UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2021

Title: SECTION VIEWS FROM 3D LASER SCANNED BUILDINGS

Author: CLAUDIOUS NHEMWA

Project Abbreviation: SCAN3D

Supervisor(s): PATRICK MARAIS

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	10
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	5
System Development and Implementation	*0*	*20*	10
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Feature Line Extraction using tensor voting algorithm

Claudious Tirivashe Nhemwa ,NHMCLA003
University of Cape Town

ABSTRACT

Heritage sites are vulnerable to natural disasters such as hurricanes and volcanoes. These lead to the deterioration and the destruction of heritage sites. The preservation of heritage sites has become one of the most important tasks for further processing and analysis of 3D models.This has led to the crucial need to scan heritage sites into 3D models for preservation and analysis. The Zamani project is a team that scans heritage sites around the world creating a large database of heritage sites. In this paper we investigate the suitability of the normal tensor voting algorithm in feature line extraction of 3D models of heritage sites. In the analysis of 3D models, the extraction of feature lines has proved to be an important task as it allows further processing such as mesh re-meshing and detection of changes in the sites over time. Scanned 3D models of deteriorated sites contain rough surfaces and noisy vertices which makes feature line extraction of these models difficult. This problem suggests the need for a robust and efficient method in feature line extraction. The normal tensor voting algorithm is robust and efficient to noisy vertices hence its suitability to the extraction of feature lines on 3D models. In this paper, we implement the tensor voting algorithm and test it on various models.

1 INTRODUCTION

The growth of the 3D modeling industry driven by innovation has led to numerous ways of creating 3D models. The use of high-precision 3D scanners and cameras has proven to be one of the best ways to scan existing structures. There are different 3D scanning methods such as Photogrammetry[12], Long- Medium-Range 3D scanners (LiDAR) [2], and Close–Range 3D scanners. LiDAR scanners are divided into two categories Laser phase scanners and laser pulse-based scanners. Both categories measure the shift or the time it takes the laser beam to bounce back to the sensor. Photogrammetry combines multiple 2D images forming a 3D model.

The Zamani project uses these techniques to scan heritage sites to preserve them for further analysis. It undertakes data collection on various heritage sites around the world and stores these models in a database in which the public and industry expects can access. The models are used by expects to analyse the sites and deduce information without visiting the heritage sites . One of the recent sites that the project worked on is the Siq of Petra[15] where aerial photogrammetry techniques were used to generate the model. Heritage sites help us to understand our past so that we can plan for the future. These sites are vulnerable to wars , natural disasters and poor tourism practises hence their conditions are often deteriorated . To analyse these models there is a need to implement a method like feature line extraction that simplifies the process.

Feature lines are powerful descriptions of the model as they show the shape of the whole structure ignoring other qualities of the model. These feature lines allow further processing of models such as surface segmentation[7], mesh re-meshing[1], and hole filling[17]. Feature lines show the edges or boundaries they make it easier to analyse changes of models over time. In addition feature lines help in the feature detection of model.

Figure 1: The Rock churches of Lalibela by the Zamani Project [13]

In this paper we implement the normal tensor voting algorithm by Wang et al [17] to extract feature lines on triangular meshes. The aim is to test it's suitability on heritage sites. Since the scanned models contain noise and rough surfaces , the tensor voting algorithm's robustness and efficiency to noise made it suitable solutions. The tensor voting algorithm produces feature lines from the data itself without making any assumptions a so no pre-processing of the mesh is required.

(Section 1).

2 BACKGROUND AND RELATED WORK

Feature line extraction is the generation of salient lines that humans can perceive. These help in the analysis of geometric 3D models. There are two types of ,models point cloud models and mesh models. Point clouds are set of x, y and z points in space while mesh models are a set of triangles that include triangles and edges. In order to detect feature lines , we need to differentiate between sharp edges and smooth edges. Sharp edges are defined as edges where the normal vector change is abrupt while for smooth edges the change is consistent. It is important to note how sharp edges are detected as they are more difficult to detect than smooth edges.

At first feature extraction methods were developed from their 2D counterparts these use first order and second order derivatives[4,

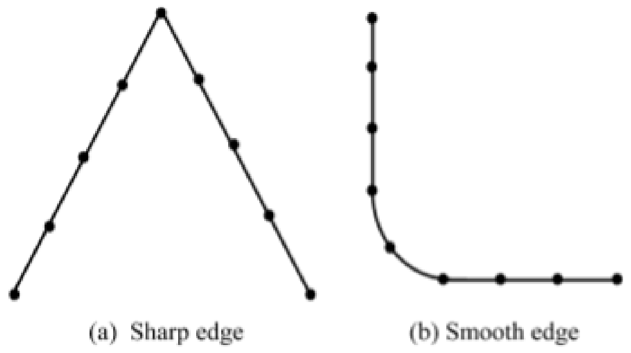

Figure 2: Types Of Edges by Lee et al [8]

11]. These methods though efficient for two 2D images are not as efficient for noisy and rough 3D models. Two significant methods were developed which are surface fitting methods [11] and local estimation methods [11]. The local estimation method by Ohtake et al [11] produces feature lines that are not connected when the shapes are complex however methods such as mesh smoothing and hysteresis thresh holding can be used to improve the output.

