# Weekly Report

Philip Woolley

2024-08-23

## 1 Abstract

**Abstract** 1. Landmark-based geometric morphometrics has emerged as an essential discipline for the quantitative analysis of size and shape in ecology and evolution. With the ever-increasing density of digitized landmarks, the possible development of a fully automated method of landmark placement has attracted considerable attention.Despite the recent progress in image registration techniques, which could provide a pathway to automation, three-dimensional (3D) morphometric data are still mainly gathered by trained experts. For the most part, the large infrastructure requirements necessary to perform image-based registration, together with its system specificity and its overall speed, have prevented its wide dissemination.2. Here, we propose and implement a general and lightweight point cloud-based approach to automatically collect high-dimensional landmark data in 3D surfaces(Automated Landmarking through Point cloud Alignment and Correspondence Analysis). Our framework possesses several advantages compared with image-based approaches. First, it presents comparable landmarking accuracy, despite relying on a single, random reference specimen and much sparser sampling of thestructure's surface. Second, it can be efficiently run on consumer-grade personal computers. Finally, it is general and can be applied at the intraspecific level to any biological structure of interest, regardless of whether anatomical atlases are available.3. Our validation procedures indicate that the method can recover intraspecific pat-terns of morphological variation that are largely comparable to those obtained bymanual digitization, indicating that the use of an automated landmarking approach should not result in different conclusions regarding the nature of multivariate pat-terns of morphological variation.4. The proposed point cloud-based approach has the potential to increase the scaleand reproducibility of morphometrics research. To allow ALPACA to be used out-of-the- box by users with no prior programming experience, we implemented itas a SlicerMorph module. SlicerMorph is an extension that enables geometric morphometrics data collection and 3D specimen analysis within the open-source3D Slicer biomedical visualization ecosystem. We expect that convenient access to this platform will make ALPACA broadly applicable within ecology and evolution.

**Summary** This paper details a tool called ALPACA for Point Cloud-based automated landmarking of 3D morphometric data. The ALPACA process begins with selecting a representative sample of the data. This representative target sample is landmarked, and is then used as the source to project landmark points on to each of the other samples in the set. Each sample is downsampled, resized, and then distorted to match the target as closely as possible, the landmarks from this distorted sample are then projected back on to the original shape, resulting in landmarks which match those made on the target sample. The authors note that this process is very sensitive to the choice of target sample, and so it is recommended to compare the results of ALPACA using different choices of the target sample.

**Citation** Porto, A., Rolfe, S., Maga, A. M. (2021). ALPACA: A fast and accurate computer vision approach for automated landmarking of three-dimensional biological structures. Methods in Ecology and Evolution, 12, 2129–2144. https://doi.org/10.1111/2041-210X.13689

# 2 Scripts and Code Blocks

Unfortunately I do not yet have access to the Github Organization, so I do not have any code for this week.

# 3 Documentation

https://www.morphosource.org/projects/0000C1059?locale=enpage=11sort=publication_status_ssi+
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.

# 4 Script Validation (Optional)

There are no scripts to validate this week.

# 5 Results Visualization

There are no results visualizations this week.

# 6 Proof of Work

For this week, my work has been signing up for the course and reviewing the ALPACA paper. I believe the ALPACA tool will be the backbone of the automated landmarking pipeline, so an understanding of how the tool works is very important going forward.

# 7 Next Week's Proposal

- Attend First group meetings for Lizard CV project.
- Prepare data storage for lizard CT scans
- Download all data for the project
- Clarify project deliverables with Bri and/or Prof. Stroud

# Week 1 Document Submission

Jacob Dallaire

August 22, 2024

## 1. Paper

Kumar, N. *et al.* (2012). Leafsnap: A Computer Vision System for Automatic Plant Species Identification. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds) Computer Vision – ECCV 2012. ECCV 2012. Lecture Notes in Computer Science, vol 7573. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33709-3_36

**SUMMARY**

This paper introduces a mobile app called Leafsnap that can identify tree species by analyzing photographs of their leaves using computer vision technology. The app has achieved high performance and has nearly a million users.

