HAAG Weekly Report

Nikita Angarski – 3D Modeling

Week 4

Time-Log

- What did you do this week?
  - Continued reading up on the non-rigid point registration method, which presented many knowledge gaps mainly to do with the kernel-based function smoothing including Fourier Transform, and the variational functional solution to the transformation problem
  - I got pycpd working with both the testing and the target dataset we are to use for the research. This included adding utility functions to read in Splicer data and editing an example to work with real data.
  - Reviewed past lectures and assigned myself to the AV role for the seminars program, and reached out to Victor who did this role in semesters' past.
- What are you going to do next week
  - Perform PCA process on the point set data.
  - Use these PCA's on getting a replacement kernel to use that's not the Gaussian example used in the original algorithm.
- Blockers, things you want to flag, problems, etc.
  - There's still a few remaining knowledge gaps to do with the non-rigid application for the point drift, but I think I have enough background knowledge to get started on

Abstracts:

A new point matching algorithm for non-rigid registration

https://www.cise.ufl.edu/~anand/pdf/rangarajan_cviu_si_final.pdf

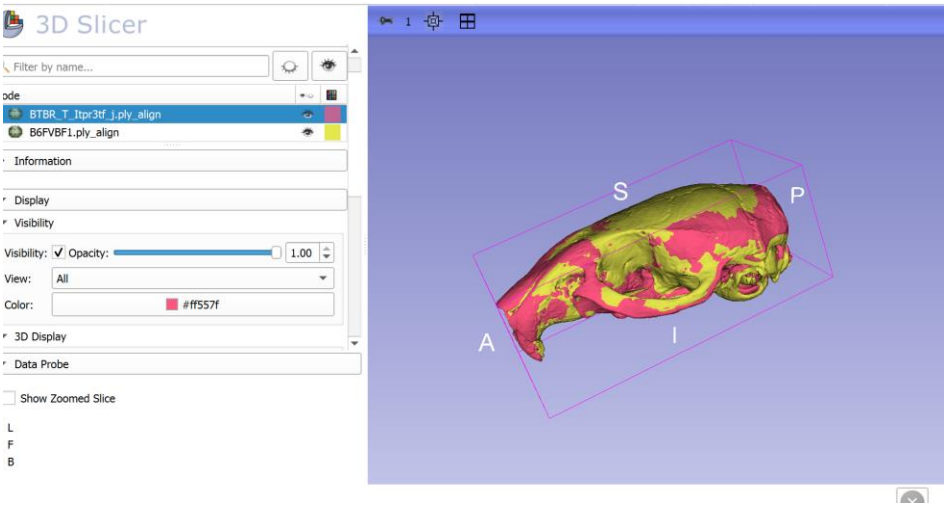
Abstract: Feature-based methods for non-rigid registration frequently encounter the correspondence problem. Regardless of whether points, lines, curves or surface parameterizations are used, feature-based non-rigid matching requires us to automatically solve for correspondences between two sets of features. In addition, there could be many features in either set that have no counterparts in the other. This outlier rejection problem further complicates an already difficult correspondence problem. We formulate feature-based non-rigid registration as a non-rigid point matching problem. After a careful review of the problem and an in-depth examination of two types of methods previously designed for rigid robust point matching (RPM), we propose a new general framework for non-rigid point matching. We consider it a general framework because it does not depend on any particular form of spatial mapping. We have also developed an algorithm—the TPS-RPM algorithm—with the thin-plate spline (TPS) as the parameterization of the non-rigid spatial mapping and the softassign for the correspondence. The performance of the TPS-RPM algorithm is demonstrated and validated in a series of carefully designed synthetic experiments. In each of these experiments, an empirical comparison with the popular iterated closest point (ICP) algorithm is also provided. Finally, we apply the algorithm to the problem of non-rigid registration of cortical anatomical structures which is required in brain mapping. While these results are somewhat preliminary, they clearly demonstrate the applicability of our approach to real world tasks involving feature-based non-rigid registration.