Canny et al[3] proposed a method that uses adaptive thresholds where they are adjusted according to the noise ratio. The noise of the model is calculated before computation using a noise estimation algorithm then the Gaussian function is used for smoothing the model.The method uses derivatives on the gradient and the zero crossing method to extract edges.

The tensor voting algorithm proposed by Medioni et al[9] calculates the a matrix using tensor voting. The vertices are classified using eigen analysis. The method is robust and efficient and does not require pre-processing. In order to calculate the vector, the method does not use higher order derivatives but instead the normal of the vector. This method is also widely used in feature segmentation [14] and feature detection[7] . Wang et al[17] improved the method by adding the calculation of salience and the neighbor supporting algorithm to improve it's robustness and efficiency. This means that all sharp edges with weak support from their neighbors are removed as noisy vertices. The results from the algorithm in the paper show positive results on noisy models.

3 METHODOLOGY/DESIGN

3.1 Overview of the design

Figure 3 shows the structure of the implemented system. First, the triangle mesh is read into memory. After the reading in of the model vertices,the normals are calculated and are given as input to the normal tensor voting algorithm. The normal tensor voting algorithm classifies the vertices into corners and sharp edges which are known as feature vertices. The salient measure of the normals is then calculated and used to filter out noisy feature vertices using neighbor supporting. Neighbor supporting is then used to connect feature lines for output where all sharp edges with weak support from their neighbors known weak edges are discarded.In Wang et al [17] proposed method using a pruning algorithm[5] was added to the tensor voting algorithm in order to remove extra noise that might be left after the salient measure. In this paper because of time constrains the pruning algorithm was not implemented. The connecting of feature lines and pruning is considered as post processing as this is done after the feature vertices are calculated.

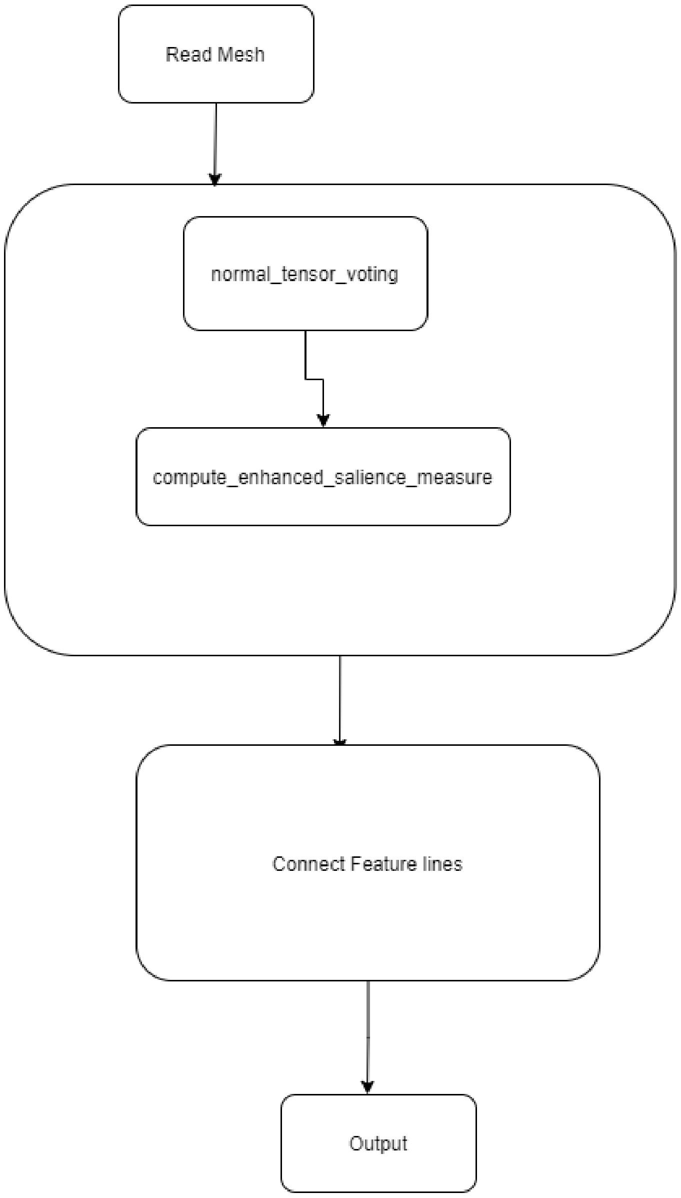

Figure 3: Architecture Diagram

3.2 Software Implementation

The software was implemented using python3 which proved to be a good choice of language because of it's numerous of libraries in 3D modelling. Open3D implemented by Zhou et al [18] was used to handle most of the computation such as reading in models, calculation of normals and creation of the half edges of the triangle mesh. The Algorithms were tensor and array intensive hence the numpy[6]and scipy [16] libraries were used to handle array computation . Lastly, To plot the graphs and the feature lines matlibplot was utilised.

