

Week 3 Document Submission

Jacob Dallaire

September 7, 2024

1. Paper

Xiao-Xiao Niu, Ching Y. Suen,

A novel hybrid CNN–SVM classifier for recognizing handwritten digits, *Pattern Recognition*, Volume 45, Issue 4, 2012, Pages 1318-1325, ISSN 0031-3203, <https://doi.org/10.1016/j.patcog.2011.09.021>.

(<https://www.sciencedirect.com/science/article/pii/S0031320311004006>)

SUMMARY

The passage discusses a novel hybrid model that combines Convolutional Neural Network (CNN) as a feature extractor and Support Vector Machine (SVM) as a recognizer for recognizing handwritten digits. The hybrid model achieved high recognition rates on the MNIST digit database, outperforming individual classifiers and demonstrating improved accuracy and reliability in handwritten digit recognition tasks. The study highlights the synergy between CNN and SVM in enhancing recognition performance and reliability in pattern recognition applications.

2. Scripts

A script was begun to query the API to retrieve all image urls but was determined that was not a viable solution due to rate limits. I found an export function on iNaturalist that creates a CSV file of requested information. There is a 200,000 row limit per file but multiple can be requested.

Using this CSV file I created the following script to query the open data aws bucket to download all images onto my machine. Unfortunately, the runtime to download all images is >18hours at time of writing. It will be necessary to multithread the script and add a check for existing images in the save folder if a delta run is desired.

```
def download_images(filepath):
    df = pd.read_csv(filepath)
    taxon_i={}
    for index, row in df.iterrows():
        taxon_id = row['taxon_id']
        image_url = row['image_url']

        if not pd.isnull(image_url) and 'https://inaturalist-open-
data.s3.amazonaws.com/photos' in image_url:
            if taxon_id not in taxon_i:
                taxon_i[taxon_id] = 0
            taxon_i[taxon_id]+=1
            save_path =
f"{folder_path}/{taxon_id}/{taxon_id}_{taxon_i[taxon_id]}.jpg"
```

```
os.makedirs( os.path.dirname(save_path),exist_ok=True)
urllib.request.urlretrieve(image_url, save_path)
if index%1000 == 0:
    print("In Progress")
print("Done")
```

3. Documentation

Created export query on iNaturalist example structure:

Queryhas[]=photos&quality_grade=research&identifications=any&rank=species&taxon_ids[]=36488,36391,36455 **Columns** id, license, image_url, taxon_id

Wrote a function to download and save all photos by the url provided in the iNaturalist exported CSV. The script also creates a file structure to aid in the ease of labeling with each taxon having its own folder.

4. Next Weeks Proposal

I will change my image download script to run in multiple threads as the current runtime is far too long. I will also create a simple classifier using CNN architecture that I will work on improving over the coming weeks.

Week 3 report

Ruiqing Wang | Lizard CV team

Time slot response:

▪ What progress did you make in the last week?

1. Review papers on DeepLabCut
2. Help assembling paper report submissions and address submission situation.
3. Using GoogleColab to run the DeepLabCut demo project
4. Check on videos I got from Dr. Strout and start data preparation.
5. Set up PACE account access

▪ What are you planning on working on next?

1. Met with Dr. Stroud and discuss about further methods and resources
2. Start labelling and preparing my training dataset
3. Check my PACE allocation and evaluate performance

▪ Is anything blocking you from getting work done?

N/A

Abstract

Paper:

Real-Time Closed-Loop Feedback in Behavioral Time Scales Using DeepLabCut

Summary:

This paper discusses using deep learning to track a mouse's whisker movements in real-time without markers. By training a deep neural network offline and transferring it to work in real-time, the researchers were able to track whisker positions and trigger outputs based on whisker movements, which can be useful for studying the relationship between movement and neural activity in mice. The challenges in DNN-based tracking approaches include communication delays between devices, data transfer latency, and the need for optimizing hardware-software interactions to enhance real-time tracking efficiency in behavioral neurophysiology research.

