Term Project for MAE 294

Toolpath generation and application in robotics

(A. y. 2003-2004)

Student: Lanciai Marco
Professor: Daniel Yang
First article


Review of the article

The authors treat different solutions about the problem of driving milling machines for models defined by freeform trimmed surfaces. In practice two approaches are commonly used to generate toolpaths for surfaces, neither of which is optimal, in general. In the first one isoparametric curves are extracted from the surface, usually in equally spaced parametric steps. These isocurves usually span the entire parametric domain of the surface and will be referred to as complete-isocurves. Although simple to determine, toolpaths created using complete isocurves equally spaced in parametric space are clearly not optimal (it doesn’t provide a valid coverage for S and its path length is not minimal) and are redundant as in the fig. 1a. This affect machining time and part finish.

An alternative method for generating toolpaths is based on contouring planes, in which the surface is intersected by (frequently geometrically equally spaced) parallel planes. The resulting toolpath is only a piecewise linear approximation to the real intersection and the size: if the surface has regions almost coplanar to the contouring plane, plane adjacent contours would be distant from each other invalidating the toolpath as in fig 1b. Recent work has been done in estimating plane spacing based on estimating a planar curvature in the direction orthogonal to that of the cutting plane at discrete points. Furthermore the toolpath generated by contouring is suitable for 3 axis milling but is inappropriate for 5 axis milling.

But in this article an algorithm for extraction of adaptive isocurves has been developed. The resulting toolpaths do not gouge locally and combine the advantages of both prior approaches. Briefly given two isocurves C_1(u) and C_2(u), on a surface S(u, v), one can compute and represent the square of the iso-distance, \(\Delta_{12}^2(u)\), between them symbolically as NURBs or as Bezier curve. Furthermore, given the tolerance \(\delta\), by subdividing the two curves at these parameters, new sub-isocurve pairs, \{C_i^1(u), C_i^2(u)\}, are formed. If the two curves iso-distance is larger than \(\delta\), a new sub-isocurve, C_{12}^i, is introduced between C_1^i(u) and C_2^i(u) recursively until the iso-distance between each pair is smaller than \(\delta\) (fig 1c)

An important observation is that since a middle isocurve is introduced if the iso-distance is larger than \(\delta\) and \(\delta\) is small, there will be only a small variation in the speed in the v direction of the surface S, \(\frac{\partial S}{\partial v}\).
Current practice and research trends

The first note about the current practice is that the toolpath derived, in general, is directly applied to the stock from which the model is to be machined. In some cases, the depth of milling required is simply too large. A rough cutting stage is usually applied in which the excess material is removed crudely (figure 4). Then, in the final stage, when the toolpath derived is applied, it is necessary to remove only a limited amount of material.

Now the trend regards timing considerations. The adaptive isocurve toolpaths for the Figure 2c have been used to mill the complete knight. The knight model consists of a single highly complex NURBs surface. Two fixtures, one for the right side and one for the left side of the knight, have been used. A "house on a hill" model (figure 3), consisting of several trimmed surfaces was used in this example. This model was milled using a ball end tool in 3 axis mode. Figure 3 shows the adaptive isocurve toolpath used in the finish stage of the model. To gain some insight regarding this algorithm, Table 1 provides some timing results for computing the adaptive isocurve toolpaths for the tests displayed. Tests were run on a SGI4D 240 GTX (R3000 25MHz Risc machine). The surface in Figure 1 is a B-spline ruled surface with 3 Beezier patches. The knight (figure 2) is a far more complex NURBs surface. Its 56 Beezier patches account for its long processing time. As you can see the house model require much more CPU time than the knight and the surface model.

My comment on current research and practice

I think that one of the characteristics of these adaptive isocurves is that the algorithm used to extract them frees the user from the need to determine either the surface parameter spacing or contour plane spacing and the direction to use to insure adjacent isocurve distances produce a valid coverage. Instead, a bound on the required distance between adjacent sub-isocurves must be specified, and guaranteed automatically. This implies that the user or the engineer will save time to prepare milling a certain surface. Furthermore, it is more optimal than the previous
methods in that the resulting toolpath is shorter and it provides a direct quantitative bound on the resulting scallop height.

Another aspect than I retain important is that the algorithm has also been used to compute gouge avoiding toolpaths for automatically milling freeform surfaces without requiring the introduction of auxiliary check and drive surfaces. And also in this case the advantage is in time saving.