3.3 Dataset

The models used to test the software and the algorithm were acquired free online and scanned heritage sites were acquired from the Zamani project.

3.4 Feature vertex classification

The first stage of the implemented method is reading in model vertices and faces. After reading in the model the data is then used to classify the vertices. Initial feature vertices are detected using the normal tensor voting algorithm. In the tensor voting algorithm, neighbouring triangles define the unit vector of the tensor of a mesh. The covariance matrix $V_v^{f_i}$ of triangle is found by the equation below:

$$V_v^{f_i} = n_f, n_{fi}^T = \begin{pmatrix} a^2 & ab & ac \\ ab & b^2 & bc \\ ac & bc & c^2 \end{pmatrix} \quad (1)$$

where $n_{fi} = (a, b, c)^T$ is the unit normal of f_i.

The tensor of the of vertex v is defined by :

$$T_v = \sum_{f_i \in N_f(v)} \mu_f, n_f, n_{fi}^T \quad (2)$$

and the weight μ is given by

$$\mu_f = \frac{A(f_i)}{A_{max}}. \exp\,(-\frac{||c_{fi} - v||}{\frac{\sigma}{3}}) \quad (3)$$

After applying the tensor voting algorithm ,vertex classification is then done as shown in the algorithm figure 4. The eigen values of the vertices are calculated and used to classify the vertices into face type, corner type and sharp edge type. Sharp edge types and corner types are referred as feature vertices. In face types ,λ_1 is dominant while λ_2 and λ_3 are close to 0. Sharp edge types have λ_1 and λ_2 are dominant while λ_3 is close to 0. Lastly, Corner types $\lambda_1, \lambda_2, \lambda 3$ are approximately equal.

Algorithm 1 Feature vertex classification

```
1:  // $\#(V)$ is the number of vertices
2:  // Face$V$, Sharp$V$, and Corner$V$ are the index sets of
    face, sharp edge, and corner type vertices, respectively
3:  for $i \leftarrow 1$ to $\#(V)$ do
4:      $\lambda_1, \lambda_2, \lambda_3$ are initialized
5:      if $\lambda_3 \leq \alpha$ then
6:          if $\lambda_2 \leq \beta$ then
7:              Face$V$ $\leftarrow$ [Face$V$  $i$]
8:          else
9:              Sharp$V$ $\leftarrow$ [Sharp$V$  $i$]
10:         end if
11:     else
12:         Corner$V$ $\leftarrow$ [Corner$V$  $i$]
13:     end if
14: end for
```

Figure 4: The algorithm was produced by Wang et al [17]

3.5 Salient measure computation

The normal tensor voting algorithm is improved by adding a salient measure step which utilises neighbor supporting. Wang et al [17] suggested this from the observation that the crest point has a maximum curvature in it's principal direction and the crest line follows the direction of the normal curvature of it's crest point. By tracing the principal direction we are likely to detect potential feature lines. However , noise vertices will have to be eliminated if they are no feature vertices in its direction.

In order to calculate the salient measure for sharp edges the initial measure Ω and the integral direction need to be calculated. The formula below is used to calculate the initial measure:

$$\Omega = \frac{\lambda_1 + \lambda_2 + \lambda_3}{2} - \frac{1}{2} \quad (4)$$

The eigenvalues λ_1, λ_2, and λ_3 and their eigen vectors are calculated and these values are used to classify vertices to different types. The Ω gives the feature intensity of each vertex ,with corners and edges having the largest values compared to faces.

According to Moreno et al [10] ,if the tensor is aligned with the tangent of the curve and λ_3 is equal to zero then the vertex must be a curve. Hence t can be equal to e_3, known as the feature direction.

The Salient measure is calculated by the equation below

$$S(v) = \sum_{v_i \in N(v)} W(v_i)\Omega(v_i)$$

with $N(v)$ being the supporting of neighbours and the weight as below

$$W(v_i) = \exp(-\frac{||v - v_i||}{2\delta^2})$$

where δ is 1.5 times the edge length.

3.6 Post- processing

The method used to connect feature lines uses 3 categories. First, if a feature vertex is not connected to other feature vertices then it is treated as noise and removed. Second, two feature vertices that exist on the same triangle are joined in a straight line. Lastly , if three feature vertices exist on the same triangle then the vertices are joined with the centroid of the triangle formed.

