map. Figure 1b shows a range image with the ambiguous section removed.

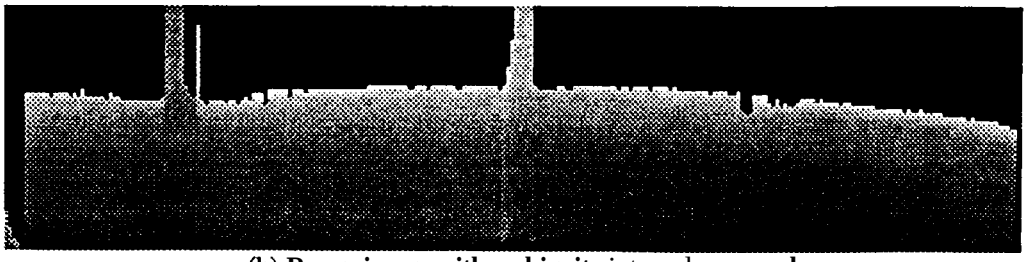
The elevation map is quantized in a two dimensional array with a resolution of 10cm per element. Figure 1c shows the resulting elevation map. Note that elevation is indicated by increased intensity. One can see that the mapping between the range image and the elevation map is not one to one. Close to the sensor, several points from the range image *can* map to the same element in the elevation map, but farther away, the data from the range image is sparse, and there arc empty elements in the map.

Overlapping points are not really a problem. We simply keep track of the maximum and minimum value for each element of the elevation map. To deal with the sparseness problem wc do a very simple interpolation to form a continuous map. First, a list of every empty cell is made. Then, each cell in thc list that has a neighbor with a known value gets the value of that neighbor. If an unknown cell has more than one known neighbor, it arbitrarily picks one of possible values. The empty elements in the sparse section of the map are filled in after two or three nitrations of this procedure. This method provides us with a rather crude, but extremely fast, mcthod of interpolating.and it is sufficient for our purposes. Figure 1d shows the result of the interpolation.

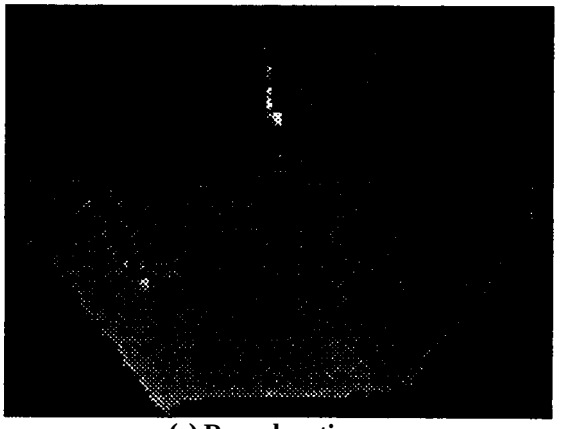
The problem with this interpolation mcthod is that it indiscriminately fills in unknown space in the elevation map, even though there is some unknown space that really is unknown, such as behind an obstacle. In order to differentiate between the two types of unknown space, we must go back to thc rangc image. The interpolation process made a list of all cells in the map that were empty that it filled in with new values. Each clcmcnt in this list is projected back onto the range image. As a first step, any map element that projects onto a rangc image clcmcnt that is in the area that was rejected as beyond the ambiguity interval



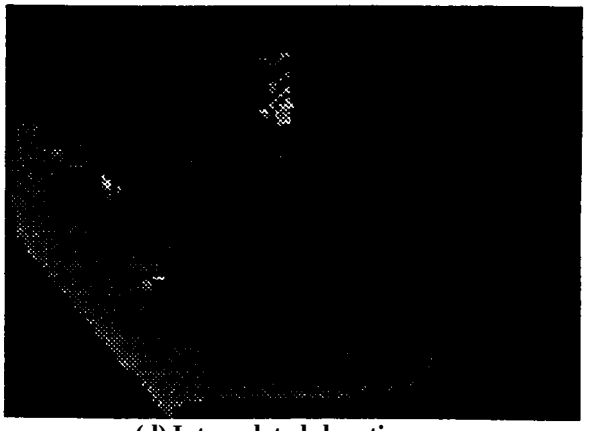
(a) Raw range image



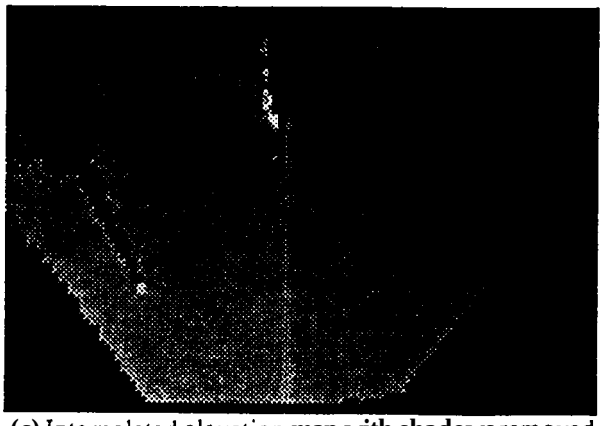
(b) Range image with ambiguity interval removed



