HAAG Week 11 Report -Lizard Jaw Segmentation

Shuyu Tian

Time-Log

What did I do this week?

- I met with the computational advisor on code review for the code I have for rigid registration.
- I was recommended to provide justification and background information on why specific functions and choices in the code were made, specifically regarding the registration portion of the code
- I wrote up the justifications and background information to the computational advisor for review for our meeting next week
- I reviewed and tested the non-rigid registration code provided by Philip
- As the webpage manager and meeting leader of my group, I updated the webpage of the Stroud group with a subpage for the Lizard Jaw Segmentation group's weekly reports and group meeting recording link.

- What I will do next week
  - I will review with the computational advisor on the registration code provide for improvements and research direction if this process is not done well
- Blockers, things I want to flag, problems, etc.
  - If rigid and non-rigid registration methods are not producing usable data, manual segmentation work on 3Dslicer will be needed again and will be a big time-sink

Abstract:

Citizen science platforms like iNaturalist generate biodiversity data at an unprecedented scale, with observations on the order of hundreds of millions. However, extracting phenotypic information from these images, such as color of organisms, at such a large scale poses unique challenges for biologists. Some of the challenges are that manual extraction of phenotypic information can be subjective and time-consuming. Fortunately, with the maturation of computer vision and deep learning, there is an opportunity to automate large parts of the image processing pipeline. Here, I present SegColR, a user-friendly software package that leverages two state-of-the-art deep learning models - GroundingDINO and SegmentAnything - to enable automated segmentation and color extraction from images. The SegColR package provides an R-based interface, making it more accessible to evolutionary biologists and ecologists who may not have extensive coding experience. The SegColR pipeline allows users to load images, automatically segment them based on text prompts, and extract color information from the segmented regions. The package also includes visualization and data summarization functions to facilitate downstream analysis and interpretation of the results.

Link: https://www.biorxiv.org/content/10.1101/2024.07.28.605475v1.full

General summary: The paper introduces SlimSAM, a compact version of the Segment Anything Model (SAM), specifically tailored for biological image segmentation tasks. SlimSAM achieves a significant reduction in model size, utilizing only 1.4% of the original SAM's parameters, making it more efficient for practical applications. The authors demonstrate that SlimSAM maintains performance comparable to the original SAM across various biological imaging datasets. This advancement offers a more accessible and resource-efficient tool for researchers in the field of biological image analysis.