The normal tensor voting algorithm and the salient measure algorithm ,filter the noise from the models. Some noise might remain hence extra pruning is required to increase the robustness of the project, This noise usually remains because the noisy vertex is close to feature vertices.The pruning algorithm suggested by Wang et al [17] was proposed by Demarsin et al [5].

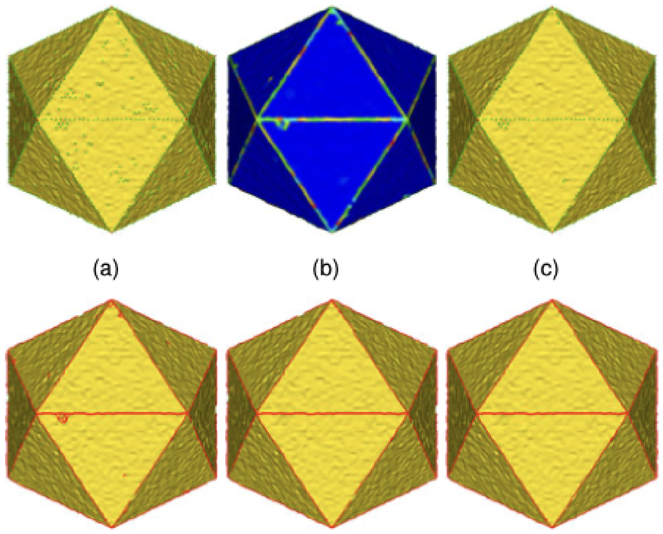

Figure 5: Branch pruning by Wang et al[17] (a) Initial detected feature vertices; (b) Color map of the salient measure; (c) Feature vertices after filtering; (d) Initial feature lines; (e) Feature lines after branch pruning; (f) Final feature lines

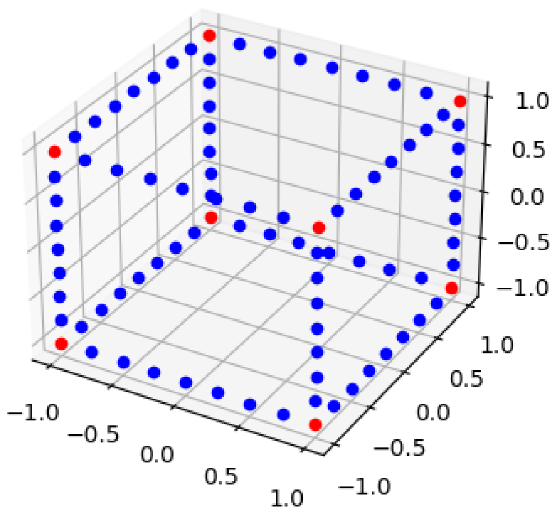

Figure 6: The plot shows the tokens generated by the normal tensor voting algorithm for a cube.

4 EVALUATION

To evaluate the algorithm the method was tested on regular shapes such as cube and cones. After regular shapes more complex shapes such a torus and fan disk model which is both complex and has a rough surface were tested on thea algorithm. Lastly , the algorithm is then tested on heritage sites.

4.1 Noise Free Models

Figure 6 shows the tokens that were generated from a cube model which has 380 Faces and 786 triangles . The normal tensor voting Algorithm generates points that are clear and precise including the corners of the edges. The corners are indicated in the tokens using red dots and the sharp edges as blue dots. After running multiple noise free models, output from the salience measure algorithm and tensor voting algorithm remained unchanged as no filtering was required.

After running the normal tensor voting Algorithm to generate feature vertices and filtering noise using the salient measure algorithm , the neighbor supporting Algorithm is used to join feature lines from the feature vertices. Figure 7 shows the generated feature lines from the 3D cube model .The results show a positive outcome as the shape of the model can be determined without prior knowledge . However, in the corners some additional lines are noticed which might have arose from noise in the corners or the neighbor supporting algorithm not filtering corners well. In addition to the cube more complex models where tested on the algorithm.

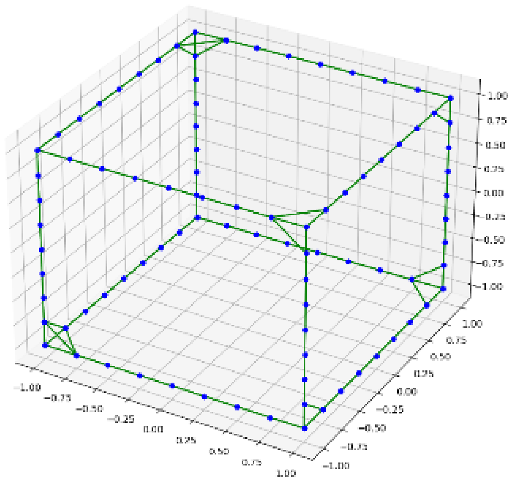

Figure 7: The plot shows the feature lines generated by the normal tensor voting algorithm for a cube.

