Week 8 Document Submission

Jacob Dallaire

October 11, 2024

1. Paper

B. Pande, K. Padamwar, S. Bhattacharya, S. Roshan and M. Bhamare, "A Review of Image Annotation Tools for Object Detection," *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, Salem, India, 2022, pp. 976-982, doi: 10.1109/ICAAIC53929.2022.9792665.

SUMMARY

The introduction highlights the significant advancements in deep learning, particularly in object detection, which has become a critical area within computer vision due to its diverse applications, including surveillance and medical imaging. It emphasizes the importance of data annotation in the lifecycle of object detection projects, noting that the quality of image annotations directly influences model performance. The text also discusses various annotation strategies—such as in-house, outsourcing, and crowdsourcing—each with distinct trade-offs in terms of cost, quality, and security, underscoring the necessity for careful selection of annotation tools and methodologies to optimize outcomes in machine learning tasks.

2. Scripts

```python
import tensorflow as tf
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load your custom-trained object detection model
detection_model = tf.saved_model.load('@TODO train my own model')

# Load an input image
image = tf.io.read_file('F:/LizardCV/Raw/36391_4333245.jpg')
image = tf.image.decode_jpeg(image, channels=3)  # or tf.image.decode_png if it's a PNG
image = tf.image.resize(image, [640, 640])  # Resize to the input size expected by the model
#image = tf.cast(image, tf.uint8)
input_tensor = tf.convert_to_tensor(image)
input_tensor = input_tensor[tf.newaxis, ...]  # Add batch dimension
image_np = image.numpy()  # Convert to NumPy array for OpenCV

# Run object detection
detections = detection_model(input_tensor)

# Access the output tensor
```

```python
output_tensor = detections[0]  # Get the first tensor
output_array = output_tensor.numpy()  # Convert to NumPy array

# Initialize lists to store detected boxes, scores, and class IDs
boxes = []
scores = []
class_ids = []

# Set a confidence threshold
confidence_threshold = 0.5

# Iterate over the detections
for detection in output_array[0]:  # Loop through the detections for the first image
    # Extract bounding box and scores
    box = detection[:4]  # First four elements are the box coordinates
    score = detection[4]  # The fifth element is the objectness score

    # If the score is above the confidence threshold, save the results
    if score >= confidence_threshold:
        boxes.append(box)
        scores.append(score)