## 2. Scripts

No scripts have been written yet as Project description was just given.

## 3. Documentation

This week I focused on registration and starting a role as meeting manager for Lizard CV group.

I have created a weekly meeting for the group.

I have been looking into the iNaturalist API documentation and gathering the ID codes for taxa as well as Thawly and Stroud users.

## 4. Results Vizualization

Using the iNaturalist API to fetch IDs

**Taxa IDs**

Anolis sagrei 116461
Anolis carolinensis 36514
Anolis cristatellus 36488
Anolis equestris 36391
Anolis distichus 36455

**User IDs**

cthawley 139486
james_stroud_lizardsontheloose 3721

API call to get list of user's identifications
https://api.inaturalist.org/v1/identifications?user_id=139486&current=true&taxon_id=116461&order=desc&order_by=created_at&only_id=false

## 5. Next Weeks Proposal

I will write a script to parse out all the observation IDs associated with the identifications made. Using this list of IDs I can serialize curl commands to download all images.

Preferably I would like to find a way to bulk download these for the iNaturalist AWS Open Dataset. With the images downloaded I can create some utility functions to preprocess the images and prepare them to be used as training data to create a new model.

I am awaiting access to the HAAG repositories.

Week 1 report on Aug 24, 2024
Ruiqing Wang

**Abstracts:**

Using DeepLabCut for 3D markerless pose estimation across species and behaviors
https://www.nature.com/articles/s41593-018-0209-y

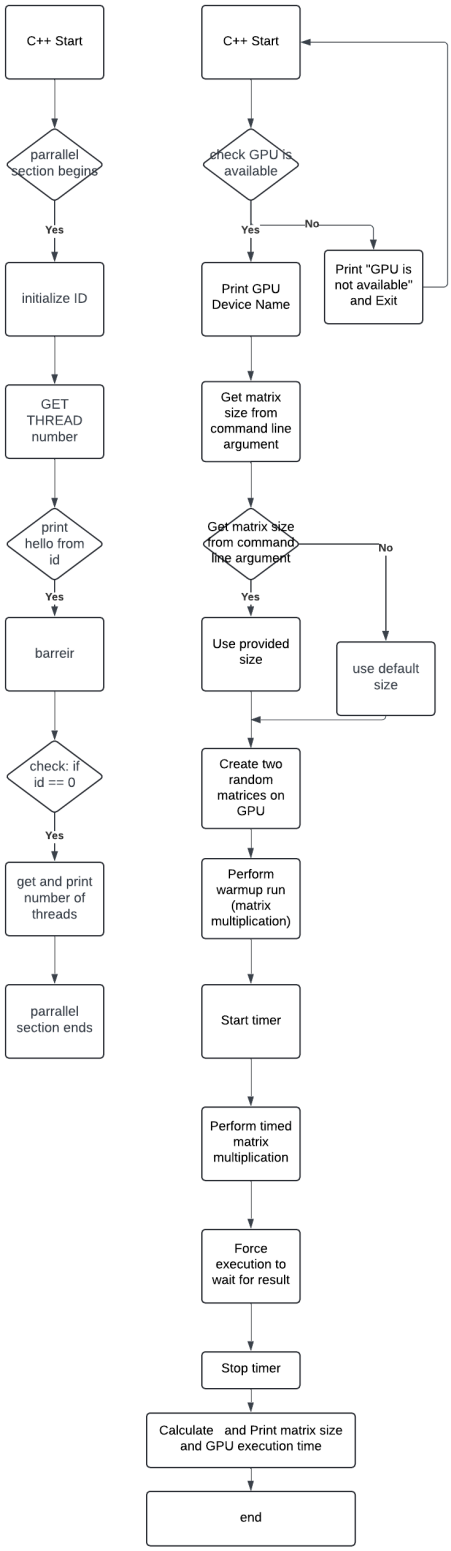
Abstract: Noninvasive behavioral tracking of animals during experiments is critical to many scientific pursuits. Extracting the poses of animals without using markers is often essential to measuring behavioral effects in biomechanics, genetics, ethology, and neuroscience. However, extracting detailed poses without markers in dynamically changing backgrounds has been challenging. We recently introduced an open-source toolbox called DeepLabCut that builds on a state-of-the-art human pose-estimation algorithm to allow a user to train a deep neural network with limited training data to precisely track user-defined features that match human labeling accuracy. Here, we provide an updated toolbox, developed as a Python package, that includes new features such as graphical user interfaces (GUIs), performance improvements, and active-learning-based network refinement. We provide a step-by-step procedure for using DeepLabCut that guides the user in creating a tailored, reusable analysis pipeline with a graphical processing unit (GPU) in 1–12 h (depending on frame size). Additionally, we provide Docker environments and Jupyter Notebooks that can be run on cloud resources such as Google Colaboratory.