Methodology:

The position estimation involves utilizing DeepLabCut 2.1.3. The DNN model processes one frame at a time with a batch size of one, incorporating a GPU-based inference stage, to estimate the positions of three specific whiskers on mice. Each mouse had a unique model trained with the default ResNet-50 network architecture, tailored to the individual characteristics of their whiskers. The DNN models consistently identified the tips of the specified whiskers across the behavioral recording sessions, providing a reliable method for tracking and analyzing whisker movements in real-time experiments.

Scripts and Code Blocks

This week I tried GoogleColab to run a DLC demo. GoogleColab provides free GPUs and has all library set up so it is relatively easy to install the DLC toolkit.

Here is the code I used:

```

▶ # Clone the entire deeplabcut repo so we can use the demo data:
!git clone -l -s https://github.com/DeepLabCut/DeepLabCut.git cloned-DLC-repo
%cd cloned-DLC-repo
!ls

[ ] %cd /content/cloned-DLC-repo/examples/openfield-Pranav-2018-10-30
!ls

[ ] %cd /content/cloned-DLC-repo/
!pip install ".[tf]"

▶ import deeplabcut

↔ Loading DLC 2.3.10...
DLC loaded in light mode; you cannot use any GUI (labeling, relabeling and standalone GUI)

[ ] #create a path variable that links to the config file:
path_config_file = '/content/cloned-DLC-repo/examples/openfield-Pranav-2018-10-30/config.yaml'

# Loading example data set:
deeplabcut.load_demo_data(path_config_file)

[ ] #let's also change the display and save_iters just in case Colab takes away the GPU...
#if that happens, you can reload from a saved point. Typically, you want to train to 200,000 + iterations.
#more info and there are more things you can set: https://github.com/DeepLabCut/DeepLabCut/wiki/DOCSTRINGS#train_network

deeplabcut.train_network(path_config_file, shuffle=1, displayiters=100, saveiters=500, maxiters=10000)

[ ] %matplotlib notebook
deeplabcut.evaluate_network(path_config_file, plotting=True)

[ ] Start coding or generate with AI.

[ ] videofile_path_lizard = ['/content/Lizard_test/0002_1_lizard.MP4'] #Enter the list of videos to analyze.
deeplabcut.analyze_videos(path_config_file, videofile_path_lizard, videotype='.mp4')

[ ] deeplabcut.create_labeled_video(path_config_file, videofile_path_lizard)

[ ] deeplabcut.plot_trajectories(path_config_file, videofile_path)

↔ Loading /content/cloned-DLC-repo/examples/openfield-Pranav-2018-10-30/videos/m3v1mp4.mp4 and data.
Plots created! Please check the directory "plot-poses" within the video directory

```

In GoogleColab, it is relatively easy to set up environment by using code: “!pip install “[.tf]” ” in its set-up folder. I successfully get the trained network and the processed video with landmark. However, when processing the sample video, I met challenges which prevented from getting the trained dataset. I need start video preparations next week.

For video extraction and network training, here is what I got:

https://github.com/DeepLabCut/DeepLabCut/wiki/DOCSTRINGS#train_network

Signature: `deeplabcut.create_new_project(project, experimenter, videos, working_directory=None, copy_videos=False, videotype='.avi')`
Docstring:

Creates a new project directory, sub-directories and a basic configuration file. The configuration file is loaded with the default values. Change its parameters to your projects need.

Signature: `deeplabcut.extract_frames(config, mode='automatic', algo='kmeans', crop=False, userfeedback=True, cluster_step=1, cluster_resizewidth=30, cluster_color=False, opencv=True, slider_width=25)`
Docstring:

Extracts frames from the videos in the config.yaml file. Only the videos in the config.yaml will be used to select the frames.

Signature: `deeplabcut.label_frames(config, multiple=False)`

Docstring:

Manually label/annotate the extracted frames. Update the list of body parts you want to localize in the config.yaml file first.

