Project 14

Motion Compensation and Evaluation of 3D Head Reconstruction

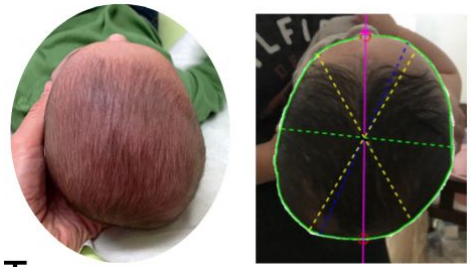
PediaMetrix Inc.

Checkpoint Presentation

Student: Unnat Antani
Mentors: Can Kocabalkanli , Ozgur Guler, Reza Seifabadi

A brief review

- Our major aim to be able to capture the head deformities on baby heads.
- Current deployed solution uses 2D vision methods to capture the necessary dimensions.



- A problem is that it loses a lot of depth information in 2D method
- 3D scanners can capture that information in a much better and detailed manner

Goals

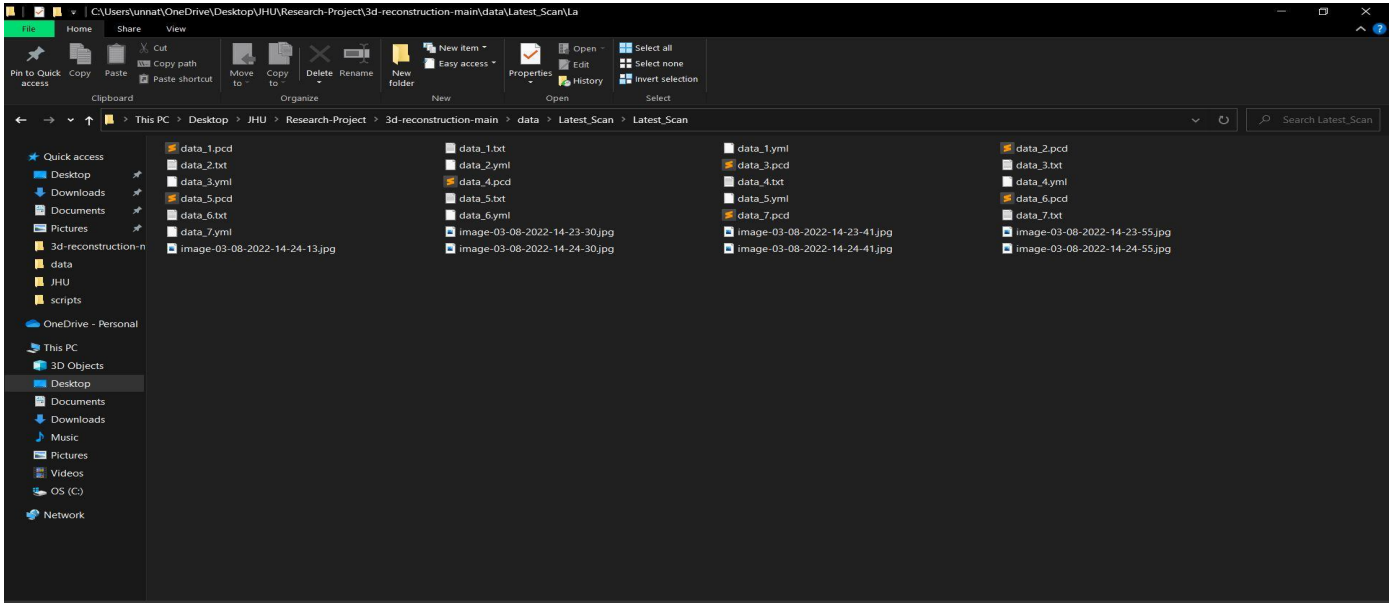
- The 3D scanning method assumes static head when taking scans, but that is not a correct assumption
- Even when taking scans, the motion of the camera as well as the baby head introduces error in obtained scans
- The major aim is to develop an algorithm which can robustly work for such cases.
- It also needs to be tested for angular resolution and for motional resolution

Current Status/Technical Approach

- The work so far was to set up the system and get the pipeline working again, getting rid of any bugs
- Take multiple new scans and methodically testing each one.
- The Sensor and the Phantom head are as shown below:

- App interface:

- The scans are uploaded on an AWS bucket and needs to be downloaded manually.
- The data formats captured are: YAML file consisting the RGB and depth information and a JPG image consisting the RGB information.
- The pipeline uses the camera parameters of focal length and generates point cloud using the YAML file
- The generated point clouds are stored as XYZ coordinates in a text file. The pipeline can also store it as pcd file



Multiway Registration[1][2]