What did you do and prove it

1. Below are screenshots of non-rigid registration code provided as well as the justification and background information on my code for registration:

```
def draw_registration_result(source, target, transformation):
    source_temp = copy.deepcopy(source)
    target_temp = copy.deepcopy(target)
    source_temp.paint_uniform_color([1, 0.706, 0])
    target_temp.paint_uniform_color([0, 0.651, 0.929])
    source_temp.transform(transformation)
    o3d.visualization.draw_geometries([source_temp, target_temp],
                                      zoom=0.4559,
                                      front=[0.6452, -0.3036, -0.7011],
                                      lookat=[1.9892, 2.0208, 1.8945],
                                      up=[-0.2779, -0.9482, 0.1556])
```

```
def preprocess_point_cloud(pcd, voxel_size):
    #print(":: Downsample with a voxel size %.3f." % voxel_size)
    pcd_down = pcd.voxel_down_sample(voxel_size)

    radius_normal = voxel_size * 2
    #print(":: Estimate normal with search radius %.3f." % radius_normal)
    pcd_down.estimate_normals(
        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_normal, max_nn=30))

    radius_feature = voxel_size * 5
    #print(":: Compute FPFH feature with search radius %.3f." % radius_feature)
    pcd_fpfh = o3d.pipelines.registration.compute_fpfh_feature(
        pcd_down,
        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_feature, max_nn=100))
    return pcd_down, pcd_fpfh
```

```
def volume_to_point_cloud(volume, threshold=38000, equal_flag = False):
    temp = np.asarray((volume > threshold))
    if(equal_flag):
        temp = np.asarray((volume == threshold))
    #temp = np.asarray(temp < 60000)
    z, y, x = temp.nonzero()
    points = np.vstack((x, y, z)).T  # Transpose to get points in (N, 3) format
    return points
```

```
def prepare_dataset(voxel_size, source_file=None, target_file=None):
    #print(":: Load two point clouds and disturb initial pose.")

    #source = o3d.io.read_point_cloud('wd/head.ply')
    #target = o3d.io.read_point_cloud('wd/full.ply')
    source_data, _ = pynrrd.read(source_file)
    target_data, _ = pynrrd.read(target_file)


    source_points = volume_to_point_cloud(source_data, 1, True)
    target_points = volume_to_point_cloud(target_data)

    # Create Open3D point clouds
    source_pcd = o3d.geometry.PointCloud()
    target_pcd = o3d.geometry.PointCloud()


    source_pcd.points = o3d.utility.Vector3dVector(source_points)
    target_pcd.points = o3d.utility.Vector3dVector(target_points)
    #trans_init = np.asarray([[0.0, 0.0, 1.0, 0.0], [1.0, 0.0, 0.0, 0.0],
    #                         [0.0, 1.0, 0.0, 0.0], [0.0, 0.0, 0.0, 1.0]])
    #source.transform(trans_init)
    draw_registration_result(source_pcd, target_pcd, np.identity(4))

    source_down, source_fpfh = preprocess_point_cloud(source_pcd, voxel_size)
    target_down, target_fpfh = preprocess_point_cloud(target_pcd, voxel_size)
    return source_data, target_data, source_down, target_down, source_fpfh, target_fpfh
```

```
source, target, source_down, target_down, source_fpfh, target_fpfh = prepare_dataset(3, source_file, target_file)
```

```
def execute_global_registration(source_down, target_down, source_fpfh,
                                target_fpfh, voxel_size):
    distance_threshold = voxel_size * 1.5
    #print(":: RANSAC registration on downsampled point clouds.")
    #print("   Since the downsampling voxel size is %.3f," % voxel_size)
    #print("   we use a liberal distance threshold %.3f." % distance_threshold)
    result = o3d.pipelines.registration.registration_ransac_based_on_feature_matching(
        source_down, target_down, source_fpfh, target_fpfh, True,
        distance_threshold,
        o3d.pipelines.registration.TransformationEstimationPointToPoint(False),
        3, [
            o3d.pipelines.registration.CorrespondenceCheckerBasedOnEdgeLength(
                0.95),
            o3d.pipelines.registration.CorrespondenceCheckerBasedOnDistance(
                distance_threshold)
        ], o3d.pipelines.registration.RANSACConvergenceCriteria(200000, 100000))
    return result
```

```
result_ransac = execute_global_registration(source_down, target_down,
                                            source_fpfh, target_fpfh,
                                            8)
```

```
st = source_down.transform(result_ransac.transformation)
```

```
draw_registration_result(st, target_down, np.identity(4))
```

```
bbox = st.get_minimal_oriented_bounding_box()
bbox.extent = bbox.extent * 2
tt = target_down.crop(bbox)
#tt = target_down
```

```
acpd = probreg.cpd.AffineCPD(np.asarray(st.points), use_cuda=False)
start = time.time()
tf_param = probreg.bcpd.registration_bcpd(np.asarray(st.points), np.asarray(tt.points))
elapsed = time.time() - start
print("time: ", elapsed)
```

```
time:  130.8828580379486
```

```
result_ransac = execute_global_registration(source_down, target_down,
                                            source_fpfh, target_fpfh,
                                            8)
```

```
st = source_down.transform(result_ransac.transformation)
```

```
draw_registration_result(st, target_down, np.identity(4))
```

```
bbox = st.get_minimal_oriented_bounding_box()
bbox.extent = bbox.extent * 2
tt = target_down.crop(bbox)
#tt = target_down
```

```
acpd = probreg.cpd.AffineCPD(np.asarray(st.points), use_cuda=False)
start = time.time()
tf_param = probreg.bcpd.registration_bcpd(np.asarray(st.points), np.asarray(tt.points))
elapsed = time.time() - start
print("time: ", elapsed)
```

```
time:  130.8828580379486
```

```
tf_param
```

```
<probreg.transformation.CombinedTransformation at 0x2a58210f110>
```

```
result = copy.deepcopy(st)
result.points = tf_param.transform(result.points)
```

```
draw_registration_result(result, target_down, np.identity(4))
```

Summary of Jupyter Notebook Functions

Below is the summary of function usage in the Jupyter Notebook for 3D image registration using Open3D and NRRD volumes for lizard heads and jaws.

preprocess_point_cloud(pcd, voxel_size)

Downsamples the point cloud and computes surface normals and FPFH features. Used to reduce computation and prepare for feature-based matching.

volume_to_point_cloud(volume, threshold=None)

Converts a 3D volume to a point cloud using intensity thresholding. This transforms medical image data into point clouds.

segment_lower_jaw(point_cloud)

Filters the point cloud to isolate the lower jaw region based on bounding box limits. Helps focus alignment on relevant anatomical regions.

prepare_dataset(voxel_size, source_file, target_file)

Loads source and target NRRD files, converts them to point clouds, segments the lower jaw, and preprocesses them for registration. This is a wrapper for earlier steps.

compute_average_transformations(transformations)

Computes the average of multiple 4x4 transformation matrices. Useful for summarizing multiple registration results.

process(num_samples=20, max_iters=5)

Main driver function that runs the full registration workflow multiple times, applying ICP refinement and computing the average transformation.

visualize_registration(source, target, transformation)

Visualizes the source and target point clouds after applying a transformation. Useful for visual validation of registration results.

Additionally, I updated the Stroud lab webpage with my group's relevant information up to week 10 of this semester (see images below).