The torus model was tested on the model which is a doughnut shaped model rotated about an axis. The Algorithm was able to detect accurate feature vertices as shown in figure 8 and feature lines in figure 9. The results show the robustness of the algorithm as it manages to detect the required feature lines. The torus is complex has 1200 triangles and 600 faces. The complexity of the shape makes the tensor voting algorithm generate a large number of feature vertices. These are connected to form feature lines. The results from noise free models show that the algorithm can produce good results on regular shapes and smooth models. Secondly , the salient measure was shown to be unnecessary for smooth noise free models hence if the noise of a model could be pre-calculated the algorithm could skip that part of the algorithm , as a result increasing performance.

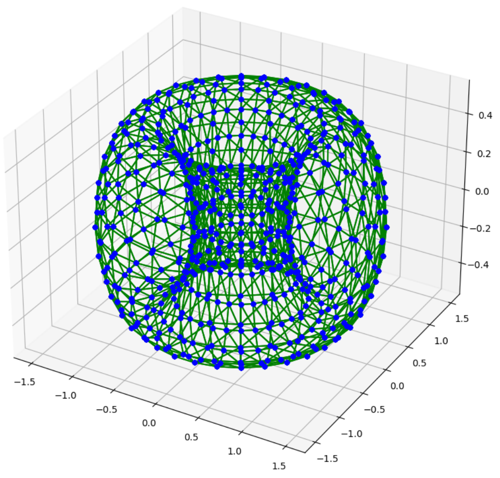

Figure 9: The plot shows the feature lines generated by the normal tensor voting algorithm for a torus.

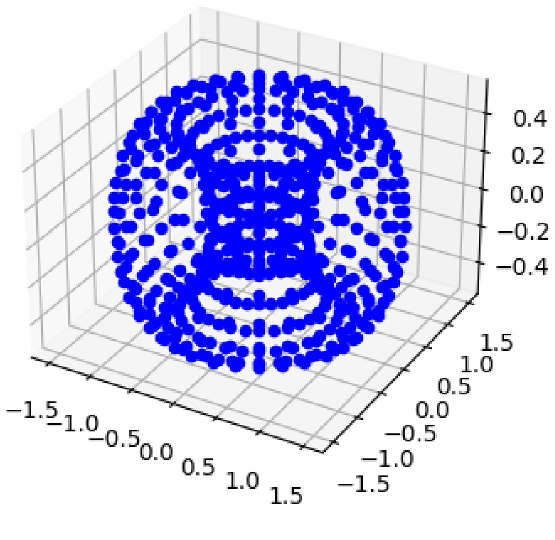

Figure 8: The plot shows the feature lines generated by the normal tensor voting algorithm for a torus.

4.2 Noisy Models

A fan disk 3D model which has rough surfaces and noise in it was used to test the algorithm further. The test was done without smoothing of the model. The normal tensor voting algorithm generates a dense number of tokens in one place as shown in figure 10 . These result shows the noise and the rough surfaces of the model. These unwanted feature vertices are then filtered by the salience computation.Figure 10 shows each stage of the method and the result produced. The salient measure Algorithm filters the noisy feature vertices from the algorithm and only tokens close to edges are left. Some inaccurate noisy vertices which are close to feature vertices are left. These produce some inaccurate lines, however the perception salient feature lines is a matter of perception. After filtering the shape of the model can easily be perceived and is a clear indication of the shape of the model. However, some additional noisy vertices were detected by the model. The pruning algorithm suggested by Wang et al [17] might be useful in removing these noisy vertices. In order to test the algorithm thoroughly there is need to test the model on more noisy models.