Furthermore the adaptive isocurve extraction method for freeform surfaces introduced in G. Elber and E. Cohen. *Adaptive Isocurves Based Rendering for Freeform Surfaces* (see reference number 9) was used as the foundation for the nearly optimal toolpath generation method described. This algorithm eliminates most of the redundancy that occurs when equally spaced complete isocurves are used as toolpath, while retaining all the advantageous properties of isocurve toolpaths.

Looking at the definitions the toolpath is exact, easy to trim and almost optimal. The use of this algorithm in three-axis milling has been demonstrated in the article but the adaptive isocurve extraction algorithm can be used to generate toolpaths for 4 or 5 axis milling, virtually unmodified. Anyway multi-axis milling operations require additional information to orient the tool.

**My vision on the future development of this research field**

A coverage based on adaptive extraction of isocurves can be used for various purposes from toolpath for machining purposes to image rendering. It changes in a significant way the role of numerical machining which is not only to improve machining accuracy, but also to shorten the product realization cycle, and to compensate for the diminishing of the skilled workforce. High accuracy and high efficiency are both important in the NC machining.

Unfortunately in for the 4 or 5 axes milling operations it’s difficult to detect gauges and accessibility to locations thus research is required. This new algorithm will be useful not only for milling but also in special inspection, when time is a critical constrain or when the support of robots is needed. The future research of this field will regard the development of new methods to mill, or cut, pieces in fastest way also for 4 or 5 axes milling operations.

Anyway new research areas in this sense involve rapid prototyping, layered fabrication, simulation in immersive environments algorithms, representations, and high level design operators for complex models in computer graphics and so on.
Since these last years adaptive isocurves are used also for Line Art Rendering. Adaptive extraction of isoparametric curves is employed here to create illustrative imagery exploiting different shaders. This image is a simple example. Thus adaptive isocurves could be used in the future also for high quality and high level design.
Second article

Path planning and control for AERCam, a free-flying inspection robot in space

Review of the article

This paper describes a prototype and the necessary path planning and control for space inspection applications. The robot is the first generation of a free-flying robotic camera that will assist astronauts in constructing and maintaining Space Station. The robot will provide remote view to astronauts inside the Space Shuttle and future Space Station, and to ground controllers. The camera is called AERCam (Autonomous Extra-vehicular Robotic Camera).

The path planning algorithm developed for the AERCam in this article is based on a geometric structure called roadmap which has the following properties: accessibility, connectivity and departability. A robot uses a roadmap to plan a path by finding a path from the start onto the roadmap (accessibility), then along the roadmap (connectivity) and then from the roadmap of the goal (departability). Once the robot determines the path from the roadmap, the path is then optimized for fuel usage.

The first roadmap used in this paper is the Generalized Voronoi Diagram (GVD) which has been commonly used in the motion planning field. To define the GVD diagram, assume the robot is a point operating in a workspace W which is populated by convex obstacles C₁,.., Cₙ. The distance between a point and an obstacle is the shortest distance between the point and all the points in the obstacle. The robot can use the GVD for motion planning. Given an arbitrary start location for AERCam, there always exists a path from the start to at least one point on the GVD. Essentially this path directs AERCam away from the closest obstacle until it encounters another obstacle which is equidistant to the first closest obstacle. Then the path is determined via a graph search of the GVD: the robot traverses an edge until it encounters a meet point. Here the robot search branches and another edge is traversed. If the robot encounters an already visited meet point, there is a cycle in the graph and again, the robot returns to a meet point with an unexplored edge associated with it.

This concept has then been applied to 3D space with the name Generalized Voronoi Graph (GVG) (see article 3 for more informations) (figure 5).
Since the planar AERCam uses its trusters to move, we are interested in minimizing the fuel usage, making the mission more cost-effective.

The scientists at NASA Johnson Space Center concluded that a sequence of straight line paths from start to goal suitably approximates an optimal solution for fuel usage (figure 6). Furthermore the optimization procedure will produce paths that bring AERCam unacceptably close to obstacles, so the operator specifies a safety parameter describing the minimum distance to an obstacle AERCam is allowed to achieve. AERCam has a ring of twelve infrared sensors that give a distance to obstacles around AERCam.