(c) Raw elevation map



(d) Interpolated elevation map



(c) Interpolated elevation map with shadows removed

Figure 1: The evolution of an elevation map

is reset to being unknown. If this is not the case, the projected range value of the map element is compared to the range value of its corresponding range image element If the two ranges arc not roughly equal, then the program infers that the map element was generated by the interpolation process expanding into a range shadow, and not by the interpolation between adjacent points. Therefore the map element is tagged as unknown. Figurc 1e shows the elevation map with the shadows removed.

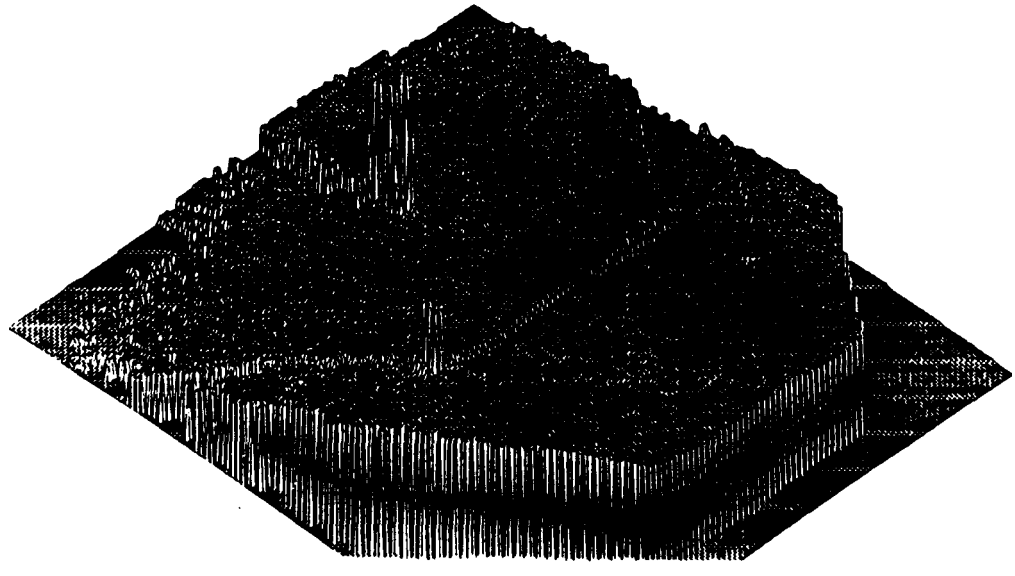
3. FUSING ELEVATION MAPS

A single range image gives information about the terrain starting roughly 6m in front of the vehicle's center. A cross country planner needs this information to plan ahead, but it also needs to know about the terrain on which it is. It does no good to avoid a rock that is 10m ahead, and hit the one that is bcsidc the vchiclc. Thus, data from several range maps needs to be fused together to form one large elevation map that contains all the information needed for planning a path.