This will be my work next week to assemble the data blocks when I started labeling and preparing the labeled training data.

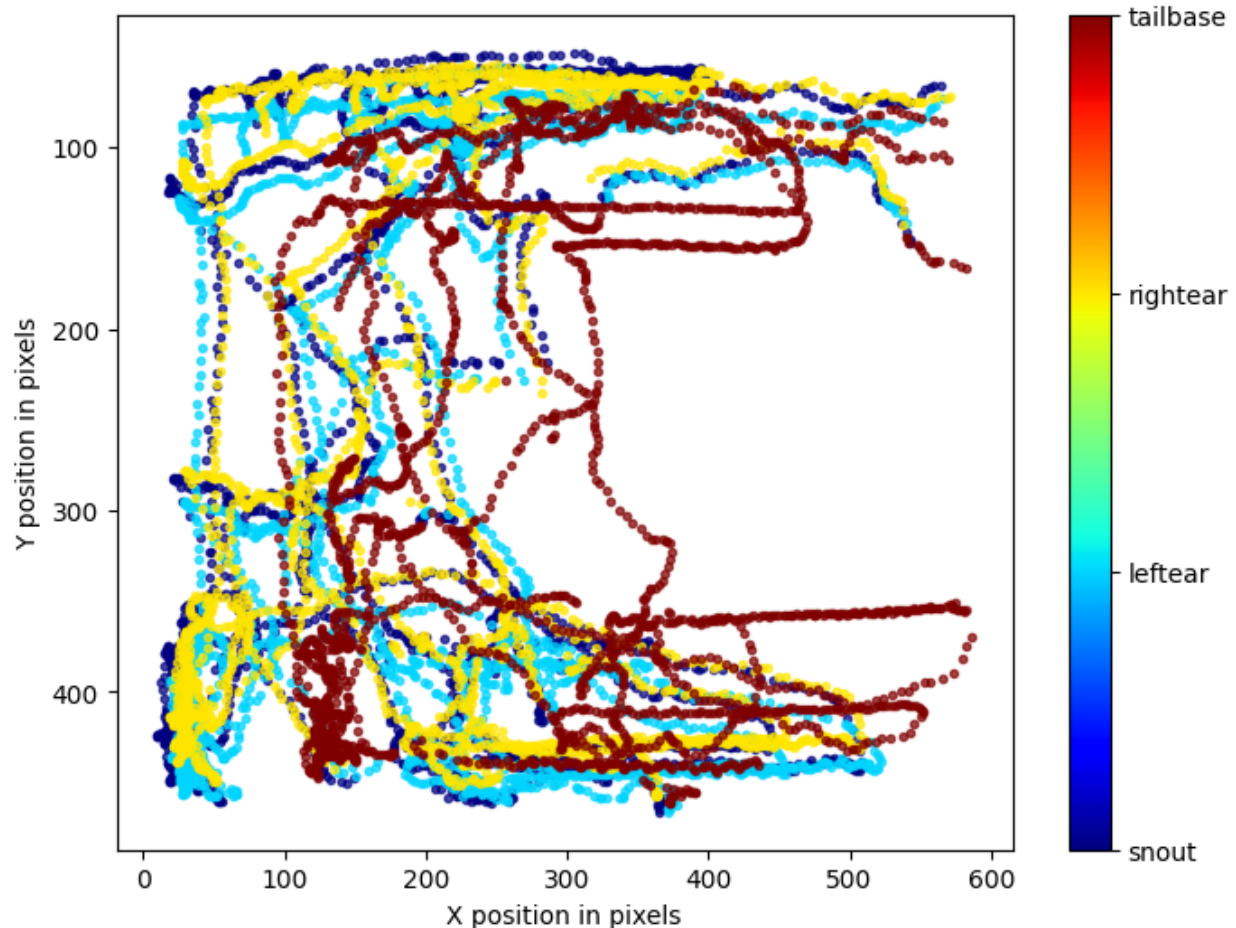
As for PACE access, I successfully login in school PACE, and I will started set up environment and test the GPU performance.