**Current practice and research trends**

The main steps for the development of the AERCam are:

- May 1996 - Conducted Sprint Preliminary Design Review
- May 1996 - Began fabrication of Sprint system (1st Generation)
- June 1996 - Conducted initial Sprint Critical Design Review
- Mar 1997 - Select target hardware and operating systems for free flyer and control station
- Jun 1997 - Complete pose estimation software for first target
- Jun 1997 - Complete low-level station keeping image processing and image acquisition
- Jun 1997 - Complete development of control station intelligence architecture
- Jun 1997 - Complete development of free-flyer intelligence architecture
- Sep 1997 - Relative motion estimation software available for integration
- Sep 1997 - Acquire custom VLSI Laplacian-of-Gaussian and binary correlator chips for compact stereo vision system
- Sep 1997 - Perform AERCam II Integrated Ground Demonstration mission scenario with integrated perception technologies.
- Sep 1999 - Perform AERCam III Integrated Ground Demonstration

Future work on this project includes using the GVG to localize the AERCam, which would be useful in the case of GPS loses tracking. The localization procedure draws from the previous work of the authors’ in mobile robot localization. Essentially somebody will exploit geometries encoded in the GVG to determine landmarks that the robot can use for localization.
Another avenue of future research is using the method of GVG to plan optimal inspection paths for AERCam. The next generation of AERCam, AERCam III, will increase the intelligence of the system and transition much more of the AERCam II off-board processing (such as vision processing) onto the free-flyer. Accordingly, miniaturization of hardware will continue to be a major focus. In addition, AERCam III will provide the ability to interface non-visual sensor payload(s) with the free-flying platform. AERCam III will focus on technologies necessary to perform autonomous inspections once at the site. The AERCam III will be used by the Space Shuttle, the International Space Station (ISS), or a transfer vehicle to the Moon or Mars. It will also improve the robustness of AERCam II systems and address technical difficulties encountered with AERCam Sprint (1st version of AERCam) and AERCam II.

The near term objective is to develop perception, navigation, and intelligence technologies and test them as part of an Integrated Ground Demonstration. For the perception technologies, technical advances will be made to allow operation in the wide range of lighting conditions experienced in space. Additionally, advances will support recognition and tracking of objects of interest that are partially occluded. Technical advances will be made in the navigation technologies in the determination of a relative position and attitude of a spacecraft. A significant challenge to be addressed by the intelligent software is the desire to use a common architecture for both autonomous and teleoperated control modes as well as to smoothly transition between these modes. The system will be required to perform many of the operations in real-time. Data from multiple sensors will require validity checks and must be fused in a manner that resolves any conflicting information.

**My comment on current research and practice**

Certainly this is a relatively new area for robots: until now the robots are teleoperated thanks to a user interface that contains a six-degree of freedom joystick but in the future an independent and autonomous job of the robot is required. Key points inside the Station can be memorized in the robot and as soon as a breakdowns are detected the robot can move automatically, and identifying the minimum fuel usage path, toward the required place for repairing. Actually the paper is relatively old (May 1999), since the technology in this area is moving really fast, and some details for the AERCam are not included, anyway the current AERCam, AERCam II, is already working at NASA. This is the symbol that who wrote the article was actually working on the project. It wasn’t a futuristic project, it was a REAL project!
I think that the really fundamentals skills implemented in AERCam II include trajectory movement, discrete movement, station-keeping, teleoperation, obstacle avoidance, landmark recognition, and visual inspection. After the AERCam has collected imagery, image comparison (comparing images of the same area acquired at different times) is performed to highlight possible problem areas. Images will be registered to form a mosaic, and changes subsequently detected in the segmented regions will be reported to the user. Therefore real-time computation is required. Anyway I think that the main characteristics of this robot can be summarized like that:

- Teleoperating the AERCam using a joystick. With its interface we can decide to move the AERCam to a certain goal position by clicking the mouse on a specific map of the station in a laptop. This is of particular importance because at any time for example AERCam can be commanded to halt: if something goes wrong with this robot we can stop the robot for further investigations.
- Teleoperating the AERCam using voice command. The astronaut for example could call the robot just calling it (also if there’s no sound in the space). The signal relative to his voice would be elaborated and converted to an impulse for the AERCam to move to the desired position. Also in this case it can be commanded to halt.
- Tracking a moving human. This is a really wonderful application because we can program it for example to follow an astronaut in his movements showing to Houston or to the other astronauts what’s he doing. This allows the astronaut to use both hand for handling tools necessary for maintenance.
- Obstacle avoidance while navigating to a specific position. We don’t need to care about acceleration, obstacles… etc because AERCam will automatically stop if the distance between it and an obstacle will reduce to less than a certain tolerance.

