

**College of Engineering**

**Department of Mechanical and Aerospace Engineering**

A close-up of a coin

Description automatically generated with low confidence

**MAE-586**

**M.S. Aerospace Engineering Project Work**

**Submitted by: Anna Davis**

**Faculty Advisor: Dr. Landon Grace**

**Research Supervisor: Andrew Gorbaty**

**Email:** [**andrewg@factor.io**](mailto:andrewg@factor.io)

**Date Report Submitted: 12/13/2022**

1. **Background**

In today’s supply chain, most parts that aerospace engineers design are manufactured at one facility (usually a Contract Manufacturer) and shipped to another facility for assembly. Every part is protectively packaged and put in a container, then transported via methods such as container ships, trucks, cargo planes, etc. The status quo of packaging and shipping today is not data-driven and incredibly conservative which leads to an unnecessary amount of waste (including many single-use plastics). This excessive protective material (ESD bags, bubble wrap, etc.) could be avoidable through optimizations aimed at minimizing materials needed. This overuse of protective materials also increases the number and size of boxes/pallets that need to ship, driving up the cost and carbon footprint of shipping, while failing to guarantee the parts are not damaged. The main goal of this project will be to minimize waste created/carbon footprint of the overall shipping process through determining the ideal amount of protective material to fit specific part configurations and quantities, therefore helping customers avoid being overcharged for shipping costs and preventing damages to parts. Damage could be caused by things such as: weight of other parts in the box, impact during shipping (dropped, bumped, etc.), and general mishandling of the packages. This project is in collaboration with Factor Technology, where the author/researcher of this project currently works in the position of “supply chain and operations engineering intern”.

1. **Initial Objective and Research**

The initial goal of this project was to create a program in Python that would import a 3D model of a part (a metal CNC machined part with varying size and shape) and analyze it to optimize the thickness of protective material (bubble wrap) needed to ensure that the part is not damaged during shipping and transportation. This was later changed to creating a program in MATLAB instead, due to time constraints and preference of the researcher. The program would also select the most appropriate size of the bubbles in the bubble wrap for the part. Upon completing some research on bubble wrap and bubble diameter, the conclusion was reached that selecting the appropriate bubble size would not be realistic since bubble wrap comes in standard sizes that equate to a certain bubble thickness and diameter. As seen in Figure 1, standard bubble wrap thicknesses correspond to a certain size diameter already [1]. For example, hypothetically if the optimal thickness was 3/16-inch and the optimal bubble size was 1 inch in diameter, this result would not realistically be able to be implemented since the standard sizes are 3/16-inch thickness with a bubble diameter size of 3/8-inch and 5/16-inch thickness with a bubble diameter of 1 inch in thickness. Requesting that the dimensions of bubble wrap sizes that are standard across most packaging companies would be unrealistic and unattainable, while also not decreasing the amount of packaging material used by a significant amount. Therefore, for the purpose of this project, the bubble diameter size was not examined, and more focus was placed on the bubble wrap thickness in terms of number of layers needed. After further investigations on bubble wrap, it was determined that using a thickness of 3/16-inch as the layer thickness for this study would be best suitable since this thickness is the standard size and most commonly used [1]. Finding the number of layers of bubble wrap at this thickness-per-layer would hopefully produce results that could be used for the majority of the parts uploaded into the program, which was the driving factor behind this decision to proceed with using only a layer thickness of 3/16 inch.

Text

Description automatically generated

**Figure 1. Bubble wrap size chart [1]**

One of the more likely scenarios a part may experience during shipping is being dropped by the person transporting it, therefore in this study the impact force needing to be neutralized by the bubble wrap is the impact acceleration due to the force of gravity acting on the part. The ability of bubble wrap to negate this impact acceleration was tested and studied in the paper, “Rice Hulls as a Cushioning Material” [2]. The researchers of this paper were comparing how well three different materials – bubble wrap, an anti-vibration rubber pad, and loose rice hulls – could reduce the impact acceleration of a box during a drop test [2]. The following relationship was found for bubble wrap, seen in Figure 2.

Table

Description automatically generated

**Table 1. Percent change trendline for bubble wrap [2]**

This chart shows the average baseline impact acceleration of 77.53g, and with each layer of bubble wrap the recorded average impact acceleration steadily decreases. This decrease was converted to a negative percentage of change, which was then used to create a predictive trendline for the percent change, with “x” being the starting thickness in inches. Extrapolating from this relationship, the percent change of impact acceleration trendline played a crucial role in the methods for optimization of this paper’s research.

1. **Methods for Optimization and Implementation**

For the optimization to begin, user-defined inputs were required. A **3D-model** and material properties are needed, and the **estimated height** that the part might be dropped from can also be entered as an input. The material properties necessary are **density** (in g/cc) and **Brinell hardness** (in kgf/mm2), which can be found online from MatWeb (<https://www.matweb.com/>) for most engineering materials. Density is often given in g/cc and the standard unit of Brinell hardness is defined as kgf/mm2. Dimensions for the 3D model are assumed to be in units of millimeters; this is because most parts that are submitted to the Factor platform have the units specified as millimeters in the technical drawings.

After obtaining all the necessary information, the geometry of the 3D model is imported, and the finite element mesh is created. From this mesh the part volume and surface area can be found. Then the volume and density are used to fine the mass. From here the mass, gravitational constant, and height dropped are used to calculate the impact acceleration force. This is compared to the Brinell hardness, which is a measure of the ability of a material to resist permanent indentation or deformation, used mainly for metals such as steel, aluminum, copper, and iron (all common metals used in the parts that users submit to Factor). A while loop is used with the condition that if the impact acceleration force is greater than the Brinell hardness, the loop continues, and another layer of bubble wrap is added. The equation for the percent change of impact acceleration trendline is used in this loop as well, with the percent change negatively increasing every time a layer is added, thus decreasing the overall impact acceleration until it is less than the Brinell hardness. A starting thickness layer of 3/16-inch (0.1875 inches) is initialized, then total thickness and number of layers increases with each iteration. For more details about the code used, see Part A of the Appendix.

3D part model examples were obtained from NASA through a free online download [3]. Through NASA’s “3D resources” website, parts were available for download with the purpose of users being able to 3D print their own model versions of NASA vehicles. The vehicle of choice was the Curiosity Rover; the only requirement for the imported 3D part is that it must be a “.STL” type file, so the parts were not restricted by size or intended use. Any .STL file can be used, and the parts from the model Curiosity Rover were chosen based on personal interest. Three different parts from Curiosity Rover were tested in this research in order to compare results: “legs-right”, “mahli-apxs”, and “upper-arm”. Each part was tested with two different materials (aluminum and stainless steel). Additionally, each combination of part and material was tested at the following drop heights: 1, 5, 10, and 25 feet. The height of 1 foot was chosen to represent a light drop, 5 feet was to represent the height a person might accidentally drop the part from while carrying it, 10 feet was to represent an intentional sabotage of someone dropping the part over a low balcony perhaps, and 25 feet was selected to represent a more unlikely but extreme case. Photos of the parts are displayed in Figure 3.

A picture containing floor, black, white

Description automatically generated A picture containing floor, tiled, tile

Description automatically generated

1. **B)**

A picture containing floor

Description automatically generated

**C)**

**Figure 2. 3D models of parts: A) “legs-right, B) “mahli-apxs”, and C) “upper-arm” [3]**

1. **Results**

The materials Aluminum 6061-T6 and Stainless Steel 301 were used for each 3D part at the heights of 1, 5, 10, and 25 feet to determine how much bubble wrap (thickness and layers) would be needed to ensure the parts wouldn’t suffer permanent damage. Aluminum 6061-T6 was selected because of its wide variety of applications in aircraft fittings, camera lens mounts, marine fittings and hardware, electrical connectors, hinge pins, pistons, valves and much more [4]. The density of Aluminum 6061-T6 is 2.7 g/cc, and the Brinell hardness is 95 kgf/mm2. Stainless Steel 301 is a high-grade steel that was also chosen for its diverse range of uses, including automobile molding and trim, wheel covers, conveyor belts, kitchen equipment, springs, truck/trailer bodies, railway/subway cars, and many other mechanical functions. The density of Stainless Steel 301 is 8.03 g/cc, and the Brinell hardness is 217 kgf/mm2 [5].

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1 foot** | | **5 feet** | | **10 feet** | | **25 feet** | |
|  | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** |
| **Aluminum** | 7 | 1.3125 | 9 | 1.6875 | 10 | 1.8750 | 11 | 2.0625 |
| **Stainless Steel** | 7 | 1.3125 | 10 | 1.8750 | 10 | 1.8750 | 11 | 2.0625 |

**Table 2.** Results for part “legs-right”

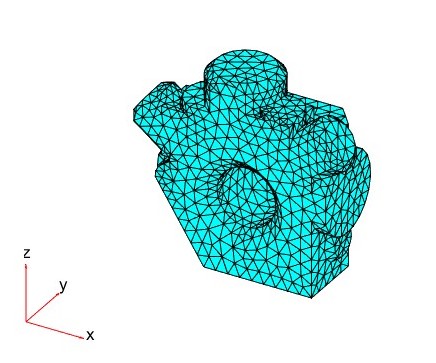
**Shape, arrow

Description automatically generated with medium confidence**

**Figure 3.** FE mesh of “legs-right”

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1 foot** | | **5 feet** | | **10 feet** | | **25 feet** | |
|  | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** |
| **Aluminum** | 7 | 1.3125 | 9 | 1.6875 | 10 | 1.875 | 11 | 2.0625 |
| **Stainless Steel** | 8 | 1.5000 | 10 | 1.8750 | 10 | 1.8750 | 11 | 2.0625 |

**Table 3.** Results for part “mahli-apxs”

****

**Figure 4.** FE mesh of “mahli-apxs”

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1 foot** | | **5 feet** | | **10 feet** | | **25 feet** | |
|  | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** | **Number of layers** | **Total Thickness (inches)** |
| **Aluminum** | 7 | 1.3125 | 10 | 1.8750 | 10 | 1.8750 | 11 | 2.0625 |
| **Stainless Steel** | 8 | 1.5000 | 10 | 1.8750 | 10 | 1.8750 | 11 | 2.0625 |

**Table 4.** Results for part “upper-arm”

**A picture containing metalware, key

Description automatically generated**

**Figure 5.** FE mesh of “upper-arm”

****

**Figure 6.** Picture of the completed Curiosity Rover model

|  |  |  |  |
| --- | --- | --- | --- |
|  | **“legs-right”** | **“mahli-apxs”** | **“upper-arm”** |
| **Average elapsed time (in seconds)** | 4.2157 | 4.0899 | 2.8117 |

**Table 5**. Average elapsed time for code to finish running per part

1. **Discussion and Limitations**

After completing all the runs of the program, a common trend across all parts and materials was that more bubble wrap layers were needed as the height between the theoretical dropping point and the ground increased. This result aligns with general expectations; since the height and impact force is increasing, a greater amount of bubble wrap is necessary to keep the part undamaged. It’s interesting to note that when the material is Stainless Steel 301, more bubble wrap is needed sooner, across all of the parts. When “legs-right” is made of stainless steel, the number of layers required goes up to 10 when height is increased to 5 feet, whereas Aluminum 6061-T6 only goes up to 9 layers at 5 feet. When the other two parts (“mahli-apxs” and “upper-arm”) are made of Stainless Steel 301, they start out needing 8 layers of bubble wrap at a height of 1 foot, while when the parts are made of Aluminum 6061-T6 they only need 7 layers. This is probably due to the density of Stainless Steel being almost 3 times as much as Aluminum 6061-T6, while its Brinell hardness is only about 2.25 times as much as Aluminum 6061-T6. All parts, regardless of material, need 10 layers of bubble wrap at 10 feet and 11 layers at 25 feet. Number of layers multiplied by 3/16-inch (0.1875 inch) gives the total thickness of the bubble wrap needed. At lower heights there is more likely to be some discrepancies between the number of layers required based on material properties, but after a certain height the numbers converge because the height at which the part is dropped has a more dominant influence over the impact force than the material properties.

The method used in the project and results are not without limitations, with one of the limitations being the time it takes for MATLAB to import 3D geometry and create a mesh of more complex parts. For example, the first part out of the Curiosity Rover files that was tested was the main “body” 3D part, but code took an unreasonable amount of time to run (over 10 minutes) and the operation was terminated in MATLAB. Table 5 shows the average time elapsed for each part, and even though all the times were under 5 seconds for these parts, it can be observed that more elaborate part geometries increase the overall time it takes to run the program. From visual inspection, it is clear that “upper-arm” is the least intricate out of the three parts, and time elapsed when running the code using “upper-arm” is over one second less on average than the time for the other two parts. Doing a deeper investigation with more FEA analysis in an actual FEA software package such as ANSYS or Abaqus on this topic would be beneficial for a faster/more efficient run time without having to worry about how complex the part might be. Further FEA analysis could also provide confirmation/validity of the results found in this project. Due to time and expense constraints, it was also not possible to get these parts CNC machined in Aluminum 6061-T6 and Stainless Steel 301 to conduct physical drop tests with bubble wrap. However, testing the parts and bubble wrap layers this way would certainly confirm or deny the validity of the MATLAB program and results.

1. **Future Research**

In addition to methods mentioned in the previous section to address this project’s limitations, further research on the behavior of bubble wrap would be beneficial. Even though bubble wrap provides satisfactory protection in most instances, the bubbles will pop at certain point if the impact force is overwhelming, but currently there is little information about how to predict when this will happen. If a part is wrapped in 10+ layers of bubble wrap but most of the bubbles pop, then it cannot protect against additional impacts and more layers will be needed, increasing material/resource usage. Also, there is no monitoring process or way to tell if or when the bubbles pop during shipping and going through every box to check is a nonsensical idea.

Future potential approaches for shipping optimization could be explored for more accurate and customizable results. Using the shape diameter function (SDF) is a promising method to obtain more information about the thickness of a part. In this paper, the parts that were used were not hollow, but SDF is useful for 3D geometries involving varying thicknesses of material in certain areas of a part. The SDF takes the 3D geometry of a closed mesh (must ensure that mesh is completely closed) and uses mesh vertices along with generated surface normals for each mesh triangle to find the thickness of an imported shape/model. The SDF is defined as “the weighted average of the length of rays traced from the given point inside the inside the cone centered at its inward-normal” [6]. Using the SDF would provide more details about the inside of the part, and how much impact/force it could withstand at certain points based on thickness. The area of minimum thickness found through the SDF is another way the least amount of bubble wrap needed could be calculated as well and would be more sufficient for larger/hollow parts than the approach implemented in this report. An example of an SDF can be found in Part B of the Appendix.

1. **Conclusion**

If the program for optimization of bubble wrap needed to ensure a part would not be damaged under impact forces during shipping was further validated through physical testing or other algorithms, there is great potential for this research to have a substantial impact on current shipping methods. Different materials besides bubble wrap could be optimized as well (such as foam, packing peanuts, and other single-use plastics), which would lead to a decrease in money spent on shipping costs (benefiting the shipper and the receiver). This would also help with the reduction of plastic/material waste being used in shipments and ending up in landfills, benefiting the environment over time (though the fastest and most thorough solution would be an increase in bio-degradable packaging materials). It is a rather frustrating experience to receive a small object in the mail that has been shipped in an overly large box with an excessive amount of plastic/packing material, and this program would help the shippers avoid this kind of situation.

Continued research using FEA and SDF could help create more accurate and efficient optimizations, as well as help engineers be more conscious of 3D part topology and thickness during the design process. Using MATLAB as a proof of concept for mesh creation, modeling, and bubble wrap layer prediction is of use to Factor Technology as this code could be re-worked and extended to provide more information on 3D geometries and give users the ability to procure a rough price estimation of the part(s) they want to send in for manufacturing. This project was able to determine the minimum layers of bubble wrap needed to maintain part integrity through an exploration of metal material properties, part volume, and the ability of bubble wrap to negate impact forces. The research also laid some groundwork for newer advances regarding 3D modeling/graphic algorithms and predictive analysis within Factor Technology.

1. **Acknowledgements**

This project is in collaboration with the author’s current place of work (Factor Technology) in the position of supply chain and operations engineering intern. The name and email address of the supervisor from Factor has been included on the title page. To ensure class credit is received for the project, it is not being directly funded by the company.

1. **References**

[1]*Bubble products*. Premier Protective Packaging. (n.d.). Retrieved December 2022, <https://www.foambubble.com/bubble-products>

[2] Malasri, S., Stevens, R., Othmani, A., *et al*. (2014). “Rice Hulls as a Cushioning Material”. *International Journal of Advanced Packing Technology, 2*(1), 112-118. 10.23953/cloud.ijapt.13.

[3] NASA. (n.d.). *Curiosity Rover*. NASA. Retrieved October 2022, <https://nasa3d.arc.nasa.gov/detail/mars-rover-curiosity>

[4] ASM Aerospace Specification Metals, Inc. (n.d.). *Aluminum 6061-T6*. ASM material data sheet. Retrieved November 2022, <https://asm.matweb.com/search/SpecificMaterial.asp?bassnum=ma6061t6>

[5] MatWeb*. 301 stainless steel*. MatWeb material data shee. (n.d.). Retrieved November 2022, <https://www.matweb.com/search/DataSheet.aspx?MatGUID=0cf4755fe3094810963eaa74fe812895>

[6] Madaras, M., Durikovic, R., *et al.* (2016). “GPU-based Approaches for Shape Diameter Function Computation and Its Applications Focused on Skeleton Extraction”. *Computers & Graphics, 59*, 151-159. <https://doi.org/10.1016/j.cag.2016.06.006>

[7] Gardan, N., and Schneider, A. (2014). “Topological optimization of internal patterns and support in additive manufacturing”. *Journal of Manufacturing Systems, 37*(1), 417-425. <https://doi.org/10.1016/j.jmsy.2014.07.003>

[8] Huang, P., Ding, Y., *et al.* (2020). “An improved contact detection algorithm for bonded particles based on multi-level grid and bounding box in DEM simulation”. *Powder Technology, 374*, 577-596. <https://doi.org/10.1016/j.powtec.2020.07.022>

[9] Zuo, Z.H., and Xie, Y.M. (2015). “A simple and compact Python code for complex 3D topology optimization”. *Advances in Engineering Software, 85,* 1-11. <https://doi.org/10.1016/j.advengsoft.2015.02.006>

1. **Appendix** 
   1. **Main Program (MATLAB)**

%% MAE 586 Project

clear all

clc

tic

%% user inputs and material properties, subject to change

dimension\_units = 'mm';

hardness = 217; % brinell hardess in kgf/mm^2

density = 8.03; % in grams/cm^3 (grams/cc)

density\_mm = density/1000; % in grams/mm^3

g = 9810; % acceleration due to gravity in mm/s^2

height = 50; % in feet

height\_mm = height\*304.8; % in mm

%% creating FE mesh

smodel = createpde('structural','static-solid');

importGeometry(smodel,'../3d\_files/upper-arm.STL');

figure(1)

pdegplot(smodel, 'FaceLabels', 'on', 'FaceAlpha', 0.5); % add timer

figure(2)

msh = generateMesh(smodel);

pdeplot3D(smodel)

%% Volume, surface area, and mass calculations

V = volume(msh); % in mm^3

[V,VE] = volume(msh);

[X,Y,Z] = meshgrid(msh);

x = X.Nodes(1,:)';

y = Y.Nodes(2,:)';

z = Z.Nodes(3,:)';

shp = alphaShape(x,y,z);

A = surfaceArea(shp);

mass = V\*density\_mm; % units in grams

%% initial constants

thickness = 0.1875; % thickness of 1 layer of bubble wrap in inches

acceleration\_force = mass\*g\*height\_mm; % in (grams\*mm^2)/s^2)

new\_acceleration = acceleration\_force;

hardness\_adjusted = (hardness\*9.806650)\*A; % kgf/mm^2 to N/mm^2 = (grams\*mm/s^2)/mm^2)\*mm^2 = (grams\*mm^2/s^2)

%% finding thickness and number of layers needed for the acceleration force to become less than the hardness of the material

while new\_acceleration >= hardness\_adjusted

percent\_change = (-31.356\*thickness - 10.156)/100;

new\_acceleration = new\_acceleration+(new\_acceleration\*percent\_change);

thickness = thickness + 0.1875; % in inches

end

num\_layers = ceil(thickness/0.1875)

final\_thickness = thickness

toc

* 1. **Shape Diameter Function (Python)**

|  |
| --- |
| from asyncore import write |
|  | import sys |
|  |  |
|  | from pyparsing import col |
|  | from converter.mesh\_converter import mesh\_converter, mesh\_creator |
|  | import igl |
|  | import scipy as sp |
|  | import numpy as np |
|  |  |
|  | class shape\_diameter\_compute(): |
|  |  |
|  | def \_\_init\_\_(self, vertices : np.array, faces : np.array): |
|  | self.vertices = vertices |
|  | self.faces = faces |
|  | self.normals = None |
|  | self.bounding\_box\_diagonal = None |
|  | self.shape\_diameter = None |
|  |  |
|  | def generate\_normals(self, print\_normals=False): |
|  | self.normals = igl.per\_vertex\_normals(self.vertices, self.faces) |
|  | if print\_normals: |
|  | print(len(self.normals)) |
|  | assert(self.normals is not None) |
|  |  |
|  | def set\_bounding\_box\_diagonal(self, print\_bounding\_box=False): |
|  | if self.vertices.any(): |
|  | self.bounding\_box\_diagonal = igl.bounding\_box\_diagonal(self.vertices) |
|  | if print\_bounding\_box: |
|  | print(self.bounding\_box\_diagonal) |
|  | assert(self.bounding\_box\_diagonal is not None) |
|  |  |
|  | def generate\_shape\_diameter(self, num\_samples : int, normalize=True, print\_shape\_diameter=False): |
|  | if self.vertices.any() and self.faces.any() and self.normals.any(): |
|  | self.shape\_diameter = igl.shape\_diameter\_function(self.vertices, self.faces, self.vertices, self.normals, num\_samples) |
|  | if normalize and self.bounding\_box\_diagonal: |
|  | self.shape\_diameter /= self.bounding\_box\_diagonal |
|  | if print\_shape\_diameter: |
|  | print(len(self.shape\_diameter)) |
|  | elif print\_shape\_diameter: |
|  | print('please compute vertices, faces, and normals before executing this function') |
|  | assert(self.shape\_diameter is not None) |
|  |  |
|  | # NOTE: creates a copy every time, when created the caller is responsible for releasing its memory |
|  | def transform\_shape\_diameter\_into\_colors(self): |
|  | out\_shape\_diameter = None |
|  | if (self.faces.any() and self.shape\_diameter.any()): |
|  | # ugly for loop here, sorry |
|  | out\_shape\_diameter = np.ones((len(self.faces), 3)) |
|  | for row in range(len(out\_shape\_diameter)): |
|  | for column in range(len(out\_shape\_diameter[row])): |
|  | out\_shape\_diameter[row][column] = self.shape\_diameter[self.faces[row][column]] |
|  | assert(out\_shape\_diameter is not None) |
|  | return out\_shape\_diameter |
|  |  |
|  |  |
|  | def get\_shape\_diameter(self): |
|  | return self.shape\_diameter |
|  |  |
|  | def write\_color\_mesh\_to\_disk( |
|  | mesh\_name : str, |
|  | mconv : mesh\_converter, |
|  | vertices : np.array, |
|  | faces : np.array, |
|  | sdc : shape\_diameter\_compute |
|  | ): |
|  | mcreator = mesh\_creator() |
|  | mconv.set\_mesh(mcreator.create\_mesh(vertices, faces, sdc.transform\_shape\_diameter\_into\_colors())) |
|  | mconv.write\_to\_disk(mesh\_name) |
|  |  |
|  | MESH\_FILE\_INPUT\_PATH = 1 |
|  | NUM\_SAMPLES = 2 |
|  | DEFAULT\_NUM\_SAMPLES = 100 |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | mconv = mesh\_converter(sys.argv[MESH\_FILE\_INPUT\_PATH]) |
|  | mconv.ingest\_3d\_file() |
|  | mesh = mconv.get\_triangles() |
|  | if mesh: |
|  | vertices, faces = mesh |
|  | sdc = shape\_diameter\_compute(vertices, faces['triangle']) |
|  | num\_samples = int(sys.argv[NUM\_SAMPLES]) if len(sys.argv) == 3 else DEFAULT\_NUM\_SAMPLES |
|  | print('generating normals') |
|  | sdc.generate\_normals() |
|  | print('set bounding box') |
|  | sdc.set\_bounding\_box\_diagonal() |
|  | print('calling shape diameter function') |
|  | sdc.generate\_shape\_diameter(num\_samples) |
|  | #write\_color\_mesh\_to\_disk("blahMesh.vtk", mconv, vertices, faces, sdc) |