- Multiway registration is the process of aligning multiple pieces of geometry in a global space.
- The input is a set of geometries(i.e., scans P_i) and output is a set of rigid transformations T_i so that the final output is $T_i * P_i$
- It works on representing the geometries as Pose graphs.
- The graph is such that the nodes represent the geometry and the edges represents the transformation between two geometries which overlap
- This intermediate transformation is obtained via Point-to-plane ICP
- The global optimization performs twice on the pose graph.
- The first pass optimizes poses for the original pose graph taking all edges into account and does its best to remove any false edges and alignments.
- The second pass runs without them and produces a tight global alignment

- They partition pose graph edges into two classes. **Odometry edges** connect temporally close, neighboring nodes. A local registration algorithm such as ICP can reliably align them. **Loop closure edges** connect any non-neighboring nodes. The alignment is found by global registration and is less reliable.
- Then, the problem is converted to an optimization algorithm of minimizing the RMSE error between the overlapping nodes on the graph.
- We also display the fitness, which is the amount of overlap between the surfaces after every iteration.

```
[Open3D DEBUG] ICP Iteration #84: Fitness 0.6763, RMSE 2.3144
[Open3D DEBUG] Residual : 3.07e+00 (# of elements : 29072)
[Open3D DEBUG] ICP Iteration #85: Fitness 0.6763, RMSE 2.3145
[Open3D DEBUG] Residual : 3.07e+00 (# of elements : 29072)
[Open3D DEBUG] ICP Iteration #86: Fitness 0.6763, RMSE 2.3144
[Open3D DEBUG] Residual : 3.07e+00 (# of elements : 29072)
[Open3D DEBUG] ICP Iteration #87: Fitness 0.6763, RMSE 2.3145
[Open3D DEBUG] Residual : 3.07e+00 (# of elements : 29072)
[Open3D DEBUG] ICP Iteration #88: Fitness 0.6763, RMSE 2.3144
[Open3D DEBUG] Residual : 3.07e+00 (# of elements : 29072)
[Open3D DEBUG] ICP Iteration #89: Fitness 0.6763, RMSE 2.3145
[Open3D DEBUG] Residual : 3.07e+00 (# of elements : 29072)
[Open3D DEBUG] ICP Iteration #90: Fitness 0.6763, RMSE 2.3144
[Open3D DEBUG] Residual : 3.07e+00 (# of elements : 29072)
[Open3D DEBUG] ICP Iteration #91: Fitness 0.6763, RMSE 2.3145
```

Other Approaches

- Another paper is to be explored which uses the octree representation of the point cloud. It uses same graph-based approach
- It embeds information on the nodes in the form of signals. These signals represent the position and color attribute.
- The problem is then converted to feature matching on a graph and estimating the shift in those features in the consecutive graphs and using that to compensate for the motion[3]

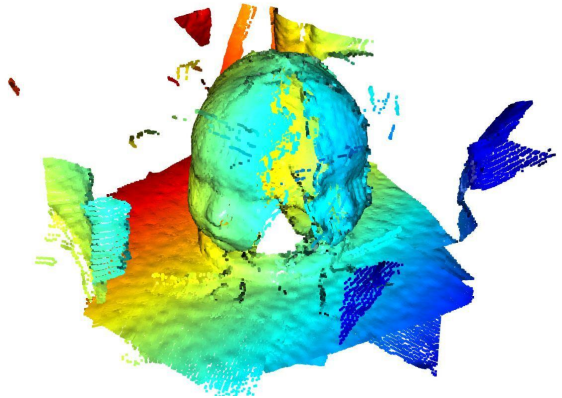
- Another approach, is to use the normals and mapping structures from the scans with similar normals/transforms performed on a sphere.
- The consecutive scans are then registered by finding and matching similar structures on the sphere.
- PhD thesis with MATLAB and C++ code.
- Convert to Python code and using the latest methods to reduce the complexities and length of the code.[4]

Testing

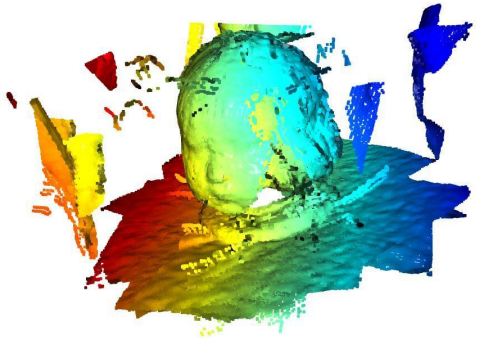
- Testing is to be done for 2 parameters,
  - Angular Resolution
  - Motional Resolution
- For Angular Resolution, multiple scans are to be taken at different angular differences.