Documentation

My demo code is stored in https://github.com/RuiqingW20/HAAG_Research/blob/main/deeplabcut_test_demo.ipynb

Results Visualization

Here is the trajectory picture showing the x,y position I got after processing:



Proof of work

Please check github link: https://github.com/RuiqingW20/HAAG_Research-

Here is the information I got when I successfully login in school hpc:

Terms of Use

This computer system is the property of Georgia Institute of Technology. Any user of this system must comply with all Institute and Board of Regents policies, including the Acceptable Use Policy (AUP), Data Privacy Policy (DPP) and Cyber Security Policy (CSP), see <http://b.gatech.edu/it-policies>. Users should have no expectation of privacy, as any and all files on this system may be intercepted, monitored, copied, inspected, and/or disclosed to authorized personnel in order to meet institute obligations.

By using this system, I acknowledge and consent to these terms.

```
#####  
# Welcome to GT Instructional Cluster #  
#####
```

If you require assistance with this system, please contact your course instructor or teaching assistant (TA).

```
[rwan753@login-ice-4 ~]$ █
```

Next Week's Proposal

1. Start Labeling and preparing training dataset
2. Met with Dr. Strout and Bree to discuss current progress and aim
3. Check my PACE allocation performance by running my former sample code

Weekly Report

Philip Woolley

2024-09-06

Time Log Reponse:

- What Progress did you make in the last week? - Segmented Jaw and teeth of example image from dataset, sent to Dr. Stroud for review. Created python script for visualizing TIFF stacks from CT images.
- What are you planning on working on next? - Create script to automatically load and process full dataset of images.
- Is there anything blocking you? - None at this time

1 Abstract

Abstract

Accurate segmentation of the jaw (i.e., mandible and maxilla) and the teeth in cone beam computed tomography (CBCT) scans is essential for orthodontic diagnosis and treatment planning. Although various (semi)automated methods have been proposed to segment the jaw or the teeth, there is still a lack of fully automated segmentation methods that can simultaneously segment both anatomic structures in CBCT scans (i.e., multiclass segmentation). In this study, we aimed to train and validate a mixed-scale dense (MS-D) convolutional neural network for multiclass segmentation of the jaw, the teeth, and the background in CBCT scans. Thirty CBCT scans were obtained from patients who had undergone orthodontic treatment. Gold standard segmentation labels were manually created by 4 dentists. As a benchmark, we also evaluated MS-D networks that segmented the jaw or the teeth (i.e., binary segmentation). All segmented CBCT scans were converted to virtual 3-dimensional (3D) models. The segmentation performance of all trained MS-D networks was assessed by the Dice similarity coefficient and surface deviation. The CBCT scans segmented by the MS-D network demonstrated a large overlap with the gold standard segmentations (Dice similarity coefficient: 0.934 ± 0.019 , jaw; 0.945 ± 0.021 , teeth). The MS-D network-based 3D models of the jaw and the teeth showed minor surface deviations when compared with the corresponding gold standard 3D models (0.390 ± 0.093 mm, jaw; 0.204 ± 0.061 mm, teeth). The MS-D network took approximately 25 s to segment 1 CBCT scan, whereas manual segmentation took about 5 h. This study showed that multiclass segmentation of jaw and teeth was accurate and its performance was comparable to binary segmentation. The MS-D network trained for multiclass segmentation would therefore make patient-specific orthodontic treatment more feasible by strongly reducing the time required to segment multiple anatomic structures in CBCT scans.

Summary This paper proposes using the Multi-scale Dense convolutional neural network to segment teeth and jaw bones in CT scans. The authors performed multi-class segmentation for jaw, teeth, or neither. Crucially, the authors pre-crop the CT scans to remove layers which do not have any jaw or teeth present, which is not a fully automated step which my project could easily implement. This could hint towards using a hierarchical CNN to first select slices of interest and then perform segmentation. While this paper does a clear job of explaining their process for data collection and methods, I believe the figures and tables in this paper do not communicate information in the way that readers expect. For example, the model diagram has no labels or information about what layers consist of. This is an aspect that I would plan to improve on if my method is developed into an article.