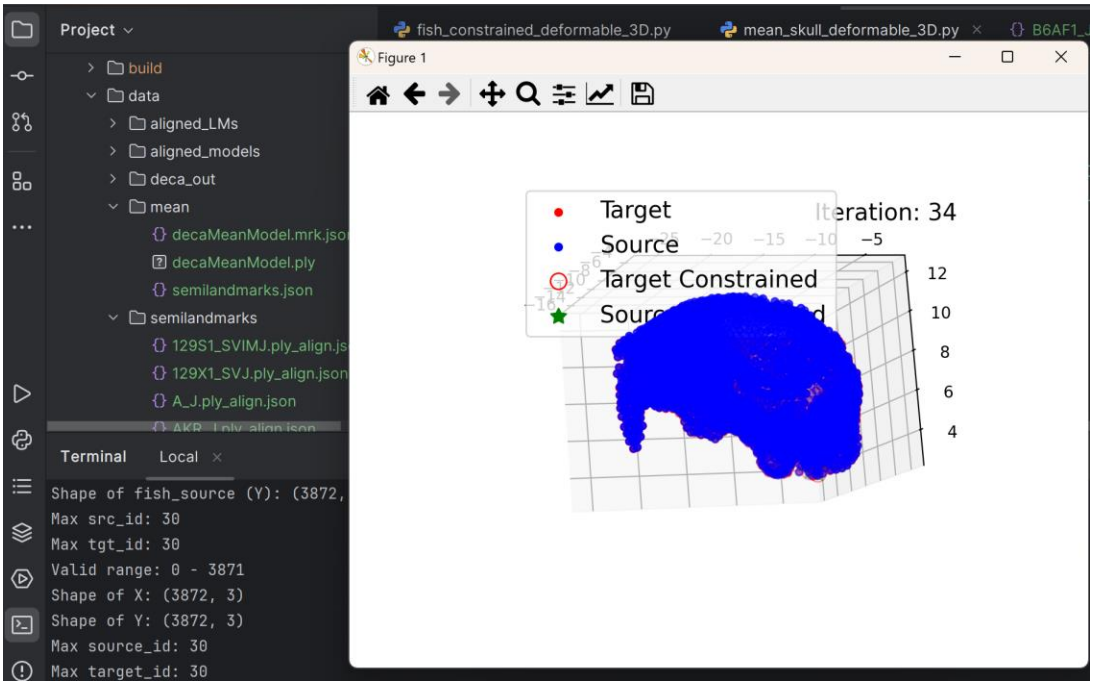
Summary: This paper presents a novel algorithm, TPS-RPM, for non-rigid point matching, a crucial problem in computer vision and medical image analysis. The algorithm addresses the challenges of **automatically finding correspondences between points in two sets**, handling **outliers (points without matches)**, and determining the **non-rigid transformation mapping one set onto the other**. The authors achieve this by combining a **thin-plate spline (TPS) for modeling non-rigid deformations** with a **softassign approach for probabilistic correspondence matching** within a deterministic annealing framework. The effectiveness of TPS-RPM is demonstrated through synthetic experiments, comparing its performance favorably to the Iterated Closest Point (ICP) algorithm, and showcased in a preliminary application to brain mapping.

What did you do and prove it

This week I continued to read the literature, especially some more background on the math behind how non rigid point registration is done. I had enough background knowledge on the project filled in to get started on testing the code and verifying what's going on in the paper and cross-checking what's happening in the codebase. I got it working with the included

data, and then moved onto making it adaptable to the dataset we received, which worked even with thousands of datapoints, as opposed to the included example's hundreds.





The image is comparing the slicer model and the python cpd algorithm output in the Python code.

Link to Two Recent commits:

Utility function + small point representation: https://github.com/Nikitos1865/pycpd-Porto/commit/c8bbea1c59bfdc53e07ca0be3934be3bba0f6025

Example with mouse skull data: https://github.com/Nikitos1865/pycpd-Porto/commit/daf124ee042daf3ee53a49165e68bf205e5d9586