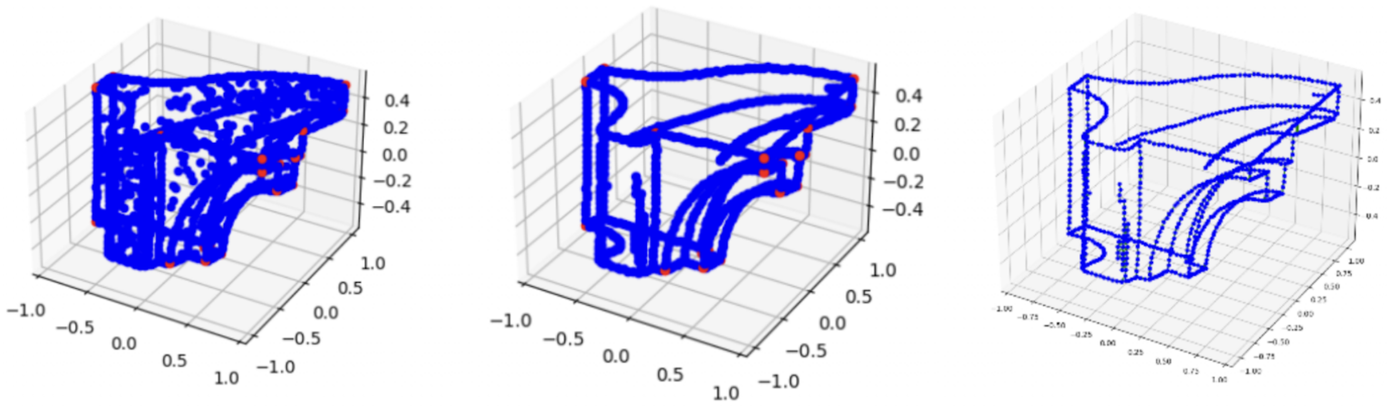

Figure 10: The plot shows output from the tensor voting algorithm ,filtering using silence measure and the feature lines extracted from a fan disk model.

4.3 Heritage Sites

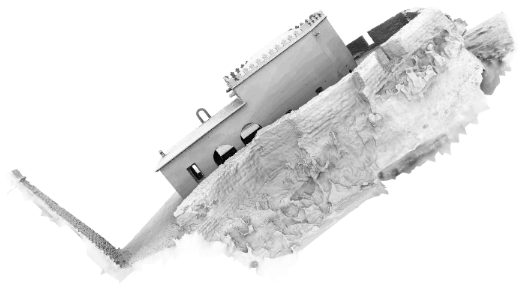

Figure 11: Model of Chapel of Nossa Senhora de Baluarte in Stone Town , Island of Mozambique

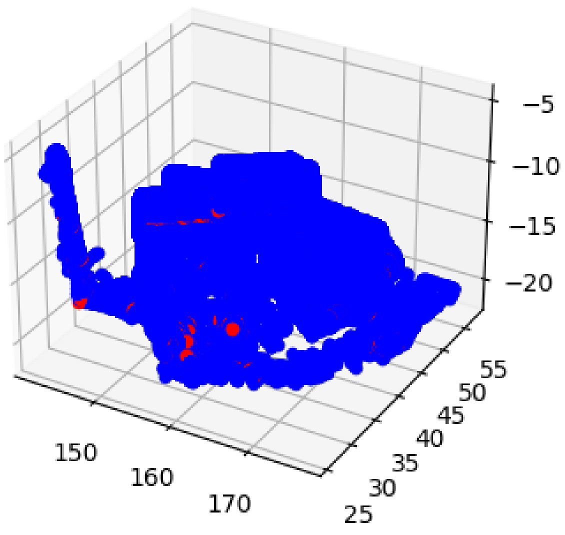

Figure 12: Generated feature vertices of the Chapel

The heritage site model supplied by the Zamani project is a Chapel in Mozambique with 4702858 points and 9403100 triangles. Figure 11 shows the 3D model of the Chapel of Nossa Senhora de Baluarte. This chapel was built by the Portuguese in 1522 and is one of the oldest European buildings in Africa. The model was used to test the suitability of the proposed solution on heritage sites. Figure 18 shows the generated feature vertices from the tensor voting Algorithm. The algorithm took 13 hours to generate feature vertices demonstrating the effect of the large number of points on the algorithm. The output from future stages of the algorithm were not generated because of the model containing duplicated half edges. This error led to the need for adding pre-processing to remove duplicate edges and reduction on the size of the model. In addition, limited resources led to one heritage site being used to test the method hence there is need to test the algorithm on more models .

5 PERFORMANCE

Runtime			
Model	Faces	Triangles	Time/sec
Cube	386	768	10.04
fandisk	6475	12946	512.09
Cylinder	102	200	12.38
Chapel	4702858	9403100	50122.28
Torus	600	1200	67.97

The code was tested on an Intel 4 core 2.50GHZ processor with 8GB of physical memory. The table shows the models used to test the algorithm and the number of triangles. In order to analyse the performance of the algorithm 5 readings of each model's run time were averaged. In the readings we deduced that not only the size of the model increases run time however the complexity vertices of the shape has an impact the algorithm. The more complex the model the more feature vertices produced hence processes after the tensor voting algorithm take more time. This also goes in hand with noisy meshes meaning additional vertices are detected. The graph shown in figure 13 show the run time and the triangle of the meshes measured. Run time increases with triangles and the results show the algorithm to be big of O(N).