Citation

Wang H, Minnema J, Batenburg KJ, Forouzanfar T, Hu FJ, Wu G. Multiclass CBCT Image Segmentation for Orthodontics with Deep Learning. *Journal of Dental Research*. 2021;100(9):943-949. doi:10.1177/00220345211005338

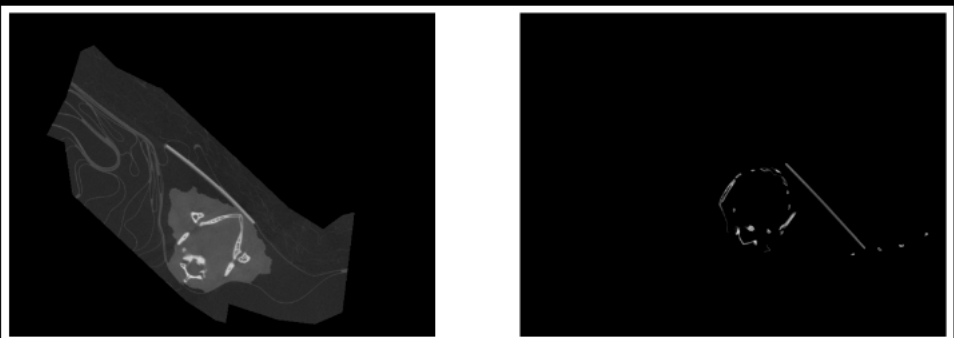
2 Scripts and Code Blocks

This week, I created the `testtif.ipynb` notebook which shows steps for opening, thresholding, and visualizing slices from CT volumes. This is done using several python libraries, including `imageio`

```
1 fig, ax = plt.subplots(1, 2, figsize=(15, 10))
2 # Draw the image in grayscale
3 ax[0].imshow(vol[700, :, :], cmap='gray', vmin=0, vmax=62000)
4
5 # Draw the image with greater contrast
6 ax[1].imshow(vol[1200, :, :], cmap='gray', vmin=23000, vmax=62000)
7
8 # Remove axis ticks and labels
9 ax[0].axis('off')
10 ax[1].axis('off')
```

✓ 0.4s Python

(-0.5, 700.5, 533.5, -0.5)



3 Documentation

Documentation for test.tif.ipynb is not complete, as the code from this notebook is intended to be rewritten into a final script which will be tested and documented.

https://www.morphosource.org/projects/0000C1059?locale=en&page=11&sort=publication_status_s
List of available MicroCT Datasets of anolis lizards that will be used for this project. When infrastructure for data storage is ready I will prepare documentation detailing the downloading and storage process.

<https://slicermorph.github.io/> Documentation for SlicerMorph, an extension of the 3D slicer tool commonly used by Biologists.