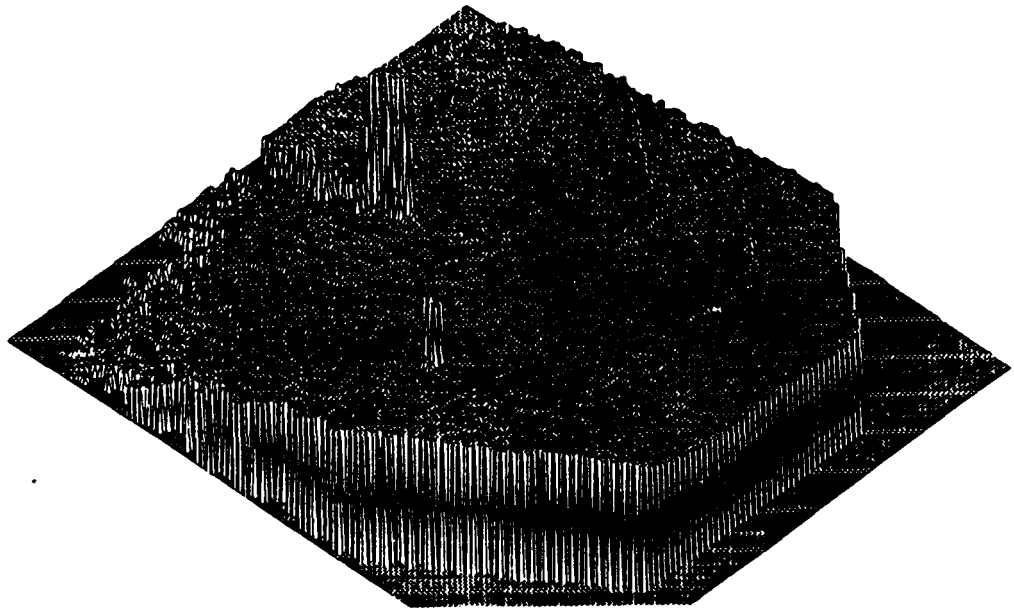
If the range images are tagged with accurate information about the vehicle position and orientation, it should be possible to fuse successive elevation maps using just this information, and knowledge of the sensor and vehicle geometry. We found this method to be inadequate. Our Inertial Navigation System reports the yaw, pitch, and roll very accurately, i.e., to within a thousandth of a degree, and the reported x and y valucs arc also accurate, only drifting one meter over a kilometer run, but the z value of the position is unacceptably inaccurate.'' Figurc 2 shows the result of using this method to fuse elevation maps. The inaccuracy of the z value causes a spurious step to appear in the elevation map. A cross country planner would believe that the step in the elevation map was really in the terrain, and the planning would fail, since the jump occurs over the whole border between the elevation maps.

The solution is simply to trust the reported orientation and x and y position information, since they are sufficiently accurate, and then to determine from the elevation maps themselves the proper offset in z. We calculate the offset by comparing the average elevation values over a patch of terrain from each of the elevation maps that correspond to the same patch of terrain in the real world. A similar method was used in the cross country work done by Hughes.'' Since we want to minimize the difference between elevation maps at the junctions of the maps. wc use the area around these junctions as the comparison patch. This area has the added advantage of using the information gathcrcd from the region of the elevation map that was the closest to the sensor. and which therefore was the densest bcforc interpolation and which is the most reliable.

We do not use every pixel in the patch in calculating the z offset. Recall that each pixel in the elevation map has a maximum and a minimum value. If the maximum and the minimum ate not almost equal in value, then there is an obstacle there. Such a point should not be used in calculating the z offset. If it is used, the slight inaccuracies in the x and y values could have an impact on the accuracy of the resulting offset. For example, if in elevation map A the obstacle was in the comparison window, and in



Figurc 2: An clevation map fused, trusting the reported z value



Figurc 3: An elcvation map fused by calculating the z offset

map B it was not. then the calculated z offset from map A to map B would be larger than the real value. We have also found such points to be somewhat noisy, and therefore even worse candidates for inclusion in the z offset calculation. Figure 3 shows the result of using this method to fuse elcvation maps.

4. THE TERRAIN PYRAMID

The resulting fused clevation map contains all the information that a cross country planner needs to traverse a stretch of terrain. If the planner was just searching along a fixed sct of paths in the terrain, as in the Hughes cross country system, then examining the elevation map would be sufficient.'' but what of a generalized cross country planner that is examining large chunks of terrain, classifying it as admissible and non-admissible, and looking for an admissible path through the tenain? Such a planner would try to classify large areas of the tcrrain as admissible or inadmissible, in order to have the widest choice of paths through the terrain. For example, in the cross counuy planner we *are* developing, the first step in planning would be to examine a 6.4m by 6.4m area of terrain in front of the vehicle center to find the maximum and minimum elevation. With a map resolution of 10cm, such an operation would have to examine 4196 elements in the map. If the difference between the maximum and minimum is less than the height of the undercarriage of the vehicle. then the entire area satisfies this constraint, and the planner can continue planning. Normal tcrrain is much less hospitable, and the elevation over such a large area will probably vary more than the allowed amount. In this case, planncr would subdivide the area into four equal pieces, and would find the maximum and minimum elevation in the two areas closest to the vehicle, examining 1024 elements for each quadrant. The planner would continue this process until it found *an* arca of terrain that satisfied the undercarriage constraint. Clearly, leaving the map in its original state makes the planning prohibitively expensive and causes much duplication of effort.