        # Get the class ID with the highest score
        class_score = detection[5:]  # Class scores
        class_id = np.argmax(class_score)  # Get the class index of the max score
        class_ids.append(class_id)

# Convert lists to NumPy arrays for easier manipulation
boxes = np.array(boxes)
scores = np.array(scores)
class_ids = np.array(class_ids)

# Define a color map for visualization
colors = np.random.randint(0, 255, size=(len(boxes), 3), dtype='uint8')

# Draw bounding boxes on the original image
for i in range(len(boxes)):
    ymin, xmin, ymax, xmax = boxes[i]

    # Convert to original image scale
    xmin = int(xmin * image.shape[1])
    xmax = int(xmax * image.shape[1])
    ymin = int(ymin * image.shape[0])
    ymax = int(ymax * image.shape[0])

    # Draw bounding box and label
    cv2.rectangle(image_np, (xmin, ymin), (xmax, ymax), color=colors[i].tolist(), thickness=2)
    cv2.putText(image_np, f'ID: {class_ids[i]} {scores[i]:.2f}', (xmin, ymin - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, colors[i].tolist(), 2)

# Show the image with detections
plt.imshow(cv2.cvtColor(image_np, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

I produced a script to use for testing some object detection models I found on model zoos. I had very little success with them and have decided it is necessary to transfer learn to create my own model that accounts for the complex nature of the training data set.

3. Documentation

I tested some existing models I found on model zoos. I found frequently they had too many classes and misidentified the Anoles frequently or for specialized models they had datasets with fixed background and were not able to identify the Anoles from the background. Figure 1 is an example of the multi class detectors and failures to identify the anoles correctly.

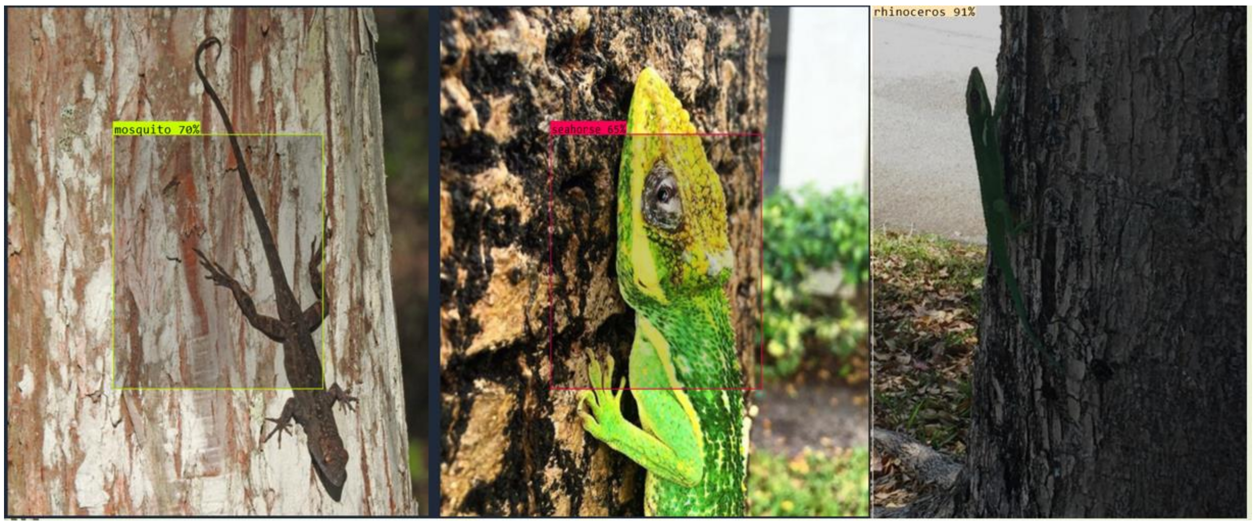


Figure 1. Multiclass object detector

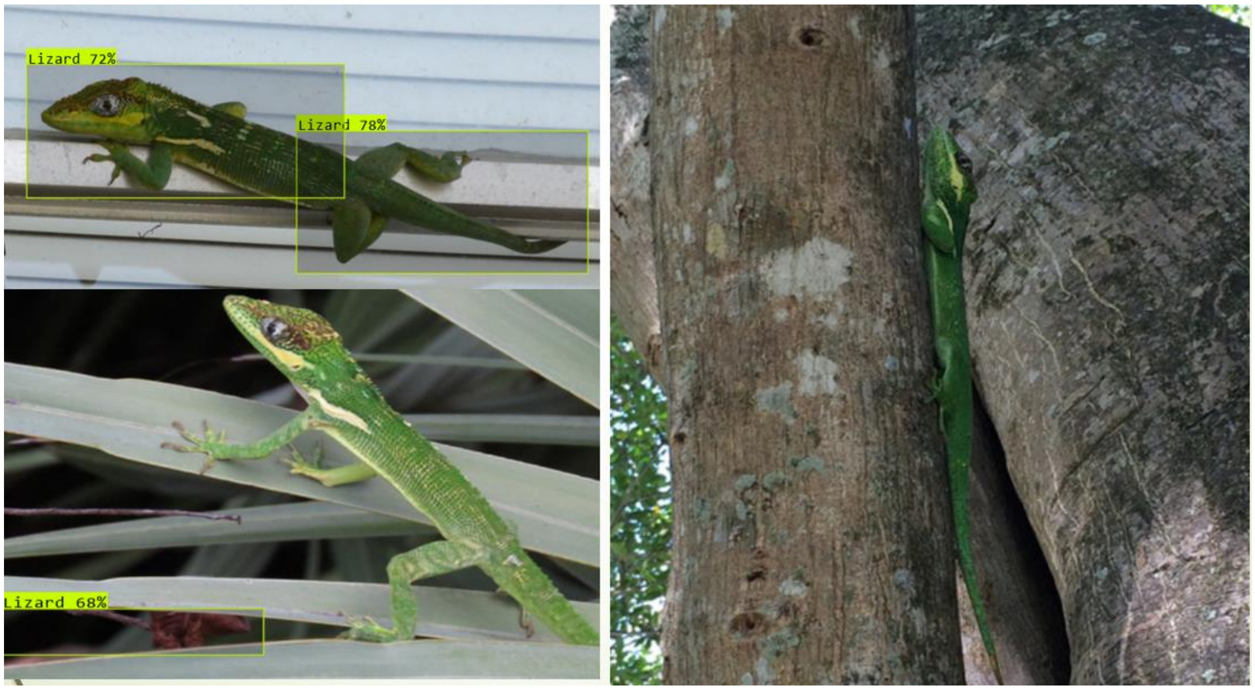


Figure 2. Specialized Lizard detector

Figure 2 is an example of the failures of the specialized Lizard detector models I tested. Incorrectly overlapping bounding boxes on a single object, misplacement of a bounding box or inability to detect objects at all.

I have been evaluating a few option on bbox annotation tools to create my own dataset to fine tune a generalized model. I decided to go with labeme (https://github.com/wkentaro/labelme?tab=readme-ov-file) as it is a light weight application.

4. Next Weeks Proposal

I aim to produce about 1000 annotated images to use to train an object detection model.

Weekly Report

Philip Woolley

2024-10-11

Time Log Reponse:

- Researched and chose quality metric for measuring segmentation - Identified possible automatic alignment tools. Continued segmenting training data
- What are you planning on working on next? - Continue segmenting training data. Retrain model with additional data and new metric. Change website structure to match request from Bree
- Is there anything blocking you? - Access to "Appearance" tab on wordpress needed to rearrange navigation menu

1 Abstract

Abstract

Open3D is an open-source library that supports rapid development of software that deals with 3D data. The Open3D frontend exposes a set of carefully selected data structures and algorithms in both C++ and Python. The backend is highly optimized and is set up for parallelization. Open3D was developed from a clean slate with a small and carefully considered set of dependencies. It can be set up on different platforms and compiled from source with mini mal effort. The code is clean, consistently styled, and main tained via a clear code review mechanism. Open3D has been used in a number of published research projects and is actively deployed in the cloud. We welcome contributions from the open-source community.

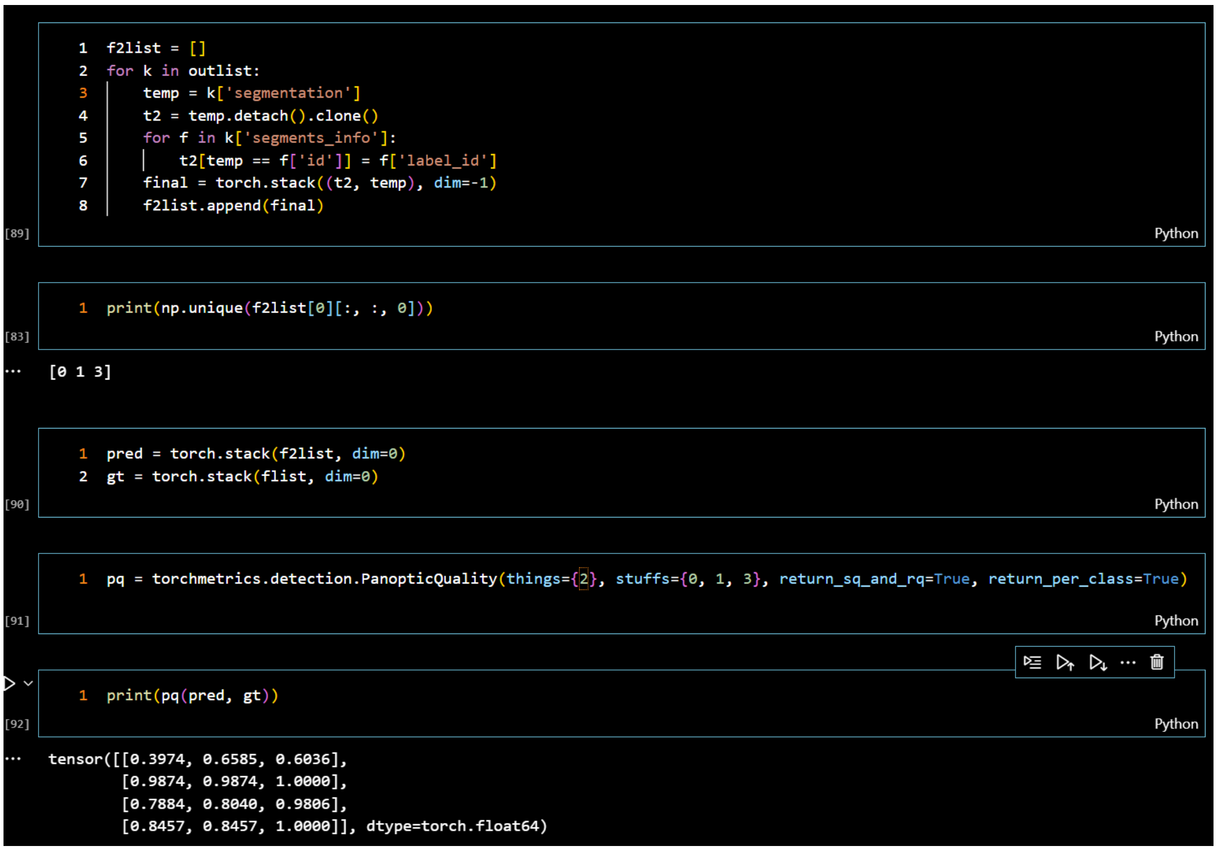
Summary This paper describes the design and implementation details for the Open3D library. This is an open source library for viewing and manipulating 3d data in python or C++. For my project I will be working with the 3d CT lizard scans and trying to automatically adjust their orientation to match the slicing axes, so several of the capabilities of this library will be useful. The most useful are converting 3d arrays to point cloud format, which is widely used for image registration, as well as functions to create a global to local registration pipeline. The paper includes several quality visualizations, and does a good job of explaining the workings of the library and common use cases. I would like to emulate something similar, while targeted for a low-code audience, in the documentation of my project.

Citation

Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun. "Open3D: A modern library for 3D data processing." arXiv preprint arXiv:1801.09847 (2018).

2 Scripts and Code Blocks

I have begun sketching the model's post processing steps in the sketchpostprocess.ipynb notebook. This week, I added a function to this notebook showing how to calculate Panoptiq Quality for the segments output by the model. This is the chosen accuracy metric for model training. Below images include implementation of the metric and the formula for how to calculate it.