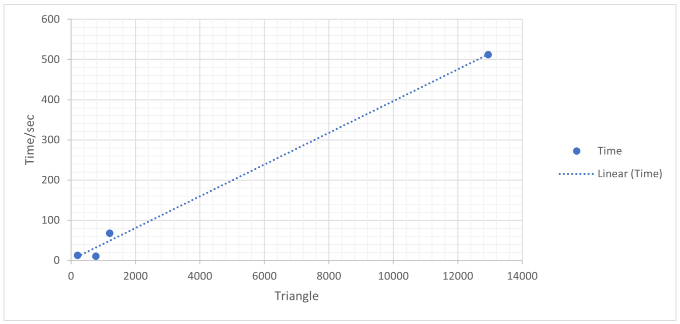

Figure 13: Graph showing the run time vs number of triangles

6 CONCLUSIONS

Important information about our past can be deduced from heritage sites. Methods to extract information from these models have been explored. In this paper we implement the tensor voting Algorithm by Wang et al [17] and test it's suitability on heritage sites. The feature lines of regular and complex shapes were produced in which the shape of the models could be perceived. The implemented solution was able to detect sharp edges and corners. In addition to detection, noisy feature vertices were filtered out. However the algorithm failed to produce required results on heritage sites because of the models containing duplicate half edges. The tensor voting algorithm shows promise to produce feature lines from large 3D heritage sites. However, further research on ways to pre-process and improve algorithm need to be explored before tackling such a big task.

7 FUTURE WORK

The running time of the algorithm can be improved by prior smoothing or re-meshing of the models which reduces the number of triangles that need to be processed. A second suggestion is to use parallel programming and to optimise the code to run on a GPU. As discussed in previously the algorithm has a pruning process which was not implemented in this paper but however for models such as heritage sites, the pruning process is required to improve the results produced.

8 ACKNOWLEDGEMENT

Thank you to my supervisor, Prof Patrick Marais , for providing guidance and feedback throughout this project. In addition I would like to acknowledge the Zamani Project for providing the models and guidance.

REFERENCES

[1] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. 2002. Interactive Geometry Remeshing. 21, 3 (2002). https://doi.org/10.1145/566654.566588

[2] Abdul Qadir Bhatti, Abdul Wahab, and Wadea Sindi. 2021. An overview of 3D laser scanning techniques and application on digitization of historical structures. *Innovative Infrastructure Solutions* 6, 4 (17 Jul 2021), 186. https://doi.org/10.1007/s41062-021-00550-9

[3] John Canny. 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8, 6 (1986), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

[4] Frédéric Cazals and Marc Pouget. 2004. *Smooth surfaces, umbilics, lines of curvatures, foliations, ridges and the medial axis: a concise overview.* Technical Report RR-5138. INRIA. https://hal.inria.fr/inria-00071445

[5] Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. 2007. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design* 39 (04 2007), 276–283. https://doi.org/10.1016/j.cad.2006.12.005

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[7] Hyun Soo Kim, Han Kyun Choi, and Kwan H Lee. 2009. Feature detection of triangular meshes based on tensor voting theory. *Computer-Aided Design* 41, 1 (2009), 47–58.

[8] Y. Lee, S. Park, Yongtae Jun, and W.C. Choi. 2004. A robust approach to edge detection of scanned point data. *International Journal of Advanced Manufacturing Technology* 23 (02 2004), 263–271. https://doi.org/10.1007/s00170-003-1695-x

[9] Gérard Medioni, Chi-Keung Tang, and Mi-Suen Lee. 2000. Tensor voting: Theory and applications. In *Proceedings of RFIA*, Vol. 2000.

[10] Rodrigo Moreno, Miguel García, Domenec Puig, Luis Pizarro, Bernhard Burgeth, and Joachim Weickert. 2011. On Improving the Efficiency of Tensor Voting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (11 2011), 2215–28. https://doi.org/10.1109/TPAMI.2011.23

[11] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. 2004. Ridge-Valley Lines on Meshes via Implicit Surface Fitting. In *ACM SIGGRAPH 2004 Papers (SIGGRAPH '04).* Association for Computing Machinery, New York, NY, USA, 609–612. https://doi.org/10.1145/1186562.1015768

[12] Fabio Remondino. 2011. Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning. *Remote Sensing* 3, 6 (2011), 1104–1138. https://doi.org/10.3390/rs3061104

[13] Heinz Rüther. 2002. An African Heritage Database: The Virtual Preservation of Africa's Past. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 34 (01 2002).

[14] Takafumi Shimizu, Hiroaki Date, Satoshi Kanai, and Takeshi Kishinami. 2005. A new bilateral mesh smoothing method by recognizing features. In *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05).* IEEE, 6–pp.

[15] Daniele Spizzichino, Claudio Margottini, Vittorio Chiessi, and Daniela Boldini. 2016. *Assessment of the stability conditions of a large-volume sandstone block in the northern sector of the Siq of Petra.* 1851–1858. https://doi.org/10.1201/b21520-231

[16] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[17] Xiaochao Wang, Junjie Cao, Xiuping Liu, Baojun Li, Xiquan Shi, and YiZhen Sun. 2012. Feature detection of triangular meshes via neighbor supporting. *Journal of Zhejiang University-SCIENCE C (Computers Electronics)* 0 (06 2012). https://doi.org/10.1631/jzus.C1100324

[18] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).

