Final Project Report
CS 633

PRM With Voxel Assisted Probability Distribtion.

Abstract :

Voxel representation of the work space is used to create a discrete understanding. Metrics are computed over the voxel space. Resultant metrics values are used to guide a PRM so as to make the sampling distribution semi-random. The guidance used is based on a search for local maxima among normal distances at each voxel. These locations correspond to medial edges in a voronoi diagram, and they represent locations which have maximal clearance with obstacles. The size of local maxima identifies narrow passages, because the local maxima in a passage will be very small (walls are nearby and normal distances will be small.). Narrow passage areas are sampled at higher rates than open areas, which leads to a focus on narrow passage discovery.

Motivation and Introduction :

The primary motivation of this project is to apply voxel space as a data structure for understanding empty space. The goal is to identify critical areas inside the voxelization, such as locations of maximal clearance, and locations which represent narrow passages. Areas of maximal clearance allow the robot to move freely, and adequate sampling of narrow passages allows the robot to navigate through tight areas.

Any given point in space could be a more or less useful location for sampling in a PRM. The usefulness of any sample location depends on the distance from opticals. A location right up against an optical, closer than the smallest distance the robot can accommodate, is useless. Similarly, there is no need to over-sample areas that are clearly larger than the robot can possibly span.

Voxelization is a discrete representation of spatial understanding. Each voxel represents a bounding box for any given piece of geometry.

Voxelization also allows the user to use a resolution that is independent of the model fidelity. Voxelizing two essentially identical models, one high fidelity, and one low fidelity, will result in approximately the same voxelization. Voxelization carries with it an up front memory cost, but it allows the algorithm to dial in its memory usage.

Voxels are also useful for storing data in a 3d accessible format. Any given voxel can convert to and from a 3d point in constant time. Given a point, the corresponding voxel can be looked up immediately in a hash manner, and any data stored in the voxel can be immediately retrieved. Storing 3d lookup data in a voxel structure is very fast, albeit wasteful. But if memory is not a limitation, voxelization can be used to accelerate many data access operations.

In this report I apply voxel space to a PRM problem. I bin the optical geometry into the voxel structure, and I calculate metrics over the voxel structure. The metric that I primarily use is approximate norm distance. Along with this metric, I store flags marking voxels that represent local maxima within the normal-distance field. This data allows me to judge a point's distance to opticals by accessing the voxel that bounds the point, and reading the norm distance value. Important areas are concentrated on, such as the voxels whose normal distances are local maxima. These locations are voronoi medial voxels, and represent areas that are in some manner large-clearance locations, and should offer a robot plenty of room to navigate.

Methods Used and Studied :

A Simple Algorithm for Complete Motion Planning of Translating Polyhedral Robots.

In this paper, the authors cover a method which uses an oct tree to bound opticals with the leaf nodes, where each leaf bounds a portion of the geometry which is judged to be simple enough to not require more decomposition.

The oct trees leaves represent cubes, which are simple graphs. Consecutive cubes represent a complex graph composited of many little cube graphs. This larger graph is searchable. If the graph is generated in c-space, then it provides paths through the environment. By searching only nodes not in collision with opticals, non collision paths can be found.

I originally intended to use a multi-resolution representation like an oct tree. Had I implemented an oct tree instead of a grid, I would also have had to decide when to terminate subdivision. A regular grid was simpler and faster to implement. Given time constraints, the paper's method was not implemented.

Efficient Max-Norm distance Computation and Reliable Voxelization.

In this paper, the authors describe methods to voxelize geometry, and calculate the norm distance at any point.

While I didn't use the methods in this paper, this paper is what made me decide to use norm distance as a metric. It is the relationship of maximal norm distance and voronoi diagrams that made it easy for me to find areas of maximal clearance.

My sampling algorithm was formed from two parts. Voxelization, and a guided random sampler that uses the voxelization.

The first part of the sampling algorithm is the voxelization class. This portion is responsible for describing the discretized space. This class relies on three main pre-processing stages, and provides query functions for looking up voxels that meet certain criteria.