```
[89]
f2list = []
for k in outlist:
    temp = k['segmentation']
    t2 = temp.detach().clone()
    for f in k['segments_info']:
        t2[temp == f['id']] = f['label_id']
    final = torch.stack((t2, temp), dim=-1)
    f2list.append(final)
```

```
[83]
print(np.unique(f2list[0][:, :, 0]))
```

```
[0 1 3]
```

```
[90]
pred = torch.stack(f2list, dim=0)
gt = torch.stack(flist, dim=0)
```

```
[91]
pq = torchmetrics.detection.PanopticQuality(things={2}, stuffs={0, 1, 3}, return_sq_and_rq=True, return_per_class=True)
```

```
[92]
print(pq(pred, gt))
```

```
tensor([[0.3974, 0.6585, 0.6036],
        [0.9874, 0.9874, 1.0000],
        [0.7884, 0.8040, 0.9806],
        [0.8457, 0.8457, 1.0000]], dtype=torch.float64)
```

$$\mathrm{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}$$

3 Documentation

The VisualizeModelResults.ipynb notebook is used for creating and viewing images of model output on validation data. Users provide a pretrained model and validation dataset, and this notebook inferences all of the images in the dataset and allows the user to review the output segmentations against the ground truth manual segmentations.

The DataProcess.ipynb notebook is used for converting slicer volume files (.nrrd and .seg.nrrd) into a HuggingFace dataset for use with the pretrained Mask2Former model. Volumes should be added to the "vols" folder, and segmentation volumes should be added to the "masks" folder.

https://www.morphosource.org/projects/0000C1059?locale=enpage=11sort=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. This is used for loading stacks of .tiff images as a volume in 3d slicer.

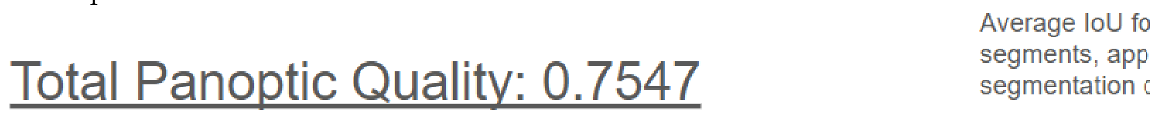
https://github.com/jmhuie/SlicerBiomech Documentation for the Dental Dynamics module, which is a 3D slicer extension for calculating tooth stress from jaw segmentations.

the outputs from my segmentation pipeline will need to be compatible with this module for analysis.

4 Script Validation (Optional)

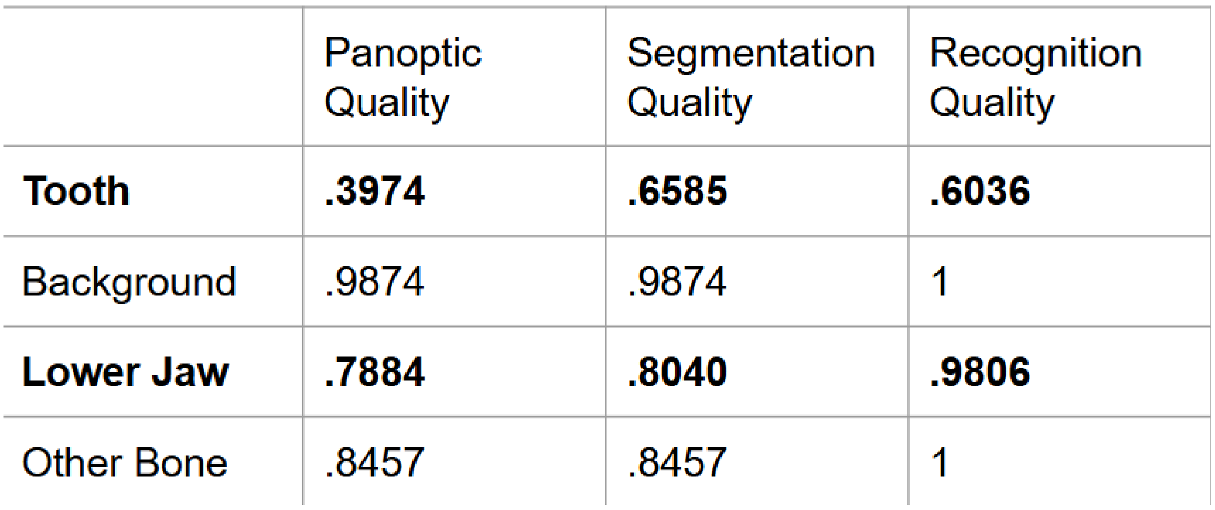
