Polymorphic Anisotropic Metric for Surfaces

Aditya Kusupati
Guide: Dr. Pierre Alliez
TITANE Team
Inria Sophia Antipolis

Report for Internal Purposes

Introduction

Meshing is one of the basics of modelling and image processing. Every surface or object can be divided using various meshing techniques for purposes like rendering, 3D printing etc., There has been significant improvements in past few years. The processing times highly depend on the accuracy, shape and number of meshing polygons used.

This work done is continuation of work for increasing efficiency in generation of metrics for advanced meshing techniques. It mainly focuses on generation of polymorphic anisotropic metrics for various surfaces. The work was significantly progressed for Rectangular metrics which are a subset of polymorphic anisotropic metrics.

Previous Work

Bertrand Pellenard, in his PhD paper proposed an approach for automatically generating isotropic 2D quadrangle meshes from arbitrary domains with a fine control over sizing and orientation of the elements. At the heart of the algorithm is an optimization procedure that, from a coarse initial tiling of the 2D domain, enforces each of the desirable mesh quality criteria (size, shape, orientation, degree, regularity) one at a time, in an order designed not to undo previous enhancements. The experiments demonstrate how well our resulting quadrangle meshes conform to a wide range of input sizing and orientation fields

I started with the iterative diffusion approach used by Bertrand. The method of Bertrand was the following:

1. We have a standard algorithm to find the largest axis aligned rectangle for a given contour of points and a inside point(query point)
2. The standard algorithm is used as a part of the Computational Geometry Algorithms Library(CGAL)
3. We discretize the rotation space of 2D plane and then rotate the entire point set about a fixed angle in the discrete space
4. At every iteration(rotation) we find the largest axis aligned inscribed rectangle with the query point as the origin
5. After all the space is searched we take the best(largest of all) inscribed rectangle for a given query point

This approach has to search entire space even after getting to a conclusive answer. So this is a bit slow. My experiments have shown that for a standard rectangular contour generated randomly using 1000 points including the generation of the point set and processing to get the aimed largest inscribed rectangle the average time was around **3.5 seconds**. This had to be improved as it is too much for a general processor to ask for in normal processing.

Aim

Dr. Pierre Alliez advised me to approach this problem using stochastic approach in order to fasten up the process. The aim was to devise a stochastic algorithm for finding out the largest inscribed rectangle given a point set which form a contour and a query point which has to be inside the rectangle obtained. We tried to randomise as many degrees of freedom as possible considering even the interlinked dofs into account.

Algorithm

The stochastic algorithm is as follows for the rectangle case(I am writing the general and final take away version written by me the progress can be found in the code files as it is in the name of old functions with intuitive naming):

1) Take the query point as the reference and start the process
2) Generate a random angle from 0 to 2π
3) Create two mutually perpendicular axes rotated by the random angle with the query point as the origin(just get the directional vectors)
4) Consider the maximum extent on both sides of one of those directional vectors in point set, thus we get the maximum range till which the rectangle can be formed to inscribe the given point set
5) Choose random extent between the query point and the extremums and get two reference lines to make the further computations
6) We find the closest point to the other directional vector in the extent covered by the new reference lines
7) We now have two parallel sides of the aimed rectangle which are parallel to the initial directional vector and pass through the closest point to the reference axis along the same direction
8) Using the similar technique as above but using the finalised sides as the reference lines generate the same for the other perpendicular direction
9) After this we have the rectangle generated using stochastic approach

Note: English might be misleading please go through the code in Stochastic.h while trying to understand

The above stated algorithm was to stochastically generate an inscribed rectangle which is supposedly the largest with the chosen random angle. Now we shall use this algorithm repeatedly to find the largest inscribed rectangle with a high accuracy.

Stochastic Part

In contrast to the discrete space approach the angles chosen till we get the desired answer are random and lead to probabilistic chances of getting the right answer(with the aimed accuracy). The approach is as follows:

1) In the first part where we script to find out the average and the worst case generations required to get the rectangle with required accuracy
2) We take a contour of a known rectangle, as we already know its area(there is a functions to do all this in Random_generator.h)
3) So what we do is script till we get the rectangle of the required accuracy which can be changes in the calling functions
4) So after scripting we get the average, worst case number of generations required to get the best possibility with a given accuracy, we shall need this data in future
5) Now we start experimenting with our algorithm, just using the rectangle generator for the worst case number of times and taking the best out of them without involving the known area of the contour
6) Thus we can check the overall accuracy of the algorithm using this method

After this method what i found out was that using the worst case number of generations was working with almost 99% accuracy in achieving the rectangles with the area with the stated accuracy with respect to the actual area.