The first stage of the voxelization is to provide point binning. This is a per-point operation that generates a hash index when given a point, and marks the corresponding

cell at this index as occupied. This is accomplished by checking the penetration of every axis within the voxel space, casting to an integer, and converting the x y z integers into a linear index. (FIG A) (FIG B includes bounding box)

This operation is completed in constant time because it is a simple hash

The second stage of the voxelization is to calculate pixel-wise norm distance for the empty space. This is accomplished by an iterative method. The method fetches a list of occupied cells, then finds all unvisited neighbors and creates a second list. The process repeats using the unvisited neighbor list as the primary list. The end result is a behavior that marks all voxels with a voxel-wise normal distance, forming a gradient that maximizes at the locations farthest from any originating obstacles. (FIG C)
The second stage also manages to create a voronoi diagram. The voxels that are a local maxima are the intersections of gradients. These intersections fall at the farthest distances to all objects. A list of all local maxima voxels is also a list of all voronoi medial voxels.

This operation is completed in constant time relative to the total voxel count. Voxel count is subject to the voxel resolution, and in some cases can be extremely large when dealing with a 3d volume. Also, because each voxel has 26 neighbors, and all neighbors are tested, the complexity is really about 26n. Combining a large constant factor with a large space creates a time consuming problem, even if the computational complexity is linear.

The storage requirements for voxelization (granted that it is a single stage single resolution binning volume) are very large. The space required is (voxel size) * (x resolution) * (y resolution) * (z resolution). Presuming each voxel has only 1 byte size, and resolution is 512, the storage requirement would already exceed 130 megabytes. A resolution of 1024 in all dimensions would exceed 1 gigabyte if using a single byte per voxel.

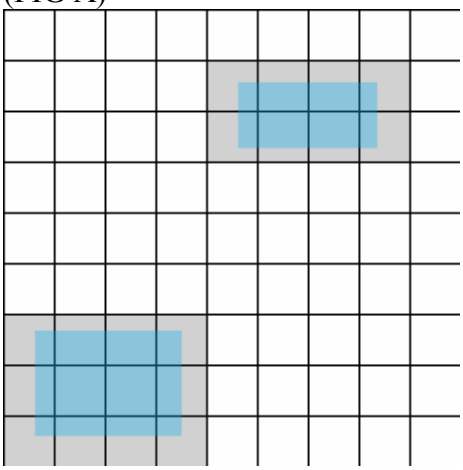
The third stage of voxelization is the explicit demarkation of local maxima voxels. This method consists of a traversal through all voxels, there the current voxel if compared to all-neighbors. If the current voxel has the largest normal distance, then it is a local maxima. A relaxation factor can be used to allow voxels that are almost-local-maxima to also qualify. A relaxation factor of 0 finds voxels that are centers of volumes. A relaxation factor of 2 allows for the case where a voxel on either side can also be a maxima. Allowing 2 neighbors to be co-maxima allows chains of voxels to be marked, resulting in lines that trace medial axes. Increasing the relaxation factor further allows minor sub-gradient intersections to draw local maxima status. (FIG D)

Performance limitations of the local maxima calculation are slightly better than the normal distance calculation. Each voxel still has to check 26 neighbors, and the total voxel count is still very large, so this method is also relatively slow even though it is a linear operation.

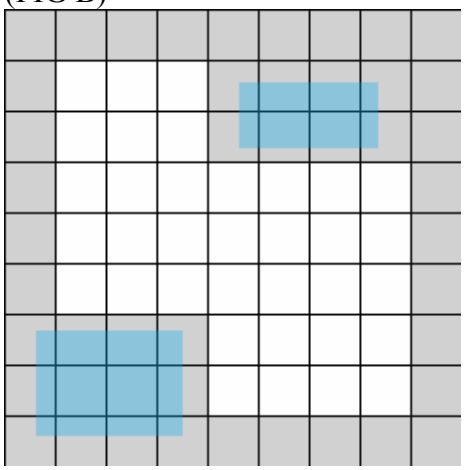
The query operations are very basic. It is a traversal through the entire voxelization. Along the way we save indices to a list if they meet certain criteria.