5 Results Visualization

These are the printed results on validation set for the first version of the model.



Total Panoptic Quality: 0.7547

Average IoU fo
segments, app
segmentation



	Panoptic Quality	Segmentation Quality	Recognition Quality
Tooth	**.3974**	**.6585**	**.6036**
Background	.9874	.9874	1
Lower Jaw	**.7884**	**.8040**	**.9806**
Other Bone	.8457	.8457	1

6 Proof of Work

Please see Code Blocks and Results Visualization

7 Next Week's Proposal

- Continue segmenting training data for ML panoptic segmentation model
- Develop testing script for 3D image registration for converting coordinate systems
- Reformat blog page as requested by Bree

Week 8 Document Submission

Lizard X-RAY Landmark Group

Mercedes Quintana

What progress did you make in the last week?

- Continued to work on website
- Quantified differences between landmarked results on manual and auto images

What are you planning on working on next?

- Find the best training data for the model
- Continue to update the website
- Find if the manual or auto images perform better in a machine learning model

Is anything blocking you from getting work done?

- Nope

Abstracts:

URL: https://openaccess.thecvf.com/content_CVPR_2019/papers/Kirillov_Panoptic_Segmentation_CVPR_2019_paper.pdf

Panoptic Segmentation

We propose and study a task we name panoptic segmentation (PS). Panoptic segmentation unifies the typically distinct tasks of semantic segmentation (assign a class label to each pixel) and instance segmentation (detect and segment each object instance). The proposed task requires generating a coherent scene segmentation that is rich and complete, an important step toward real-world vision systems. While early work in computer vision addressed related image/scene parsing tasks, these are not currently popular, possibly due to lack of appropriate metrics or associated recognition challenges. To address this, we propose a novel panoptic quality (PQ) metric that captures performance for all classes (stuff and things) in an interpretable and unified manner. Using the proposed metric, we perform a rigorous study of both human and machine performance for PS on three existing datasets, revealing interesting insights about the task. The aim of our work is to revive the interest of the community in a more unified view of image segmentation. For more analysis and up-todate results, please check the arXiv version of the paper: https://arxiv.org/abs/1801.00868.

Summary: The paper aims to create a quality metric to judge performance of a machine learning model on two distinct tasks: semantic segmentation and instance segmentation.

Scripts and Code Blocks:

I created a script to unify the two different image systems and then plot each individual lizard and overlay both landmarking styles on top of it called compare_manual_auto_landmarks.py.

Documentation:

compare_manual_auto_landmarks.py:

1. Read in tps files for both image sets using the image mapping txt I put together
2. Put the images in the same order
3. Create a folder and fill with all matching lizards

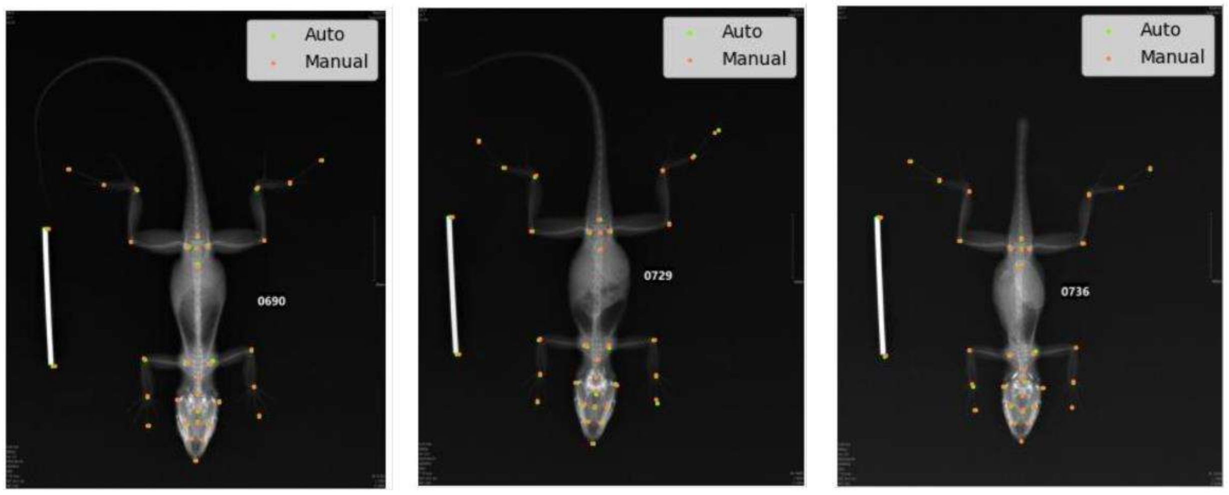
Script Validation:

I have no validation steps now.

Results Visualization / Proof of Work:

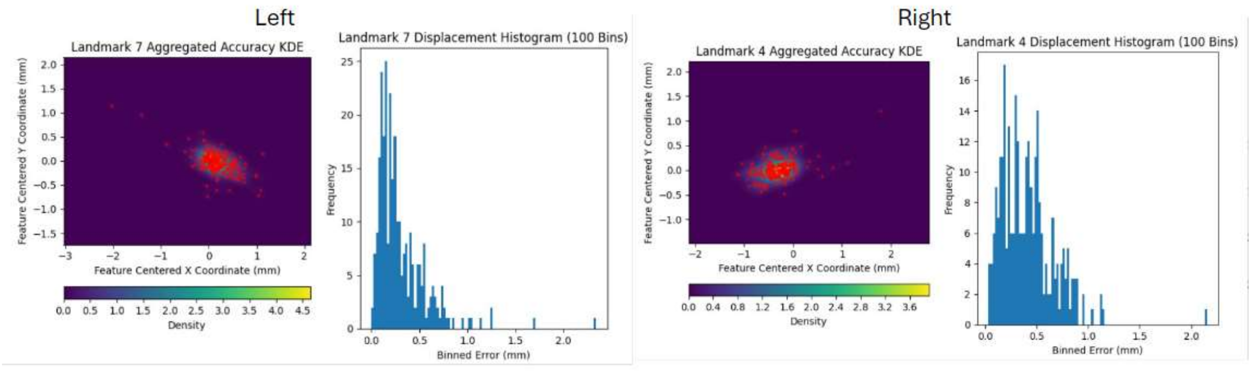
Here is a slide showing three of the lizards from the dataset with the mapping styles overlaid. The rest are available in the lab dropbox.

The agreement between the two methods looks consistent



Here are some statistics computed about the similarity between the two landmarking styles:

Eye sockets could be marked in two places



Either on the inside or outside of the skull, which could make for some variation between image batches

Next Week Proposal:

I plan to keep working on the website to keep it updated with the new meetings and work done. I plan to look through the combined dataset with Jon and find the best lizards to use in the training dataset. Then train two separate models on each dataset and compare them.