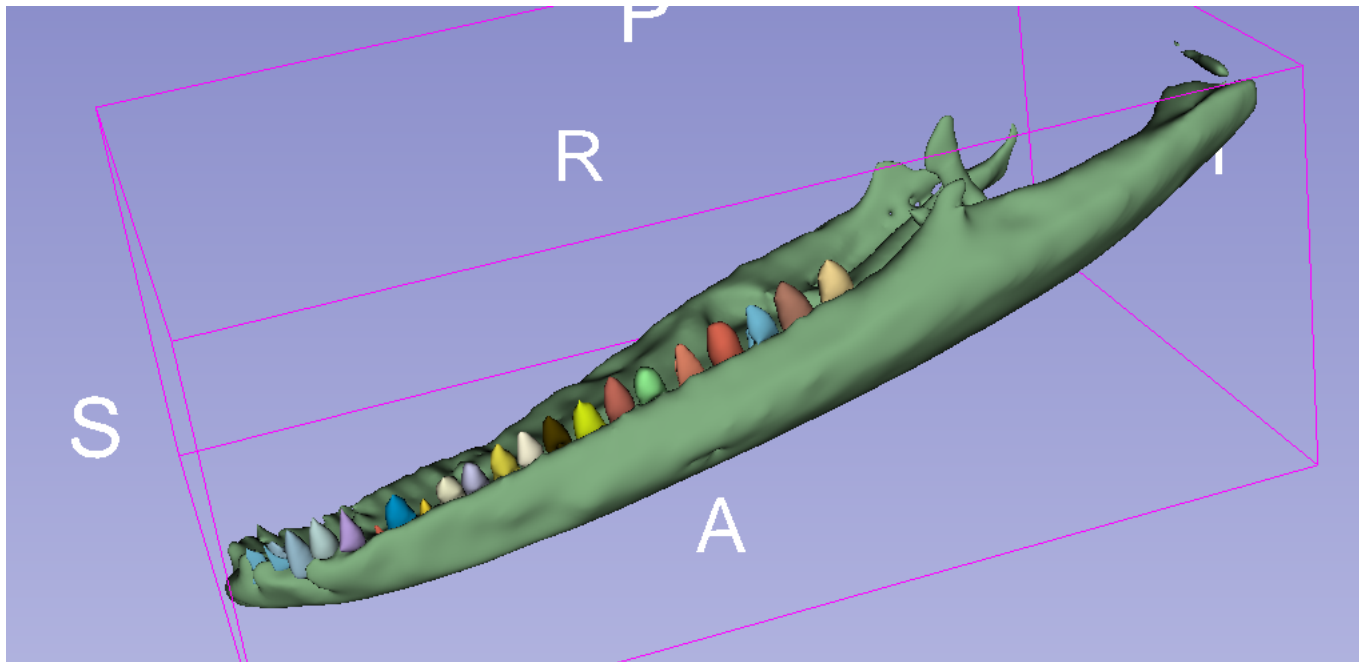
<https://github.com/jmhuie/SlicerBiomech> Documentation for the Dental Dynamics module, which is a 3D slicer extension for calculating tooth stress from jaw segmentations. the outputs from my segmentation pipeline will need to be compatible with this module for analysis.

4 Script Validation (Optional)

There are no scripts to validate this week.

5 Results Visualization

Here is an image showing the example segmentation of lower jaw and teeth that I performed.



6 Proof of Work

See Code published on github as well as image showed above.

7 Next Week's Proposal

- Create script to load and threshold all images in dataset
- Investigate UNet +ResNeXT segmentation method, see if applicable to jaw or tooth segmentation
- Keep up with any required blog posts for webmaster role

Week 3 Document Submission

Lizard X-RAY Landmark Group

Mercedes Quintana

Abstracts:

URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers. The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions¹, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

Summary: ResNet is a competition winning neural network architecture that allows the user to considerably increase depth by reformulating the layers in a neural network.

Scripts and Code Blocks:

From discussion at the Lizard group meeting and further discord guidance, I have now have architecture to visualize specific landmark error between the ground truth and model output. This visualization will have a couple additions such as a histogram of error lengths and a separate rose plot of dimension.

Found in landmark_skew.py:

Read in test tps data for model and ground truth -> calculate common feature on each X-Ray for a conversion factor to millimeters -> calculate difference between ground truth and model output -> convert to millimeters -> estimate a kernel density function of error -> display with matplotlib

Documentation:

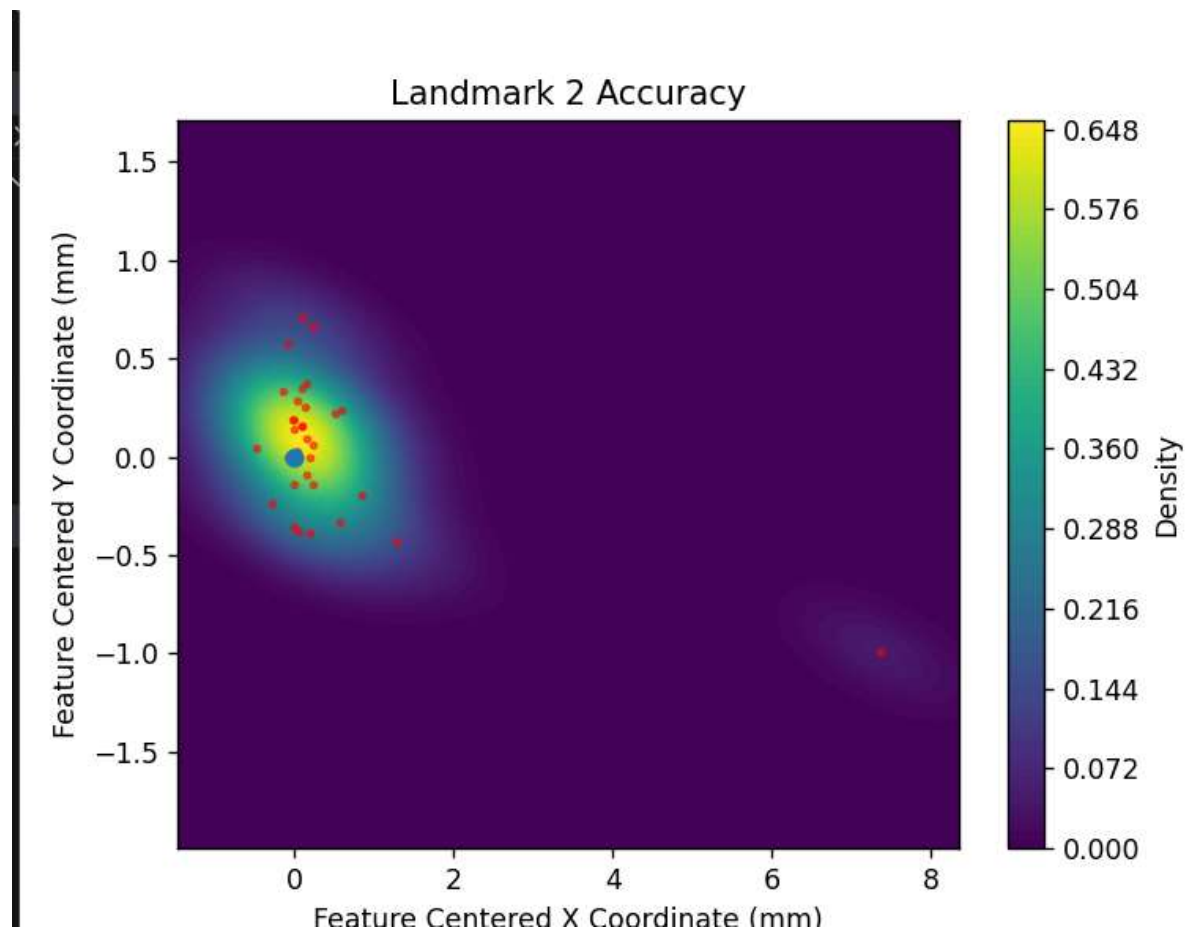
1. Read in data (found in XRAY Lizard Github) for ground and model output
2. Choose a specific landmark to visualize
3. Find the differences between the ground truth and the model output in pixels.
4. Convert and store millimeter conversion using each individual staple.
5. Estimate kernel density function
6. Display with matplotlib

Script Validation:

I have no validation steps now.

Results Visualization / Proof of Work:

This is the error visualized from Landmark 2.



Next Week Proposal:

Next week I plan to rework the image preprocessing now that I understand ml-morph better. Specifically, the images must be centered for the library to work better. I also plan to start the hyper tuning that will result from this. I also need to set a meeting with Dr. Porto

to discuss ml-morph, the current pipeline we use, and about possible updates to the method.