This operation is linear to the quantity of voxels.

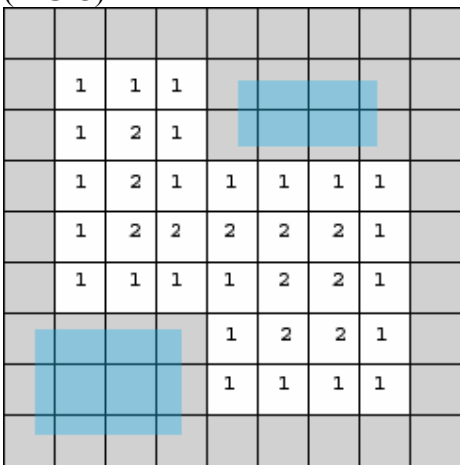
(FIG A)



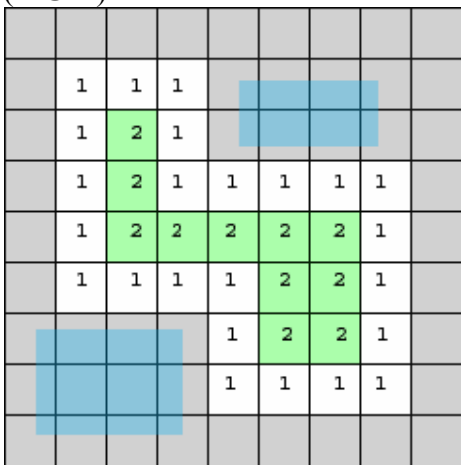
(FIG B)



(FIG C)



(FIG D)



The second part of the sampling algorithm is the configuration generation class (cgen). It is this class that uses the voxelization class to do it's work.

First, cgen parses the obstacle geometry. It up-samples the point density until each triangle's points are spaced closer than the voxels can resolve. These points are binned into the voxelization.

Second, cgen marks the voxelization borders as occupied. This makes the bounding box be treated as an obstacle.

Third, the voxelization's built in normal-distance-calculator and local-maxima-calculator are called to process the voxelization.

Fourth, the voxelization is searched for voxels which meet certain local maxima criteria. The voxel indices are placed into a list.

lastly, the list is used to provide the locations that are used in a random sampler. The specific sample locations are dithered, so configurations are not exactly located at voxel positions, but are probabilistically centered around valid voxels. The pose of each sample is completely random. Locations are randomly dropped to reduce sample count. Locations with larger maxima normal values are dropped at higher rates, so that sampling is biased to narrow areas and passages.

The result of these samplings are used to feed the configuration linker, and ultimately solve the navigation problem.

Results/Findings :

A minimum resolutions of 2 voxels is required to detect a narrow passage. Hence an adequate resolution parameter is required. 2 voxels are important, so that there can always be at least 1 complete voxel in a passage when the passage is not voxel-aligned. This would result in a 0 - 1 - 0 pattern, of which 1 is a local maxima, and lies between passage walls located at 0 and 0. 1 voxel away is the smallest representable normal distance.

Performance is generally slow with high resolutions, and quick at low resolutions. Giving exact numbers is not important, because any resolution higher than 2-voxels-per-feature will have similar results. Higher resolutions would only have a finer level of exact-center-detection, but both would still detect narrow passages.