Summary: The text describes a method for markerless pose estimation using deep neural networks, specifically focusing on the feature detector architecture from DeeperCut. By leveraging transfer learning with pretraining on ImageNet, the network can achieve high accuracy in pose estimation tasks with minimal training data, around 200 images. The also the mark with human labeled and computer labeled , and most of them fit very well. This approach allows for the localization of various body parts in different experimental settings, as demonstrated by tracking body parts in mice and fruit flies during specific tasks.

**Scripts and Code Blocks**

This week my job is to get familiar with computing on PACE. To submit a job on PACE, I made sample code that could be submitted. I also create another simple C++ code to test my processor using MPI library. Since I haven't been added to repo, I used my own repo to get familiar with project and vision control. My code could be reached here:
https://github.com/RuiqingW20/HAAG_Research-

## Flowchart:


C++ Start
parrallel section begins
Yes
initialize ID
GET THREAD number
print hello from id
Yes
barreir
check: if id == 0
Yes
get and print number of threads
parrallel section ends
C++ Start
check GPU is available
Yes
No
Print "GPU is not available" and Exit
Print GPU Device Name
Get matrix size from command line argument
Get matrix size from command line argument
Yes
No
Use provided size
use default size
Create two random matrices on GPU
Perform warmup run (matrix multiplication)
Start timer
Perform timed matrix multiplication
Force execution to wait for result
Stop timer
Calculate and Print matrix size and GPU execution time
end

Data: cpu_test.cpp

- id: Thread-specific ID.
- nthreads: Number of threads in the parallel region.
- Output messages are printed to the console.

Important Code Blocks:

- Parallel Region: Initializes and uses thread-specific data.
- Barrier: Synchronizes threads at a specific point.
- Condition Check: Outputs the number of threads from the master thread.

Data: gpu_test.py:

- Checks for GPU availability.
- Generates large random matrices directly on the GPU.
- Conducts a matrix multiplication operation, a common task in machine learning and scientific computing.
- Measures and reports the time taken for this operation.

Important Code Blocks:

- GPU Availability Check Ensuring that the computation runs on a GPU and verifying the device.
- Matrix Creation and Multiplication (on GPU): Matrix multiplication is a common operation that can significantly benefit from GPU acceleration, making it a good test for checking GPU functionality.
- **Timing:** Accurately measuring how long the operation takes on the GPU gives insight into performance.

## Documentation:

This C++ program demonstrates basic parallel programming using OpenMP. It prints a message from each thread in a parallel region and then displays the total number of threads from the master thread.
The gpu test program is designed to verify the availability and performance of a GPU using TensorFlow. It performs matrix multiplication on the GPU and measures the time taken for the computation.

**Results Visualization:**
My current account is still pending. Once I could have account access to PACE, I will attach my result here.