We attempted to model the terrain to makc qucrying more efficicnt We used polygons to model each patch of terrain that has roughly the same slope. These polygons were hooked together to form a polygonal mesh. The planner would just have to check the polygons, not the map elements thcmsclvcs, to find out about terrain elevations, so we believed that a polygonal mesh would be more computationally efficient. The simple case where the elcvation map is relatively flat, *can* be modeled by a polygonal mesh consisting just a few polygons. For this case, a polygonal mesh is a much more efficient representation of the terrain than an elevation map. The polygonal mesh also has the advantage of filtering out noise, since when building the mesh, if only the patches of terrain that had the exact same slope arc allowed to form polygons, then even if the terrain is perfectly flat in the real world, the resulting polygonal mesh will bc fragmented by the noise in the data. Therefore some variation in slope must be allowed to achieve a reasonable result, and this has the side effect of filtering out sensor noise.

Unfortunately. we found that real world terrain defeated the polygonal mesh. On red images of relatively complex terrain,

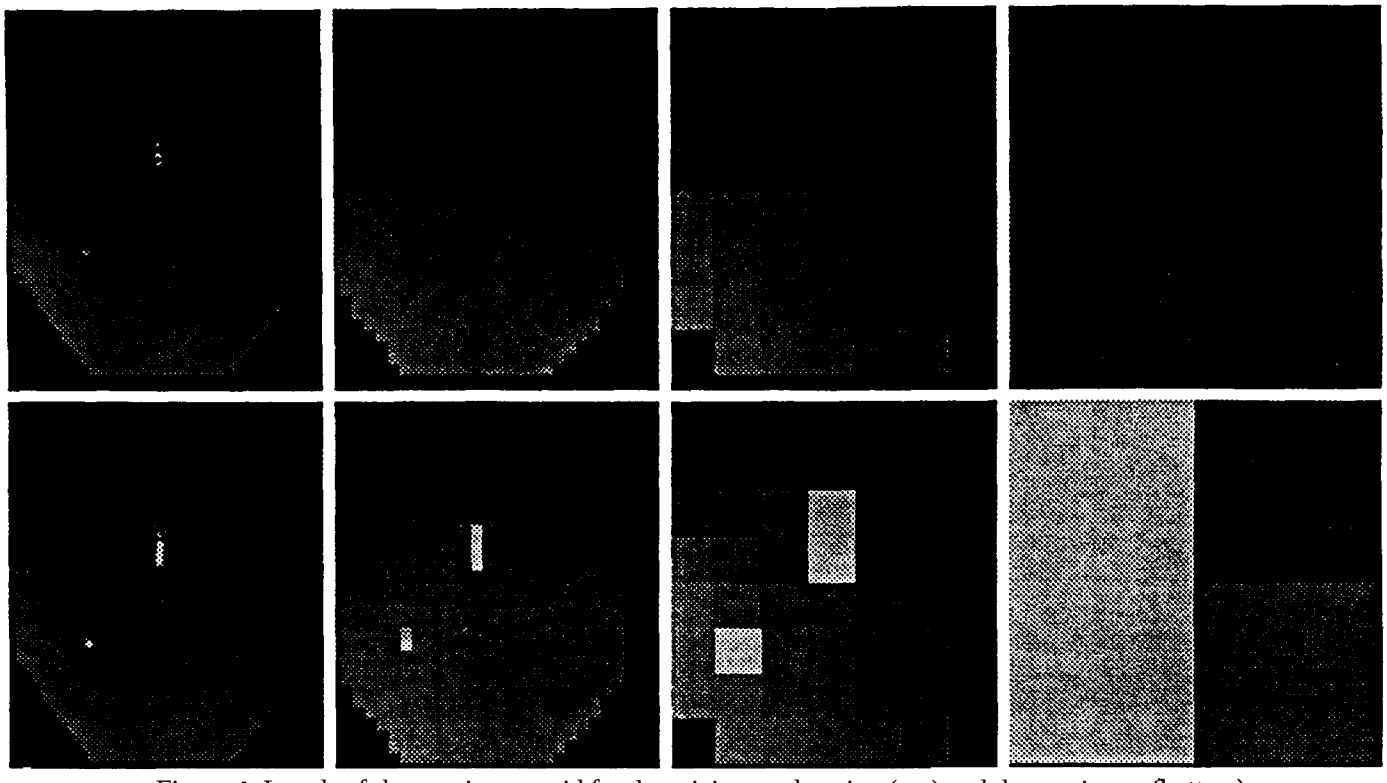


Figure 4: Levels of the terrain pyramid for the minimum elevation (top) and the maximum (bottom)

so many polygons were generated that the computational overhead of doing all of the necessary polygon intersections and minimum/maximum examinations when getting information about the terrain was greater than the computation used with just the raw elevation map to obtain the same information. We also found that building the mesh was very computationally expensive for complex terrain. 2

From this experience, we believe that any model of the terrain complex enough to permit an accurate judgement of traversability in an outdoor environment would have too high a computational overhead to be efficient. Even the noise filtering aspects of such models are not enough to redeem them, since we might achieve a similar effect by filtering either the range images or the elevation maps, or perhaps by a more sophisticated interpolation process.