These 4 characteristics give to the AERCam an incredible flexibility and ease of execution of the movements. I retain that NASA in this case is investing in giving a better life to astronauts, for better supporting them from the earth and monitoring easily what’s exactly happening in the station. With AERCam NASA is trying to improve the Space Station, to develop space travel and space tourism. Monitoring the Station is the simplest way to guarantee safety and reliability. And AIRCam could be the next supervisor, the next “BIG BROTHER” of the Space Station.
My vision on the future development of this research field

Regarding the future I think that a lot of research is going to be done in this field. The ISS needs not only inspection but also maintenance. Some works can be very difficult for astronauts because there is no space for their body in some small inaccessible area of the station or because sometimes the object to repair is in a really dangerous position. Computational Geometry is again useful because with the GVD provides a valid instruments to develop new robots capable to even substitute the work of men in the future. The ultimate goal for the AIRCam is to have a system that can be stored in a carrier, just like putting a car in a garage, and that can then be commanded to fly to a specific location on the parent spacecraft to perform an automated inspection. The system will autonomously fly to the commanded location, perform the inspection, and intelligently screen the data to determine if human intervention is required. From the NASA projects I can summarize that progress toward ultimate goal will be made in three phases. In its final configuration, AERCam will have the capability to:

- autonomously fly to a commanded location (such as a desired viewing position for a point of interest) while avoiding obstacles,
- autonomously fly search patterns for visual or non-visual inspections,
- intelligently screen inspection data to determine if further human analysis is required, and

The goal of AERCam II and III is provide teleoperational capabilities as a backup to autonomy, the user will be to monitor and control free flyer position and attitude with respect to the parent vehicle or with respect to an object in the field of view of the cameras. The autonomous functions will include the ability to detect and avoid obstacles. Anyway I think that in next 10-15 years the approach of NASA will be directed to produce not only AERCam more and more efficient but real autonomic robots able to repairs damages to the station.

Of course the use of GVD is particularly useful for internal spaces in the station, but in the future this robot could be designed to work also outside the station. A camera correctly mounted on it and specific tools could preserve the astronauts to risk their life for some repairing or maintenance. In this case a teleoperation would be useful for a better control of the work done by the robot. Actually now the development of the robot is limited by the fuel cost. In the next future everybody hope for an increasing use and production of hydrogen. This element not only provide a lot of energy per mole but
also is really light. This implies that the AERCam and other future maintenance robots will be completely independent, thank to their own internal tank for hydrogen and its lightness.

I personally think that this kind of technology will have increasingly application in space and also in some underwater difficult operations, where the conditions are pretty similar to the space. Until now its use is relative to maintenance but in the future this system could even save lives of astronauts or recovering spacecrafts.
Review of the article

Unlike conventional robots, serpentine robots possess multiple and redundant degrees of freedom which, while enabling unusual agility, make a serpent robot difficult to manage. Effectively, planning a path for a serpentine robot is akin to searching a multi-dimensional configuration space between a start and target snake configuration.

The approach in this article is to adapt the structure of a rigorous motion planning scheme to a sensor-based implementation based on a geometric structure, termed a roadmap.

A roadmap called the Generalized Voronoi Graph (GVG) has been introduced in this article and in this work the author used the results of the GVG approach: the serpentine robot is no longer just reaching a desired location, but it is following a path to get there given by the Generalized Voronoi Graph. In m-dimensions, a Generalized Voronoi Edge is the set of points equidistant to m obstacles, such that each point on the edge is closer to the m obstacles than any other obstacle. A GVG edge is one-dimensional and GVG edges intersect at Generalized Voronoi Vertices. A Generalized Voronoi Vertex is a point equidistant to m + 1 obstacles such that no other obstacle is closer to the m + 1 obstacles. The GVG is the collection of GVG edges and vertices (see figure 7).