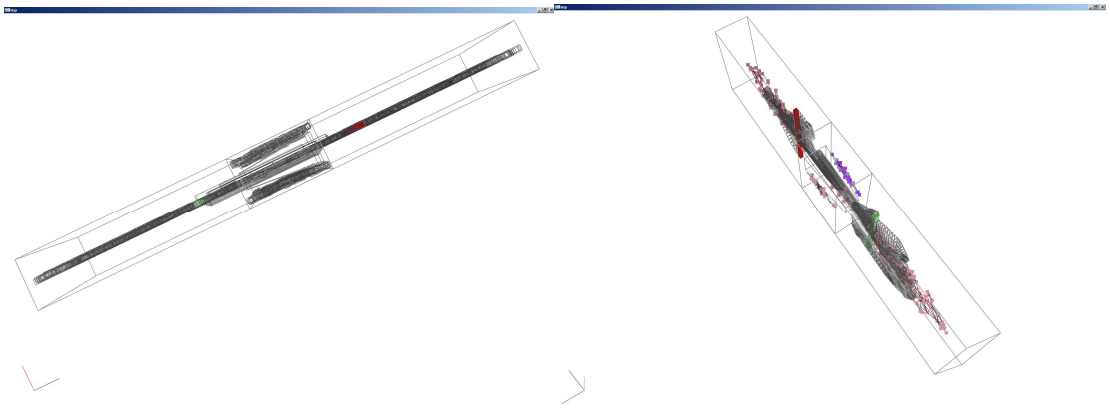
The reason why I say there is little importance to performance figures, is that different resolution settings can have vastly different run times, and can have incredibly similar results. Having twice the dimensional accuracy requires doubling three dimensions, which increases computational work by 8.

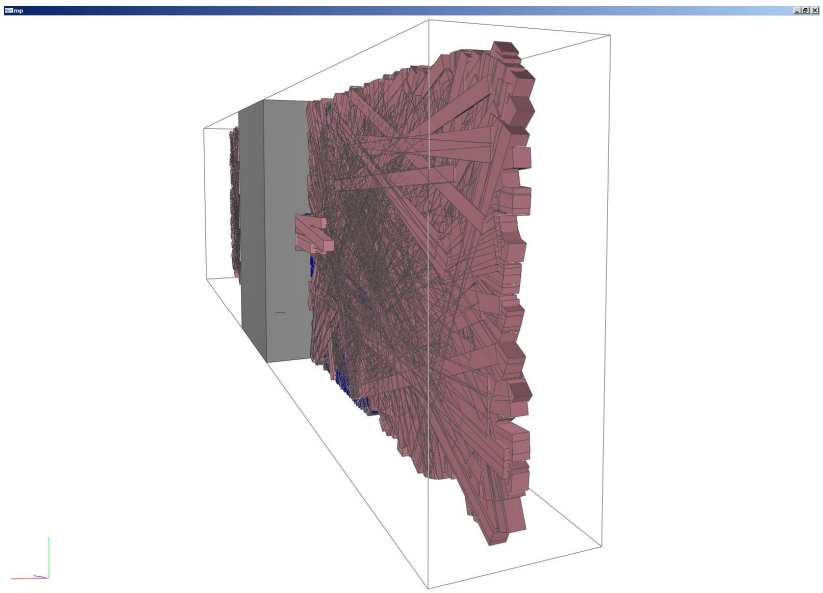
Because the samples are dithered each time a sample is tested, having exact accuracy is wasteful. Exact locations are not even used.

The important thing is to simply have "enough" resolution. What is enough is subjective beyond the point of 2 voxels per feature. Sometimes you need accurate clearance, and sometimes you need fuzzy clearance. Experimentation is required to find the best settings.

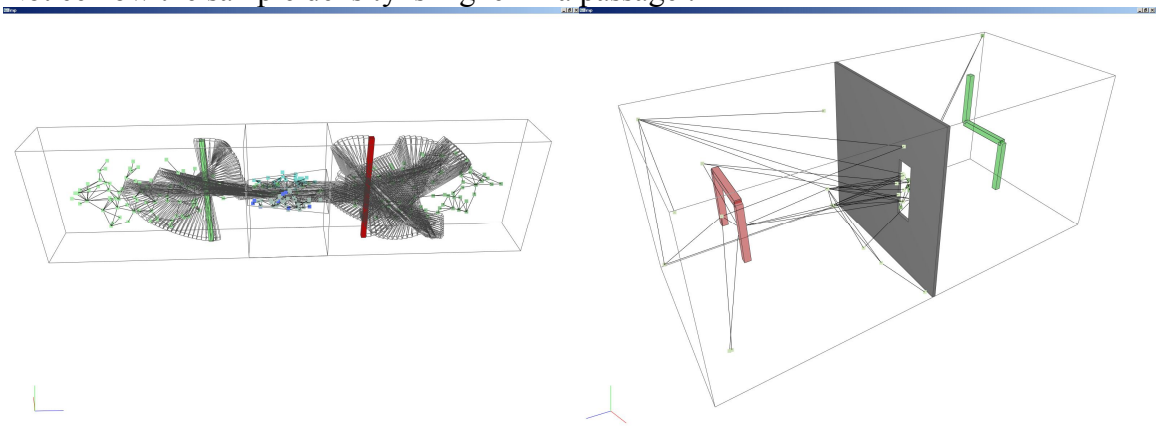
To state a number, a resolution setting of 512 runs in approximately 8.5 seconds. Whether or not 512 is required (or simply 20 is good enough) depends on the problem.