This method is pretty fast and can solve the problems asking for an accuracy of around 95% of the actual area existing. The time required for the entire process with 10000 generations was around **0.75 seconds** including the generations which was a significant growth. The average generations required till 96-97% accuracies are around 1000 so this fitted in appropriate fashion but when we jump to 98-99% and above accuracies then there was exponential growth the in the generations required which indirectly affected the generations we need to make to find the correct answer using the stochastic approach. Now we use the Diffusion part to overcome this exponential growth.

Diffusion part

After using the above algorithm to get to the best possible rectangle around 95% of the actual area we start using the diffusion to get to 98-99% accuracy most of the times. The keen idea is that when we strike the rectangle with an area around 95% it is likely that there exists a local maxima(hopefully a global) in terms of area of the inscribed rectangle around that region of random angle generated.

We use this key idea to capture the required rectangle. As and when we get the best rectangle for say 10000 generations we start applying diffusion in both the directions just by discretizing the space same as Bertrand's approach but we proceed only till when the area is promisingly increasing and comes to a halt. We apply the same technique on both the sides of random angle at which we got out base rectangle for diffusion.

After scripting using this approach I have found that 95% of cases we reach around 98-99% of the actual area and the extra diffusion is nothing but generating a stochastic rectangle at the angle obtained after diffusion. the time for each query rectangular contour averaged out to **0.80 seconds** which is pretty good when compared to the exponential growth.

This techniques actually can be used even to get to 95% from 90% thus further reducing the generations. The major problem is that most of the times they turnout to be local maximas, whereas when we get to 95-97% they with high probability will diffuse to get to a local maxima with an area better than itself.

This approach can be further improved using better diffusion techniques and known distributions of areas of quads in the entire space while generation.

Other works

1) Can use various shape generators in the Random_generator.h for many testing purposes which involve contours with point sets and a inside point which needs to be our query point
2) The functions in Stochastic.h do the stated functionalities for the rectangle case
3) Similarly I have designed a crude form of stochastic algorithm for triangular metrics(which can be used for conical surfaces) which works perfectly till an accuracy of 65%
4) The algorithm for triangle can be further improved and diffusion can be employed thus we could have triangular metrics at our disposal using stochastic algorithm
5) Write all the function in window.cpp and use them to run all the required process using Qt as UI
6) As an extension to rectangles we can easily modify them to get to rhombus metrics(used for hyperbolic surfaces) just by considering non perpendicular axes about the query point which requires two random angle generations

Conclusions

Stochastic approach scales down the required times for processing of metrics. This can help a lot in meshing algorithms in future. I hope in future if not me someone will carryout the work and complete all the possible anisotropic metrics for surfaces and thus easing the process of Modelling and image processing.

Deliverables

Along with this report you need to get all the source code for continuing the further work. The source code tar.gz file shall be there always in the drive link provided to Dr. Pierre Alliez, the folder has few data files for reference purposes, other files were shared on forge and have expired unfortunately thus we need to rerun the scripts to get the data and the graphs of the iterations that I have stated above. It has a Readme.txt which shall contain brief summary about the files and their functions. The code is intuitively named and commented mostly for the ease of understanding. In case of any doubts feel free to contact me at my email-id with Dr. Pierre Alliez.

References

1) http://www-sop.inria.fr/members/Bertrand.Pellenard/doc/imr12-paper.pdf
2) http://www-sop.inria.fr/members/Bertrand.Pellenard/doc/imr11-paper.pdf
3) CGAL Manuals - http://www.cgal.org/
4) Stochastic geometry for image analysis. imulation et optimization. Chatelain F., Descombes X., Lafarge F., Lantuejoul C., Mallet C., Minlos M., Schmitt M., Sigelle M., Stoica R. and Zhizhina E. *Wiley/Iste,* Ed. X. Descombes, 2011