Edge tracing continues until one of two conditions are met: (1) the robot encounters a Generalized Voronoi Vertex, or (2) the robot hits a boundary point. A Generalized Voronoi Vertex is sometimes called a meet point because that is where three GVG edges meet. At this point, the robot is equidistant to three obstacles. Now, the robot notes the approximate location of the meet point and continues tracing a new GVG edge until it either reaches another meet point or a boundary point. A boundary point is where a GVG edge intersects the boundary. At this point, the robot simply turns around and returns to a previous meet point that has unexplored directions. Once all meet points have all directions explored, the robot terminates its exploration process. Effectively, the GVG generates a sequence of backbone curves that bring the serpentine robot from a start configuration to a goal. However, it is possible the robot cannot be fitted to back-bone curves generated by the GVG, so the GVG path must be deformed to accommodate the serpentine's mechanical limits. Typically, a
serpentine robot is segmented into kinematically sufficient "bays." This means that each bay can arbitrarily position and orient its end-plate with respect to its base. A fitting procedure is used to place and orient each bay along the entire backbone. Motion planning for a serpentine robot is achieved via a *follow-the-leader approach*. The head of the snake moves along the GVG, while the rest of body follows (figure 8). Effectively, the GVG generates a sequence of backbone curves that bring the robot from start to goal. Sometimes, this sequence is termed a *backbone curve-path*. We assume that if the head is not already at a boundary point, that it is possible to advance the snake such that the head moves an increment $\delta s$ farther along the graph. In other words, we advance the robot along the GVG by adding a $\delta s$ increment from the GVG to the current backbone curve, thereby forming a new backbone. Unfortunately, the initial guess of the backbone curve-path may contain some configurations to which the mechanism cannot be fitted. Joint limits and finite bay length prevent the serpentine from following all the backbone curves exactly. So, although the backbone curve-path is guaranteed to be a collision free path, a discrete segmented approximation of the curve may intersect obstacles. In this case the backbone curve must be continuously deformed to satisfy joint limits and avoid obstacles.

**History, current practice and research trends**

Robotics engineers have already produced serpentine robots for various applications. At the Tokyo Institute of Technology they built the first serpentine robot and studied how such mechanisms can locomote in the plane. This serpentine robot was also mobile.

Research serpentine robots in the United States began with the hyper-redundant manipulator built at the California Institute of Technology where they also introduced some new snake robot locomotion theory, as well.

Takanashi of NEC developed a serpentine robot for the purposes of search and rescue for survivors in collapsed buildings. In cooperation with Takanashi, engineers at the Jet Propulsion Laboratory (JPL) adapted the NEC design and built the JPL Serpentine Robot for use in space station inspection.

Currently, the JPL Serpentine Robot is being used to investigate bridge inspection. Since serpentine robots can reach more locations at more approach angles in convoluted environments than
conventional robots, they provide enhanced capabilities for bridge inspection, which is a costly and dangerous operation. Serpentine robots can reduce the cost and virtually eliminate the danger, while at the same time minimizing traffic delays that are a result of bridge inspection. So, instead of having a person crawl under a bridge or hang from a supported bucket, a serpentine robot can thread its way through the truss structure of a bridge while providing views to the inspector on the roadbed.

In medical field, serpentine robots offer high utility for minimally-invasive surgery. This type of operation virtually eliminates the need to open up large portions of the body to perform surgery, allowing for quicker recoveries and hence shorter hospital stays. In 1994, roughly one third of the 21,000,000 operations performed in the United States could have been achieved using existing minimally invasive surgery techniques. This estimate does not factor in that workers would sooner return to their jobs because of the significantly shorter recovery time. Current minimally invasive technology can only reach superficial portions of the body; serpentine robot can extend this reach to allow for minimally invasive operations throughout the body.

The Pacific Northwest Lab has designed and built a snake robot for the Naval EOD Technology Division in Indian Head, VA. This snake will be used for surgical disarming of bombs. Just as a surgeon can use an arthroscopic surgical instrument to make repairs deep inside the body, without causing damage to the surrounding tissue, a technician can probe the internals of a bomb without accidentally detonating using a serpentine robot. This application requires that the robot also explore a completely unknown three-dimensional environment. Three-dimensional sensor based planning is necessary for snake robots to assist in the search and rescue of victims in collapsed buildings where access is limited and the geometry of the environment is difficult to evaluate.

My comment on current research and practice

I think that we can achieve start-goal motion planning in two different ways: (i) goal location is known and (ii) goal location is not known (we can call it inspection).

For the first point the solution is relatively easy, the known goal search uses a depth-first search heuristic to reach a goal with known coordinates: each time the robot encounters a meet point, it chooses a new edge to explore which locally minimizes the robot's distance to the goal. If the robot encounters a boundary point, then it back-tracks to the previous meet point with an unexplored direction, and explores a direction associated with it. This graph search of space continues until the robot is within line of sight of the goal, or all meet points have no unexplored directions associated with them.
For the second point the solution is much more complex. If the coordinates of the goal location are not known, then the simulator explores the GVG until the goal is within line of sight. And actually I think that this situation is the most common: examples are when we want to search for trapped survivors in search-and-rescue operations or to inspect a complex structure for defects. In this case we usually assume that the robot can identify the goal at a distance by sight. And actually this should be the main field of research for the serpent robots: they can provide strong force, great flexibility and at same time relatively high speed. I think the world market is ready for these kinds of robots and a lot of industries, universities and scientists are looking forward for them.