A OVERFLOW FORM OTHER SECTIONS

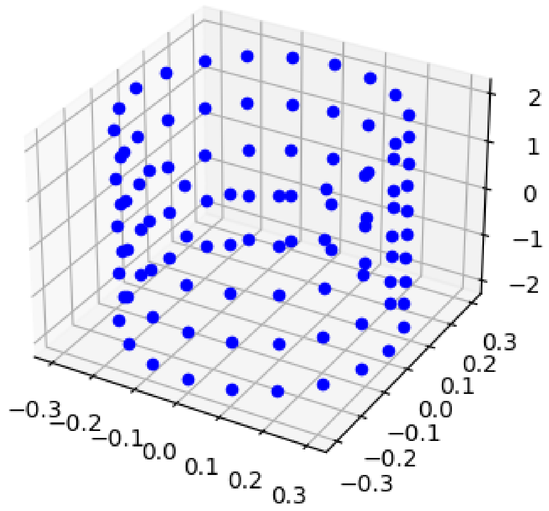

Figure 14: Generated feature vertices of the Chapel

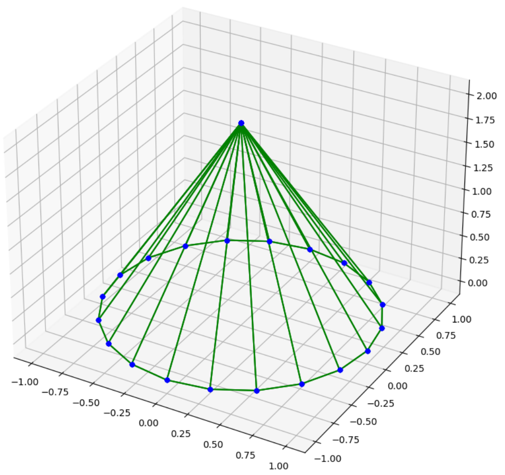

Figure 16: Generated feature lines of a cone

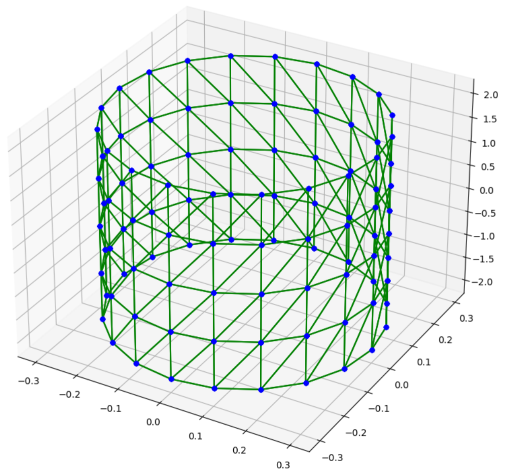

Figure 15: Generated feature lines of a cylinder

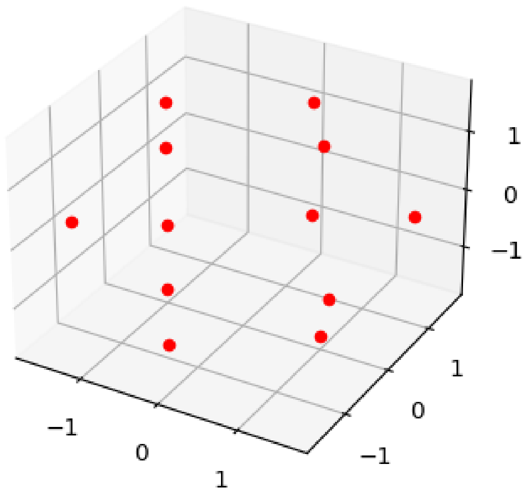

Figure 17: Generated feature vertices of a icosahendron

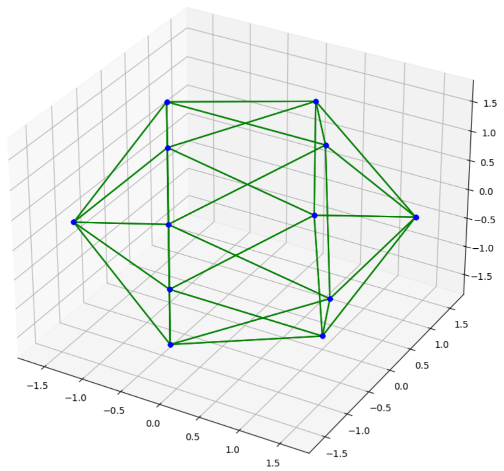

Figure 18: Generated feature lines of icosahendron