We found no way around using the elevation map itself for inspecting the terrain, so we looked for a way to use the elevation map more efficiently. One approach could be to build an elevation map with a much lower resolution than 10cm, thus reducing the number of operations needed to find the maximum and minimum over large areas. Unfortunately, the more "efficient" you make this, the less useful it becomes. As the map elements grow larger, the details of the terrain that a planner can pick out become smaller. For example, if there is a pole with a radius of 3cm and a height of 5m in a map element that is 1m on a side, then there is a lot of area in this map element that might be traversable that is rejected as untraversable because of the pole. To extrapolate this approach ad absurdum, the most "efficient" representation is one map element with the maximum and minimum over the whole patch of terrain. Obviously, the planner could only use this representation if the whole patch of terrain was admissible.

We decided to combine the advantages of the low resolution elevation maps with the advantages of the high resolution maps by creating a hierarchical structure that contains maps with many resolutions. We call this structure a terrain pyramid, since it can be thought of as a pyramid whose base is the highest resolution elevation map, and whose top is the map just containing the maximum and minimum elevation values over the whole area of terrain. Finding the maximum and minimum elevation over an area using this structure is very similar to building a quad-tree representation of the area.[12] The elevation of large parts of the area can be checked with a low resolution map, while the border of the area can be checked with higher resolution maps. Thus, since the planner usually does a lot of querying for elevations over different areas. the efficiency of the access pays for the initial overhead of building the terrain pyramid. Figure 4 shows various levels of a terrain pyramid.

Building the terrain pyramid is quite easy. The bottom layer is simply the elevation map that we built before, with a remlution of 10cm. Consider each group of four elements in this map, $elem_{2i,2i}$, $elem_{2i+1,2i}$, $elem_{2i,2i+1}$, and $elem_{2i+1,2i+1}$. Then $elem_{i,i}$ in the next level of the terrain pyramid contains the maximum and minimum elevation value over the group of four elements. This step is repeated for each level in the terrain pyramid, until the top is reached. The top level should hold the maximum and minimum over the whole area. The number of comparisons required to build the maximum and the minimum features is just the total number of elements in the map. This follows from the fact that to build the first level of the map requires 3 comparisons for each p u p of four in a n by n map, i.e., $3\times(n\times n)/4$ comparisons. The next level requires $3\times(n\times n)/16$ comparisons, and the one after that $3\times(n\times n)/64$. The total number of comparisons is $3n^2\times(1/4 + 1/16 + 1/64 ...) = 3n^2/3 = n^2$.

5. INCORPORATING OTHER FEATURES

While the terrain pyramid is very efficient for finding bounds on the terrain elevation over an area, there are some features that our planner needs that would be very inefficient to extract from the terrain pyramid as it is. For example, another constraint besides the undercarriage height of the vehicle is the maximum step that the vehicle's wheels *can* climb over. In order *to* tell if a wheel position satisfies this constraint, the planner would query the pyramid. If there is difference in the maximum and minimum elevation of the area that the planner queries that exceeds the maximum step size, then there are two possibilities for the structure of the terrain. First. there could be a gradual change in elevation over the area, in which *case* the area is navigable. Second, there could be a sudden jump in a small area, in which *case* the area is innavigable. Just from querying the maximum and minimum elevation over the area, there is no way to tell the difference between the two possibilities. The planner has to examine the highest resolution elevation map pixel by pixel to look for jumps in elevation between adjacent elements.