I think that one of the main problems with this solution is the possible use of the coarse discrete approximation cutting corners. This requires a means to detect obstacle collisions in the discrete approximation and a process for deforming the initial backbone away from the problem area. A proper solution to the joint limits problem involves the global optimization of the backbone curve with an upper limit on the backbone's curvature. I think the future research in this field will be addressed to develop a technique using the calculus of variations to optimize the backbone curve for cost functions involving length, curvature, safety, and energy. I guess at this point that they should plan to use the GVG as a first approximation to the backbone curve, then iteratively deform this seed curve while considering numerous obstacle constraints, and a limited maximum curvature.

My vision on the future development of this research field

Despite the distinct need for serpentine robots and the advantages over traditional mobile robots (like flexibility and reachability), I believe that they are not commonly used. I think the reason is because there were previously no adequate control strategies to direct them. Therefore, this work employs a roadmap termed the Generalized Voronoi graph. Using the Generalized Voronoi Graph for serpentine robot motion planning has several advantages. First, it can be used in a sensor based way, without a priori knowledge of the environment. Second, using a follow-the-leader approach to define backbone curves through the environment, the computing cost associated with a highly redundant manipulator can be greatly reduced. Computational efficiency allows for real time control without full
knowledge of the environment. Savings is achieved by not having to perform the manipulator inverse
kinematics, not having to compute to configuration space, and by reducing an m dimensional
environment to a one dimensional roadmap search space.

Furthermore I retain that an important aspect of this technique is that it can be generalized to three
dimensional environments without a significant increase in computation. The fitting procedure for a series
of discreet links along a continuous backbone curve is no more computationally expensive than in the
two-dimensional case. The time required to compute and update a snake configuration in a three-
dimensional environment is of the same order of magnitude as in the two-dimensional case.
And yet using the GVG already maximizes safety for a given route through the environment by
keeping the snake joints as far away from the walls as possible. Optimizing safety is desirable in many
real applications such as navigating under the rubble of a collapsed building. Although using the GVG
ensures that the backbone curve is placed midway between all obstacles and never intersects with
corners, a real snake configuration using links with a high length to width ratio will fit to a coarse
approximation of the backbone. This leads to the possibility of the coarse discrete approximation
cutting corners.

Finally I think that the ultimate goal of this work is sensor based planning for snake robots in
unknown environments: most of the time we would like to inspect a certain unknown area. At this
point the method just developed seems to fail but it’s enough to use a range sensor on the snake head.
Once we can access distance to nearby obstacles from such a sensor suite, then the GVG-based
method for determining the backbone curve can be implemented in a sensor based fashion. But of
course sensors along the body of the serpentine robot to allow for fine-tune adjustments using a
local sensor based planner during the follow-the-leader mode are needed.
Figures for each article

First Article:

**Figure 1:** (a) *Isocurves* are obviously not an optimal solution as a toolpath for this surface. (b) *Contouring* with equally spaced parallel planes is more optimal but is piecewise linear. (c) *Adaptive curves* are, in general, more optimal, exact and compact.

**Figure 2:** (a) Toolpath using *isocurves* will be not optimal in this complex surface. (b) *Contouring* with equally spaced parallel planes is too sparse in coplanar regions. (c) *Adaptive isocurves* are more optimal, exact, and still correctly spans the entire surface
Figure 3: Adaptive isocurves toolpath for the “house on the hill” model

Figure 4: Parallel plane contouring is used to generate 3 axes pockets for rough cutting
Second article:

**Figure 5:** The AERCam determined a path from a start to a goal location by accessing the GVD, traversing the GVD and then departing the GVD.

**Figure 6:** Using the GVD, AERCam determine a path as a sequence of straight line segments from a start to a goal location.
**Third Article:**

*Figure 7:* Generalized Voronoi Graph (GVG) whose set of points are equidistant to \( m \) obstacles.

*Figure 8:* Movement of the serpentine along the Generalized Voronoi Graph.
References for each article

First Article:


Second Article:

**Aaron Hurst:** Towards Sensor Based Coverage with Robot Teams. ICRA 1992: 961-967


**David Kortenkamp, Marcus Huber, Charles Cohen, Ulrich Raschke, Clint Bidlack, Clare Bates Congdon, Frank Koss, Terry E. Weymouth:** Winning the AAAI Robot Competition. AAAI 1993: 858-859


Third Article:


