Thesis

Virtual environments

Mar 16, 2019

# Configuring Virtual environment

## Configuring virtual environment to allow easy switching of CNN distributions.

Going to follow the steps at this link <https://towardsdatascience.com/setup-an-environment-for-machine-learning-and-deep-learning-with-anaconda-in-windows-5d7134a3db10>. It is one of many. However, it is a good start.

1. Download and install Anaconda, python v3.7.
   1. Installing package.
   2. The GUI version of the installer had issues completing.
   3. Trying the command line installer.
   4. Successfully installed Anaconda.
   5. Verify the installation: Anaconda Launchpad runs. \*\* success.
2. Update Anaconda
   1. **“conda update conda”**
   2. **“conda update –all”**
3. Install CUDA Toolkit and cuDNN
   1. Download CUDA 10.1. It is the version currently available.
   2. Cuda installed in /Developer/NVIDIA/CUDA-10.1.
   3. Samples are in the <Installed Location>/samples.
   4. Download cuDNN 7.5.0. Intended version for CUDA 10.1.
   5. **Unzipped the contents of the package into /Developer/NVIDIA/CUDA-10.1/cuda**
   6. Need to add cuDNN to path: **/Developer/NVIDIA/CUDA-10.1/cuda/bin**
4. **Create Anaconda environment – Creates a new environment so that it does not impact the original environment. (This is the step we will need to repeat)**
   1. **“conda create -n tensorflow pip python=3.6”.** Installation was successful.
   2. To activate: “**conda activate tensorflow**”. To deactivate: “**conda deactivate**”.
5. Install Deep Learning
   1. Run: pip install tensorflow-gpu \*\* successful.
   2. Testing with Python “import tensorflow as tf” failed.
   3. **Run: pip install tensorflow**
   4. Test with Python “import tensorflow as tf” succeeded.
   5. **Install Keras: pip install keras**
   6. Manually testing MNIST: <https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py>. Runs successfully.

The process of installing Anaconda was simple. Creating the virtual environment and then installing the deep learning tools was easy.

Nvidia Docker containers

Mar 17, 2019

# Configuring Nvidia Containers

## The Nvidia containers may make deployment easier. Investigate how to deploy.

1. Following instructions here: <https://developer.nvidia.com/nvidia-container-runtime>
2. \*\* Not possible on Mac. Can be run on Ubuntu.

Boundary First Flattening

March 29, 2019

# How do we create our own mesh?

If we can create our own mesh, we can then create set test data. The information is potentially here: https://github.com/GeometryCollective/boundary-first-flattening/issues/21

Looking at the .obj file through Microsoft Excel, it appears to be ASCII text. There are alternating lines of “v” and “vt”.

The “v” represent the 3D coordinates while the “vt” represent the flattened UV view.

According to the link, the faces are constructed using the vertices.

\*\* The face definitions in the OBJ file are not contiguous.

Additional information about the OBJ file: <https://all3dp.com/1/obj-file-format-3d-printing-cad/>

Python has libraries for handling/creating meshes: <https://pymesh.readthedocs.io/en/latest/>. Investigate running pyMesh in docker. When running docker instance, need to map the OS file system: <https://stackoverflow.com/questions/47464614/how-do-i-access-the-local-filesystem-from-a-python-docker-image>. Command to run docker: docker run -v /Users:/Users -it pymesh/pymesh

To execute my file, run this command (Be sure to add comment to the code): exec(open("/Users/hengsun/Documents/Thesis/generate.py").read())

There is an application called Blender. It is an animation package that allows creation of mesh. \*\* Too complicated right now.

The format of the .obj file is indeed v <x> <y> <z>.

Tried to create a simple pyramid. However, kept receiving error that there are non-manifold vertices.

pymesh has sample generators. One of the shapes it can generate is the icosphere: mesh=pymesh.generate\_icosphere(1, (0,0,0), 2).

Mar 31, 2019

Finally able to generate a mesh (half sphere) that BFF will accept.

We can produce target shapes as well.

Now what?

The sphere has rather limited shape options. What other meshes can we generate?

Building BFF

April 7, 2019

# BFF command UI does not allow automatic definition of the target UV geometry. Need to build custom version or integrate with custom code.

* Download github repository: <https://github.com/GeometryCollective/boundary-first-flattening>

**Installing SuiteSparse**

* Try to build by following instructions
  + \*\* Need to build SuiteSparse first.
* <https://github.com/jlblancoc/suitesparse-metis-for-windows>

1. Install LAPACK and BLAS: <https://pheiter.wordpress.com/2012/09/04/howto-installing-lapack-and-blas-on-mac-os/>
   1. Compile BLAS is easy. Install is easy.
   2. Compile LAPACK requires that you make a copy of the make.inc. Compile is easy. Install is easy.
   3. Next step in build \*\* stuck. \*\*
   4. Alternative is to use home brew to install: brew install suitesparse. This installed, however, still had a problem with compiling.
   5. In file /Users/hengsun/Documents/Thesis/suitesparse-metis-for-windows-1.4.0/metis/GKlib/gk\_arch.h, had to comment out the rint macro. \*\* Solution.
2. Build BFF.
   1. git clone <https://github.com/GeometryCollective/boundary-first-flattening.git> \*\* successful.
   2. cd boundary-first-flattening && git submodule update --init –recursive \*\* successful.
   3. mkdir build && cd build && cmake .. \*\* successful.
   4. make -j 4 \*\* successful, but with many warnings. Files are placed in **/Users/hengsun/Documents/Thesis/boundary-first-flattening/build.**
   5. If there are compile errors, look at: <https://stackoverflow.com/questions/52509602/cant-compile-c-program-on-a-mac-after-upgrade-to-mojave>. Basically need to run “***xcode-select –-install”.***

**Next: Modifying Command-line**

1. Try to run the command line and see what happens. It seems to produce an output file. However, the viewer doesn’t seem to remember where the cones were placed or the shape of the output. Instead, the viewer loads the original mesh file and restarts. The UV coordinates are different, depending on the number of cones defined. How do we visualize it?
   1. Option 1: Extract just the UV from the files. Simple Python script could do it. \*\* Success!
2. Command to run BFF Command line: /Uers/hengsun/Documents/Thesis/boundary-first-flattening/build/bff-command-line newIntersect2.obj out.obj
3. Trying to now configure the DenseMatrix to populate the boundaries.
4. **Now we can modify the command-line and try to access the additional methods, specifically BFF:flatten**.
   1. Was able to get it to work somewhat. Added new variable “—angle=<float>”.
   2. Interesting values: --angle=0.1, --angle=0.2. The angles seem to be fractions of Pi. Jagged @0.4, Jagged@0.5
   3. Interesting add – we can change how the angles are applied. They need not all be the same size.
   4. If the angles are too small, we start to produce triangular shapes.
   5. Things we can control – the size of the mesh and the angles that the edge of the mesh will have, within a limit.
5. Pymesh has a method that called collapse\_short\_edges. This method has the ability to remove edges that are below a certain threshold. For our implementation, we are removing edges that are below less than 0.5\*average edge length.
6. Can now also produce a rectangular notch in the flattened output. However, due to the way the flattening algorithm works, we cannot guarantee the length of any edge. As a result, we cannot get a perfectly symmetric notch.
7. /Thesis/boundary-first-flattening/build/extract.py is the what we are using to reform the flattened imaged. This command takes trimmedout.obj as input. This should be a icosphere or some other 3-d object. It will output trimmedout\_flat.obj. This will be a flattened version of the 3-d object.
8. **Need to figure out why the extract flattens the image while the BFF processed image does not appear flat.**

Chain Codes

Sep 8, 2019

# Implementing Chain codes

## Chain codes can be used to represent the outer perimeter of a shape, IF the coordinates form a closed loop.

The simplest recognizable enclosed shape using chain codes would be the triangle. The code would consist of three clockwise 120 degree turns or three counter-clockwise 120 degree turns.

How do we represent chain codes for the project?

Image Contour etching

Nov 11, 2019

# Summary

Given an image with one or more regions, we would like to produce a contour etched image of each region.

To produce the contour etching, we will use techniques introduced in Boundary First Flattening (Rohan Sawhney and Keenan Crane).

We start with an geodesic dome. We place a single point on the surface of each facet located on the dome. Each point can be connected to neighbouring facet to form contour lines.

The contour lines will be influenced by the shape of the original region (TBD).

We then utilized BFF to map the dome in 3D onto an arbitrary 2D shape. The 2D shape will conform to the regions in the original image. The transformation will also warp the contour lines on the dome, resulting in contour etched shape that matches the region from the original image.

# TODO:

1. Figure how to generate an geodesic dome with n vertices along the boundary.
   1. Create raster in Python \*\* done.
   2. Display the Raster \*\* done.
   3. Create circle in raster \*\* done.
   4. Populate the perimeter vertices
      1. Draw radial lines on the circle. \*\* done.
      2. Find all the points that are in the radius. Already calculated when drawing radial lines. \*\* done.
2. Robert Bridson – algorithm for dart throwing.
   1. Figure out how the algorithm works. Kind of understand the algorithm. Have questions about the actual implementation. There is an implementation on git: <https://gist.github.com/rougier/c168856c01159833ad06381388821fe6>
   2. Key concept - Poisson disc sampling: <https://www.jasondavies.com/poisson-disc/>
   3. Poisson disc tutorial: <https://www.youtube.com/watch?v=7WcmyxyFO7o>
   4. **Poisson disc sampling using Robert Bridson’s algorithm:** [**https://github.com/emulbreh/bridson**](https://github.com/emulbreh/bridson)
   6. Dart throw the additional points – Reject any that do not fulfill requirements. Using Bridson’s algorithm implementation in Python. \*\* done.
   7. Find the slope on a circle: <https://math.stackexchange.com/questions/232571/how-do-i-get-the-slope-on-a-circle>. Use the slope to modify the ‘r’ value.
   8. Find the slope of any algorithm: <https://www.symbolab.com/solver/slope-calculator/slope%20x%5E%7B2%7D%2By%5E%7B2%7D%3D1>
   9. Use Delaunay triangulation to create geodesic dome: <https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.spatial.Delaunay.html> \*\* done.
   10. Understanding Numpy array slicing: <https://machinelearningmastery.com/index-slice-reshape-numpy-arrays-machine-learning-python/>
   11. Create 2D image file. \*\* done.
   12. Will BFF work with 2D image file? \*\* **Yes, BFF can load and process the flat circle mesh!!! Unfortunately BFF operates on the 3D coordinates, not the 2D coordinates. Make the mesh output as 3D, but with z=0. \*\* Works.**
       1. /Users/hengsun/Documents/Thesis/boundary-first-flattening/bff-command-line test1.obj test1\_out.obj --angle=0.3
       2. /Users/hengsun/Documents/Thesis/boundary-first-flattening/extract.py test1\_out.obj test1\_out\_flatten.obj

Nov 18, 2019

TODO:

1. Build problems. Create fake link: **sudo ln -s /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.15.sdk /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.14.sdk**
   1. **\*\* Fixed it.**
2. **Change the BFF Command line to accept an input file. Need to modify CommandLine.cpp->flatten.**
   1. Hard coded the angle file to be “chaincode.txt”.
   2. Each line of the chaincode.txt represents one angle.
   3. \*\* Success. Can create a letter H.
3. Was able to create 3D dome. BFF accepts the dome without error. \*\* done.

Nov 22, 2019

# Barycentric coordinates

# Automatic Chaincode blob generation

# Barycentric coordinates

TODO:

1. How to determine which face maps from pre-BFF to post-BFF.
2. Barycentric Coordinates: <https://codereview.stackexchange.com/questions/41024/faster-computation-of-barycentric-coordinates-for-many-points>
3. More Barycentric: <https://github.com/bmechtley/barycentric>
4. How to determine which face maps from pre-BFF to post-BFF.
5. Produce arbitrary shaped blob.
   1. How do we feed this information into BFF?
   2. Given a blob, generate chain code for perimeter. Count the number of vertices.
   3. Freeman chain code for letters: <https://www.kaggle.com/mburger/freeman-chain-code-second-attempt>
   4. Alpha Shapes: <https://plot.ly/python/alpha-shapes/>
   5. Concave hull: <https://towardsdatascience.com/the-concave-hull-c649795c0f0f>
   6. Minimal Convex hull: <https://kukuruku.co/post/building-a-minimal-convex-hull/>
   7. Generate the geodesic with appropriate number of vertices.
   8. Implement BFF to make use of Chain code.
   9. Produce 3D image file \*\* Defer.

Dec 9, 2019

# Plotting meshes

# Mapping original triangles to newly created mesh triangle

TODO:

1. Create On-mouse over mesh of original. \*\* Working.
2. Create subplot. First subplot for original mesh. \*\* Working.
3. On-mouse over and highlighting the triangle. \*\* Working.
4. Git add .
5. Git commit -m “message”
6. **Git push**
7. Second subplot for generated mesh.
8. Plot both original mesh and new mesh together.

Dec 15, 2019

# Mapping the original mesh to flattened mesh. Draw straight lines in the original mesh and map to flattened mesh.

TODO:

1. Parse intermediate obj file.
   1. Create coordinates for original mesh. \*\* done.
   2. Create coordinates for flattened mesh. \*\* done.
   3. Create faces for original mesh. \*\* done.
   4. Create faces for flattened mesh. \*\* done.
   5. Separate the original mesh from flattened mesh. \*\* done.
2. Check in code.
3. Change event handler to consider which mesh the event is coming from: <https://stackoverflow.com/questions/39351388/control-the-mouse-click-event-with-a-subplot-rather-than-a-figure-in-matplotlib>
4. Check in code.

Dec 15, 2019

# Draw straight lines in the original mesh and map to flattened mesh.

# Implement mirror barycentric dots.

TODO:

1. Event handler on click to add dots to Original Mesh. \*\* done.
2. Event handler on click to add dots to Flattened Mesh. \*\* done.
3. Check in code.
4. Get the coordinates.
   1. Get the barycentric coordinates for the point. \*\* done.
   2. Plot the barycentric coordinates in the Flattened Mesh. \*\* done.
   3. Get the barycentric coordinates from the Flattened Mesh. \*\* done.
   4. Plot the barycentric coordinates onto the Original Mesh. \*\* done.
5. Check in code.

Dec 25, 2019

# Draw straight lines in the original mesh and map to flattened mesh.

# Implement mirror barycentric dots.

TODO:

1. Matplotlib tutorial about Figures and Axes: <https://realpython.com/python-matplotlib-guide/>
2. Add blob drawing area.
3. Potentially utilize Tkinter: <https://matplotlib.org/3.1.0/gallery/user_interfaces/embedding_in_tk_sgskip.html>
4. Potential base for draw program: <https://matplotlib.org/3.1.1/gallery/event_handling/ginput_manual_clabel_sgskip.html#sphx-glr-gallery-event-handling-ginput-manual-clabel-sgskip-py>
5. Check in code.

# Heading 1

## Heading 2

Subtitle

Date

# Heading 1

## Heading 2

Subtitle

Date

# Heading 1

## Heading 2

Subtitle

Date

# Heading 1

## Heading 2

Subtitle

Date

# Heading 1

## Heading 2

Subtitle

Date

# Heading 1

## Heading 2