**Proof of Work:**

rwang753 update submitted job script

Code | Blame | 25 lines (21 loc) · 1002 Bytes

```
#!/bin/bash
#!/bin/bash
#SBATCH -JSlurmCPlusExample            # Job name
#SBATCH -account=rwang753    #tracking account
#SBATCH -N1 --ntasks-per-node=4          # Number of nodes and cores per node required
#SBATCH --mem-per-cpu=1G                 # Memory per core
#SBATCH -t1:00:00                               # Duration of the job (Ex: 15 mins)
#SBATCH -phive                     # queue name(where job is submitted)
#SBATCH -oReport-%j.out                  # Combined output and error messages file
#SBATCH --mail-type=BEGIN,END,FAIL       # Mail preferences
#SBATCH --mail-user=rwang753@gatech.edu # E-mail address for notifications
cd $SLURM_SUBMIT_DIR                     # Change to working directory

echo "TASKS_PER_NODE=" $SLURM_TASKS_PER_NODE
echo "NNODES=" $SLURM_NNODES
echo "NTASKS" $SLURM_NTASKS
echo "JOB_CPUS_PER_NODE" $SLURM_JOB_CPUS_PER_NODE
echo $SLURM_NODELIST

module load gcc   #remain to be changed
module load mvapich2

mpicxx main.cpp -o mpi_main
mpirun ./mpi_main
```

Here is my main code:

Code | Blame | 28 lines (25 loc) · 658 Bytes

```
//
//  main.cpp
//  test_threads
//
//  Created by Ruiqing Wang on 8/22/24.
//

#include <iostream>
#include <omp.h> // need this to call functions, omp_get_....

int main (int argc, char *argv[]) {
  int id;
  std::cout << "C++ Start" << std::endl;
#pragma omp parallel private(id)
  {  // Parallel Section begins
    id = omp_get_thread_num();
#pragma omp critical
    std::cout <<  "hello from " << id << std::endl;
#pragma omp barrier
    if ( id == 0 ) {
      int nthreads = omp_get_num_threads();
      std::cout << nthreads << " threads said hello!" << std::endl;
    }
  }  // Parallel Section ends
  std::cout << "End" << std::endl;

  return 0;
}
```

```python
def main():
    # Check if GPU is available
    if not tf.test.is_gpu_available():
        print("GPU is not available")
        return

    # Print GPU device name
    print("GPU Device Name:", tf.test.gpu_device_name())

    # Get the size of the matrix from command line argument
    if len(sys.argv) > 1:
        size = int(sys.argv[1])
    else:
        size = 5000  # Default size

    # Create two random matrices
    with tf.device('/GPU:0'):
        matrix_1 = tf.random.normal((size, size))
        matrix_2 = tf.random.normal((size, size))

        # Warmup run
        _ = tf.matmul(matrix_1, matrix_2)

        # Timed run
        start_time = time.time()
        result = tf.matmul(matrix_1, matrix_2)
        # Force execution to wait for result
        result.numpy()
        gpu_time = time.time() - start_time

    print(f"Matrix size: {size}x{size}")
    print(f"GPU time: {gpu_time:.4f} seconds")

if __name__ == "__main__":
    main()
```

Code Blame 15 lines (15 loc) · 751 Bytes

```
#!/bin/bash
#SBATCH -GPUExample_RW # Job name
#SBATCH -rwang753 # Charge account
#SBATCH -N1 --gres=gpu:1 # Number of nodes and GPUs required
#SBATCH --gres-flags=enforce-binding # Map CPUs to GPUs
#SBATCH --mem-per-gpu=12G # Memory per gpu
#SBATCH -t15 # Duration of the job (Ex: 15 mins)
#SBATCH -phive-gpu # Partition name (where job is submitted)
#SBATCH -oReport-%j.out # Combined output and error messages file
#SBATCH --mail-type=BEGIN,END,FAIL # Mail preferences
#SBATCH --mail-user=rwang753@gatech.edu # e-mail address for notifications
cd $HOME/slurm_gpu_example # Change to working directory created in
$HOME
module load tensorflow-gpu/2.9.0 # Load module dependencies
srun python $TENSORFLOWGPUROOT/gpu_test.py gpu 1000 # Run test example
```

## Next Week's Proposal:

- Get my own account on PACE
- Start small practice project on PACE
- Get familiar with DLC model
- Connect with PI and Bree to check the tasks
- Set up environment for DLC model