Samples successfully staying near center of volume, away from bounding box and obstacles :

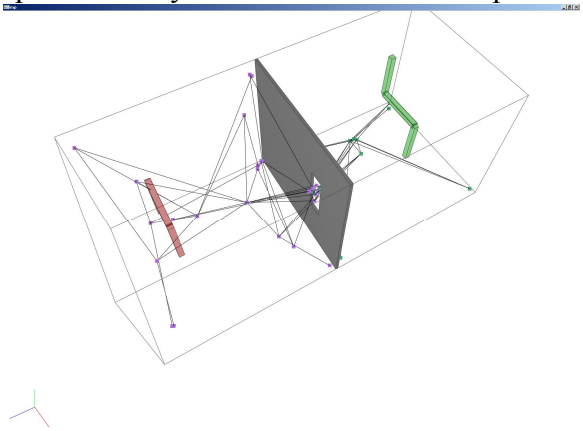




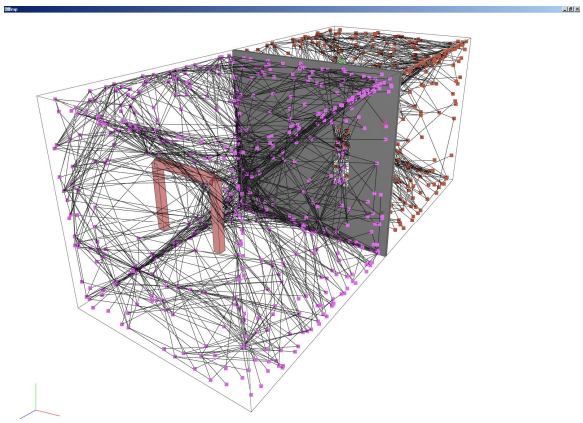
Notice how the sample density is higher in a passage :

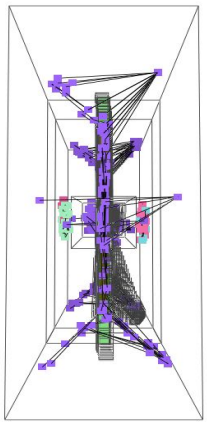


Notice how the sample density at the narrow passage is very high, but the samples in free space are very limited. Too limited to provide similar neighboring configurations.



High relaxation factors lead to areas that are not maximal-clearance, but are fearful of intersection. In these images, you can see how sample points approach walls using paths that are maximal distance from neighboring walls. Inside a box, this results in a sampling pattern that is similar to 6 pyramids with their tops connected in the center.





Conclusion :

The goal of creating a sampler that would discover narrow passages was met. The voxelization managed to highly sample narrow passages while simultaneously sparsely sampling empty space.

The goal of limiting samples to areas of obvious clearance was also met. The sampler would avoid generating configurations in places with lesser innate probability for successful passage.

The synergy of both goals came together with the sampler generating more points in a narrow passage, while keeping to the centerline of the passage.

The failure of this approach is that it is too focused. What I discovered was something that should have been obvious, but didn't seem all that obvious at the time.

- Areas that are NOT clearance maximal in work space are required in order to generate samples that allow an object to pass between areas of good clearance.

Clearance maximal locations are those which are approximately (within reason) farthest away from all obstacles.

Unfortunately, areas which are clearance maximal for a sample point are not really clearance maximal for the robot. For the robot to move through an area of tight fit, it may have to move to areas that are not clearance maximal sample points.

This is intuitively obvious because the sample points, while optimally placed in work space, are in collision in configuration space.

What this means is that in order to successfully generate good samples in work space, the robot needs to be assessed. The normal distances need to be adjusted by the important radial distance of the robot. A sample with a normal distance of 5, with a robot of 10 points in radius, would actually be at -5.

Discussion :

To continue the last paragraph, the robot should have been used to alter the placement of sample points.

The locations generated by the algorithm are still good locations because they represent areas of good clearance. If a point has best clearance at a certain location, then a robot is likely to have best clearance in a similar location.

Clearance in configuration space is implied by this sampling technique, but it is not guaranteed.

One thing that I don't want to do is exaggerate the importance of configuration space in determining samples. This method is still a PRM. The results of collision tests on the random samples builds a clearing subset of configuration space. My approach tries to use work space to limit the quantity of c-space that is constructed.

The ultimate failure of my algorithm is not that it is unaware of c-space, and that is only tries to stay in areas that are hopefully good in s-space.

The ultimate failure of the algorithm is that it is unable to build a proper 'fuzzy' sample field around the maximal clearance locations. The sample points are simply too constrained. With not enough samples, the search finds configurations that are valid inside a narrow passage, and it finds configurations that are valid outside the narrow passage, but it under samples nearby areas which are required in order to offer a proper entry and exit path to and from a narrow passage.

Future Work :

The large storage issues can be mitigated by not using a rigid voxelization, and instead using an oct tree. In this structure, areas with no data would be unoccupied. However, the ability to query for normal distance would have to rely on a different test, such as a star shaped intersection test, which samples a point in space and accepts the shortest environment intersecting ray as that point's normal distance.

However a strength of rigid voxelization is that a rigid buffer can easily be circularized. What this means is that the volume that the voxelization describes can be incrementally mobile along any base axis. Imagine that you can add a new set of voxels to one side of the voxelization, and drop off the opposite side's set of voxels. This would be progressive updating and motion of the voxelization. In reality, rather than actually adding and dropping voxels, the X, Y, or Z indices can be applied an indexing-offset. Then only a single side of the voxel volume needs to be updated at the indexing-offset index. When the volume can update and shift in any direction like this, it is useful in mobile applications. Imagine a vehicle which is progressively binning LIDAR data in a self-localized manner. A coarse voxelization of LIDAR data would allow this vehicle to do mid-level navigation among local obstacles.

I realize now that the proper approach would have been to use the generated sample points as a guide to another layer of analysis. This new processing layer would calculate a voxelized version of explicit configuration space in limited regions around the sample points. Then the sample points would be replaced with new sample points that are good choices inside the configuration space representation.

When using a cropped/limited space, smaller, more memory and time saving voxelizations could be used. Also, because the area would be so limited, the voxelization could be with a higher fidelity while still being minimal in processing impact.

This would result in a detailed c-space approximately optimal-clearance point set.

I could approximate this approach without much extra work by re-iterating the normal-distance calculation, and storing the results as a new metric. This new metric would be normal distance from the original local maxima. Then I could search for all voxels within a certain distance of the local maxima. This would give me a subset of voxels with sufficient 'fuzzyness' around the local maxima to provide proper entry/exit from narrow passages.

Bibliography :

- **Efficient Max-Norm Distance Computation and Reliable Voxelization**
  - Gokul Varadhan, Shankar Krishnan, Young J. Kim, Suhas Diggavi and Dinesh Manocha
  - Published : Eurographics Symposium on Geometry Processing (2003)

- **Efficient Max-Norm Distance Computation and Reliable Voxelization**
  - Gokul Varadhan, Shankar Krishnan, Young J. Kim, Suhas Diggavi and Dinesh Manocha
  - Published : Eurographics Symposium on Geometry Processing (2003)