In order *to* alleviate this problem we incorporated a measure of terrain discontinuity into the terrain pyramid. The terrain discontinuity feature for a target element in the map is defined as the difference in elevations between the element with the maximum elevation value and the element with the minimum elevation value in a 3 by 3 window centered on the target element. If there are any unknown values in the window, thcn the terrain discontinuity is considered unknown as well. We calculated the terrain discontinuity for each element in the elevation map to form the bottom level of the terrain discontinuity feature. We propagated maximum discontinuities up the terrain pyramid in the same way as elevations. *So,* if the planner finds that the maximum value of the terrain discontinuity feature over an area of terrain is less then the maximum step size then even if the difference between the maximum and minimum elevation for that patch is greater than the step size, the planner knows that the area is admissible for the wheel to run over. If the maximum value of the terrain discontinuity feature for an area is larger than the maximum step size this does not mean the vehicle has to climb that step. It might be that instead of having a step up, there is a step down. There is no easy way to tell the difference just from the terrain discontinuity. So what does the terrain discontinuity feature gain you? It *can* definitely tell if an area is admissible more efficiently than the elevation values by themselves. If the terrain discontinuity value is too large in an area, then the planner can use the heuristic of searching other areas first *to* find a smoother one to go through. Only in the case where there are no areas are smooth enough would the planner do a more laborious search through the ambiguous terrain..

Another constraint that a planner must obey when planning a path is the vehicle tipping constraint. Even if there are no obstacles that would jam into the undercarriage and no steps that the vehicle can't climb, if the slope the vehicle is on is too steep then the vehicle will fall over. As with the stcp constraint, we decided to incorporate another feature into the terrain pyramid to assist in discrimination between terrain that was too steep and terrain that was passable. The obvious feature *to* use would be the slope of the terrain at every point. A truc measure of this could be calculated by fitting planes to small windows of the terrain, and using the resulting slope. Wc implemented this, and found that the expense of calculating this feature was not justified by the savings in planning time.

We did not give up on finding a feature to fit the needs of thc planner. What the planner needs to know is how the terrain changes over large areas. It does not necessarily have to know thc uue maximum slope in order to discriminate between terrain that might be too steep and terrain that is traversable. We use instead what we call the gradient feature. The gradient feature is produced by running the same operator that produced thc terrain discontinuity map, but running it on a lower resolution level of the terrain pyramid. Typically, we use the map that is 2 levels above the base elevation map. That means that each pixel that the operator works on represents 16 map elements, or an square 1.6m on a side. Generating this feature only takes 1/16th of the time to calculate and propagate up thc pyramid as docs thc tcrrain discontinuity feature, and it suffices to give the planner an indication of the steepness of the terrain.

The above illustrates the philosophy we had in representing the terrain. If preprocessing the terrain by building the terrain pyramid and adding extra features helps the planning process, then we do it, unless the cost of adding the feature is greater than the benefit We do not add any other features just in case they will be useful, but rather we add features to the pyramid that are designed specifically to help with the checking of specific constraints.

6. ACCESSING THE PYRAMID

The module that acquires range images and builds the terrain pyramid runs on a different machine than the module that does planning. The pyramid builder packages the pyramid and sends it to the planner module. The only compression that is done is that each sequence of unknown elements greater than length 2 is sent as a single unknown value followed by the number of unknown elements in the sequence. This is done because the number of elements in the base level of the terrain pyramid must be a power of four. This is obvious from the algorithm, which reduces the number of elements by a factor of four for each level. Unfortunately, the dimensions of a real elevation map generated from the range images hardly ever meets the dimension requirements, and has to be padded out with unknown values. This method of storage sacrifices memory to lower the bookkeeping costs, but memory is easy to add, while speed is precious in a real-time application such as cross-country planning. We did not want to waste bandwidth with useless data though, so we do this simple compression.

The receiving module unpacks the pyramid, and stores it. The receiving module can query the map for information over an area. Our hierarchical terrain representation is well equipped to handle areas of any shape, since all that would be *required* would be to build a quad-tree representation of the shape, and then to search the cells for the minimum and maximum of the relevant features. We decided that this arbitrarily shaped area capability was not needed, and limited the shape of the area to be *rectangular*. The algorithm for extracting the maximum and minimum of a feature is simple. The planner requests the range of a feature over the bounding box with lower left corner, (x_{min}, y_{min}), and upper right corner (x_{max}, y_{max}). First, the smallest bounding box with the dimensions of a pyramid element is found. For example, if the planner requests the minimum and maximum elevation from map with a resolution of 10cm and an origin of (10.0,10.0) over the area bounded by (15.0,15.0) and (16.0,16.0), then the smallest bounding would be a box 1.6m on a side with lower left corner at (14.8,14.8). 1.6m is the resolution of an element four levels up from the base of the pyramid. Then, if the pyramid bounding box and the requested bounding box are the same, we are done. If they are not, the pyramid bounding box is split into quadrants. Each quadrant is checked to see if it is completely inside the requested bounding box. If it is, the maximum and minimum of the requested feature from the quadrant is added to a list. If a quadrant is not completely in the requested bounding box, then it too is split into quadrants, and the process is repeated until the minimum resolution is reached. When the requested bounding box has been completely partitioned, the list of feature values that has been maintained is searched for the maximum and minimum value. Figure 5 shows how the requested bounding box in our example is split up. Note that for this example. only 13 nodes had to be accessed; if the terrain was not represented as a pyramid, then 100 elements would have been examined.