Home / Lizard Jaw Segmentation /

Lizard Jaw Segmentation Group Meetings and Recordings

Updated On March 20, 2025

Spring 2025 Week 11 Computational Advisor/Group Meeting Recording

https://gtvault-my.sharepoint.com/personal/stian40_gatech_edu/Documents/Recordings/Computational%20Advisor%20Meeting_%20Lizard%20Jaw%20Segmentation-20250319_170154-Meeting%20Recording.mp4?web=1&referrer=Teams.TEAMS-ELECTRON&referrerScenario=RecapOpenInStreamButton.view.2b8a23c1-2849-409b-8a09-0935ed879990

Spring 2025 Week 10 Group Meeting – No meetings this week due to scheduling conflicts

Spring 2025 Week 9 Computational Advisor/Group Meeting Recording

https://sites.gatech.edu/haagstroudprojects/wp-admin/user-new.php

Spring 2025 Week 8 Meeting Recording

Weekly Lizard Jaw Meeting-20250226_170110-Meeting Recording.mp4

Spring 2025 Week 7 Computational Advisor/Group Meeting Recording

Weekly Lizard Jaw Meeting-20250219_170647-Meeting Recording.mp4

Spring 2025 Week 6 Meeting Recording

Weekly Lizard Jaw Meeting-20250212_170746-Meeting Recording.mp4

Spring 2025 Week 5 Computational Advisor Meeting Recording

Lizard Jaw Segmentation_ Student Researcher _ Computational Advisor Meet-20250206_171303-Meeting Recording.mp4

Spring 2025 Week 4 Meeting Recording

Weekly Lizard Jaw Meeting-20250129_150624-Meeting Recording.mp4

Spring 2025 Week 3 Meeting Recording

Weekly Lizard Jaw Meeting-20250122_150152-Meeting Recording.mp4

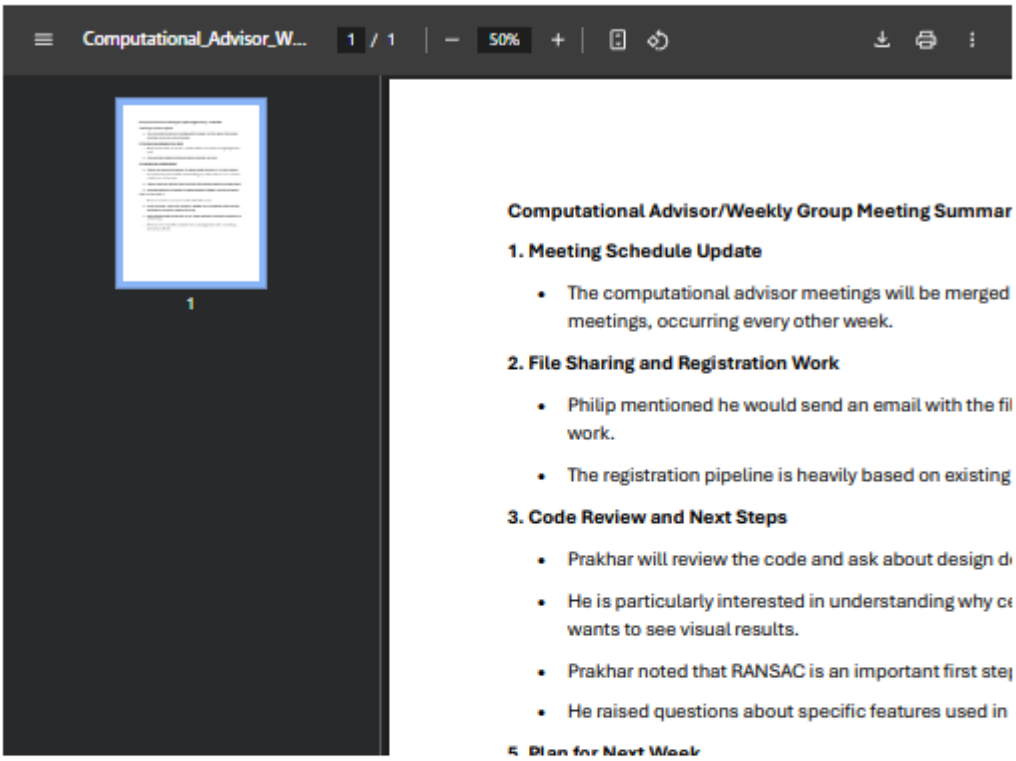
Spring 2025 Week 3 Computational Advisor Meeting Recording

Lizard Jaw Segmentation_ Student Reasearcher _ Computational Advisor Meet-20250120_150433-Meeting Recording.mp4

Lizard Jaw Segmentation Weekly Submissions

Updated On March 20, 2025

Week 11 Computational Advisor/Group Meeting



Computational_Advisor_Weekly_Meeting_Week_11 DOWNLOAD

Week 10 Weekly Report