**Week 1 Document Submission**

**Mercedes Quintana**

**Abstracts:**

One Millisecond Face Alignment with an Ensemble of Regression Trees

This paper addresses the problem of Face Alignment for a single image. We show how an ensemble of regression trees can be used to estimate the face's landmark positions directly from a sparse subset of pixel intensities, achieving super-realtime performance with high quality predictions. We present a general framework based on gradient boosting for learning an ensemble of regression trees that optimizes the sum of square error loss and naturally handles missing or partially labelled data. We show how using appropriate priors exploiting the structure of image data helps with efficient feature selection. Different regularization strategies and its importance to combat overfitting are also investigated. In addition, we analyse the effect of the quantity of training data on the accuracy of the predictions and explore the effect of data augmentation using synthesized data.

Summary: This paper discusses a method using regression trees to accurately estimate facial landmark positions from images quickly. It also explores techniques like gradient boosting, regularization, and data augmentation to improve accuracy and combat overfitting.

**Scripts and Code Blocks:**

I spent the week learning about the project and downloading the appropriate packages: ml-morph, and Ayush's GitHub code. As of now, I still have imglab to download, the package to visualize lizard x-rays with landmark predictions, but I am hoping to have everything up and running soon.

**Documentation:**

I first downloaded ml-morph and Ayush's GitHub repositories. Using Visual Studio code, I created a virtual environment in order to install opencv-python and the other necessary packages to create functioning repositories as detailed in Ayush's documentation (URL shown below). To make sure the ml-morph was in working order, I used, at Ayush's suggestion, the tutorial project found in the GitHub ReadMe (URL found below).

Ayush Documentation: https://github.com/Human-Augment-Analytics/Lizard-XRAYs/tree/main

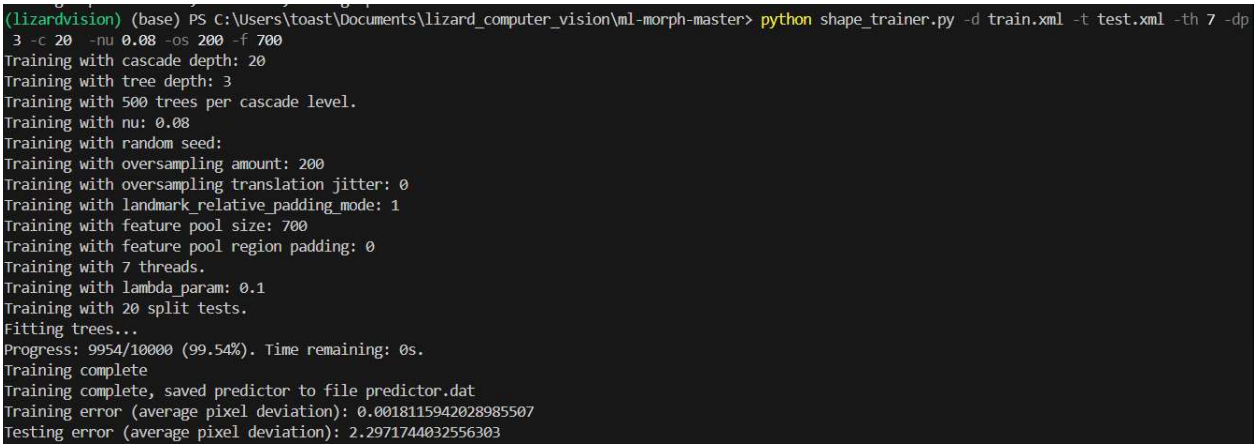
ml-morph: https://github.com/agporto/ml-morph

**Script Validation:**

I have no validation steps now.

**Results Visualization / Proof of Work:**

These are the results from training the tutorial project found on ml-morph:




**Next Week Proposal:**

Next week is my first lab meeting with Dr. Stroud. To track the accuracy of the model, Ayush has been using the average amount of deviation from each site across all the images. For this meeting, I would like to get more statistics on the deviation since these images are annotated by hand and averages are very susceptible to outliers. With these other forms of quantifying the model's performance, we will better understand how to improve the model.