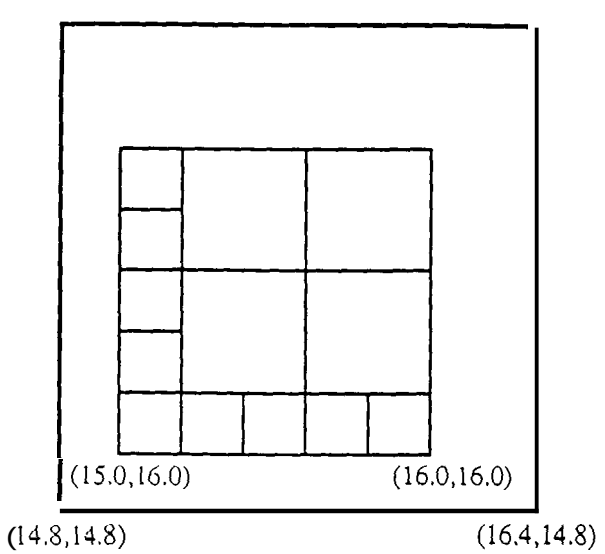


Figure 5: The partitioning of a bounding box

7. PERFORMANCE

The terrain pyramid builder ran on a Sun SparcStation with 32M of memory. The total time to collect a range image, convert it to an elevation map, fuse it with the previous elevation maps, and ship it to the planner was 2.4s. This includes 0.5s to acquire the range image, 1.1s to convert the range image to a elevation map, 0.1s to create the terrain discontinuity feature, 0.4s to build the terrain pyramid, and 0.3s to ship the terrain pyramid to the planner. The planner runs on a separate, identical, SparcStation, and the two modules are connected via an EtherNet.

Considering that the cross-country planner that we are building is expected to take 2 to 10 seconds per planning cycle, the 2.4s cycle time of the terrain pyramid builder is more than adequate. Figure 6 shows one slice of the constraint space generated by running our cross-country planner on the elevation maps we have been using as examples. The crossed areas represent inadmissible areas of terrain, i.e., areas on which it is illegal to place the center of the vehicle. As the figure shows, the planner generated a path through the terrain. In the process of planning this path, the planner made 1493 queries of the terrain. If we were using the raw elevation map we would have had to examine 530,610 map elements to generate this path. Using the terrain pyramid, we only had to look at 78327 pyramid nodes. This is close to a 7 to 1 savings. Considering how little of the time spent building the terrain representation was spent building the terrain pyramid, this is a large overall saving of processing time.

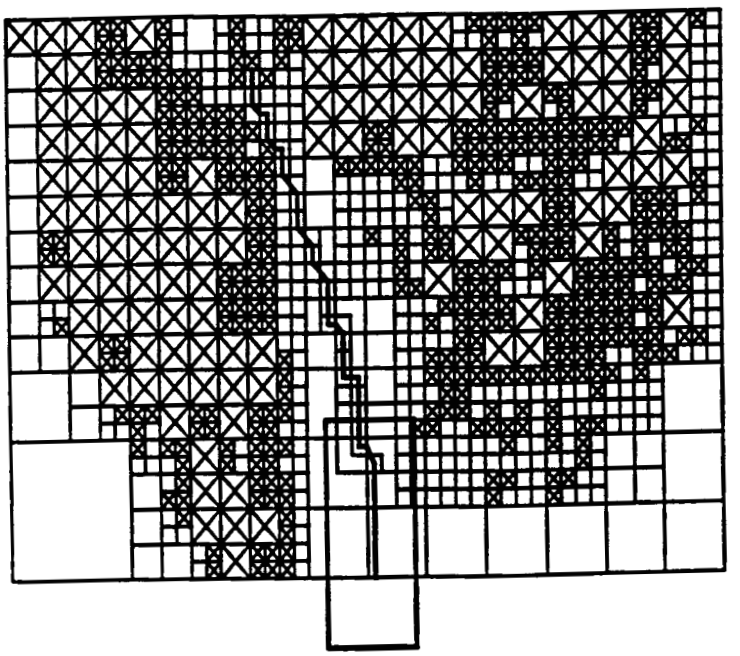


Figure 6 Planning results

8. CONCLUSION

Our goal was to provide our cross country planner with a method of getting information about the terrain around it efficiently. Without some intermediate data processing, the planner would have to search individual elements of the elevation map to obtain the information it needs. Since the planncr examines many overlapping areas, using this method causes much duplication of effort.

There is no apparent compact and elegant method to model the terrain that a cross country system needs to traverse. For one thing, unlike in road following, there are no regions that are guaranteed to be traversable. Unlike indoor navigation, the cross-country system has to deal with arbitrarily shaped regions of untraversable terrain, not just discrete objects. On top of that, due to the complexity and randomness of natural terrain, any attempt to make the problem simpler by fitting mathematical models to the terrain ends up with a representation that is just as intractable as an elevation map.

Instead of looking for a more compact representation, we achieved our goal by implementing a system that represented the

terrain hierarchically, in a pyramid of elevation maps of descending resolution. Getting information about a patch of terrain using the terrain pyramid is more efficient than using the raw elevation map in the same way that representing an object with a quad-tree is more efficient that representing it with pixels. For a small initial cost, the amount of duplicated computational effort needed by the cross country planner is vastly reduced.

By considering what the planner needed to know about the terrain, we were able to reduce the computations by adding extra features to the terrain pyramid, such as terrain discontinuity and terrain gradient. We only added such features if the initial overhead of calculating the feature was less then the computational benefit to the planner.

This illustrates our approach to the problem of terrain representation for cross country planning. We examined the task that the data is being used for, and we preprocessed the data when the preprocessing was relatively cheap and reduced the duplication of effort for the planner. We see this task oriented approach as a good paradigm for tasks in which the data is dense, intractable, and very difficult to model efficiently.

9. ACKNOWLEDGMENTS

We would like to would like to thank every one in the Autonomous Land Vehicle group for there input and time in this research. Special thanks go to Chuck Thorpe for providing much needed direction, to Omead Amidi for his work with the INS, to Mike Blackwell for nursemaiding our range scanner, and to Jim Frazier for all his time spent driving our software around. This research was sponsored by DARPA, DOD. monitored by the US Army Engineer Topographic Laboratories under contract DACA 79-89-C-0014, titled "Perception for Outdoor Navigation."

10. REFERENCES

1. J. Crisman, C.E. Thorpe, "Color Vision for Road Following," in Vision and Navieation: The Carnegie Mellon Navlab, C. Thorpe, *ed.*, Kluwer Academic Publishers, 1990.

2. B. Mysliwetz and E. Dickmanns. "Distributed Scene Analysis for Autonomous Road Vehicle Guidance," Proceedings of the SPIE Conference on Mobile Robots, November, 1987.

3. G. Giralt, R. Chatila, M. Vaisset, "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobots", International Symposium on Robotics Research. 1985.

4. A. Stentz, "Multiresolution Constraint Modeling for Mobile Robot Planning." Proceedings of SPIE Symposium on Advances in Intelligent Robotics Systems. November, 1989.

5. R.T. Dunlay and D.G. Morgenthalcr, "Obstacle Detection and Avoidance from Range Data", in the Proceedings of the SPIE Mobile Robots Conference, 1986.

6. M. Hebert, T. Kanade, I. Kweon, "3-D Vision Techniques for Autonomous Vehicles," Robotics Institute, Carnegie Mellon University, technical report no. CMU-RI-TR-88-12, August. 1988.

7. D.M. Zuk and M.L. Dellcva, "Three-Dimensional Vision System for the Adaptive Suspension vVhicle," DARPA technical report No. 170400-3-F, January, 1983.

8. C. Thorpe, "Outdoor Visual Navigation for Autonomous Robots," in Vision and Navigation for the Carneeie Mellon Navlab, C. Thorpe, ed.. Kluwer Academic Publishers, 1990.

9. M. Hebert, T. Kandade, "The 3D Profile Method for Object Recognition", in Proceedings of the IEEE Computer Society, June, 1985.

10.0. Amidi, "Integrated Mobile Robot Control," in these proceedings.

11. M. Daily, J. Harris, D. Keirsey, K. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong, "Autonomous Cross-Country Navigation with the ALV," in Proceedings of the IEEE International Conference on Robotic and Automation, 1988.

12. H. Samet, "The Design and Analysis of Spatial Data Structures", Addison Wesley, 1990.