Results so far

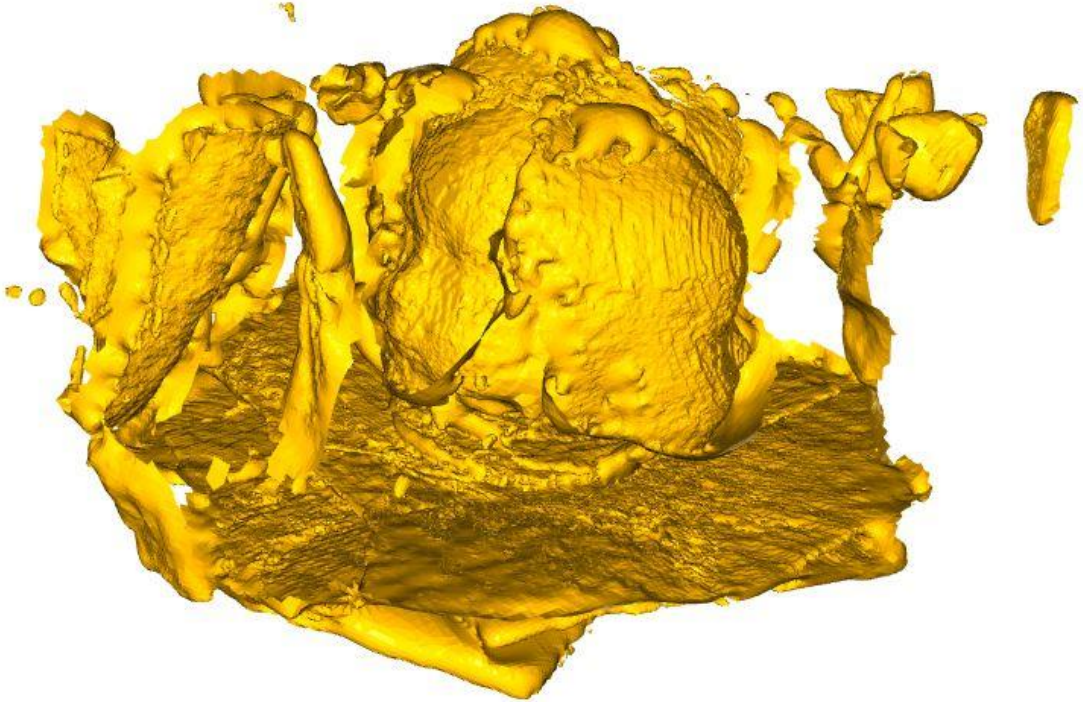
- The result of the existing vanilla ICP on the new scan and the multiway registration are as follows:



Existing Pipeline

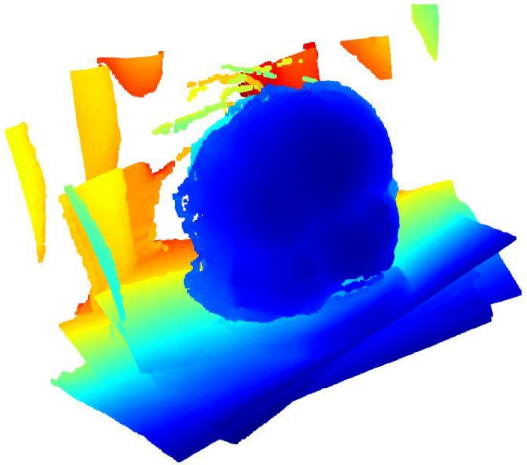


Multiway Registration

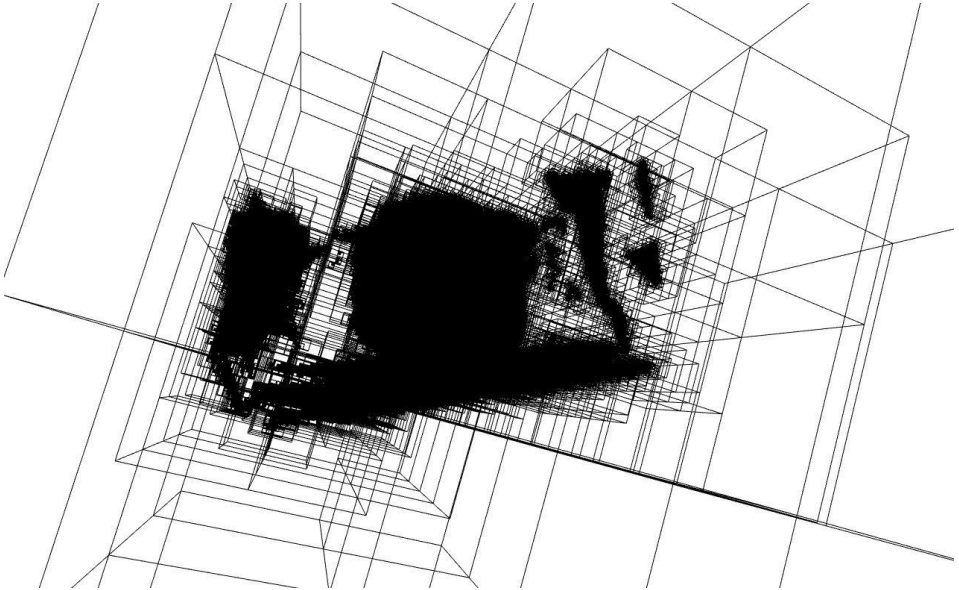


Testing in MATLAB

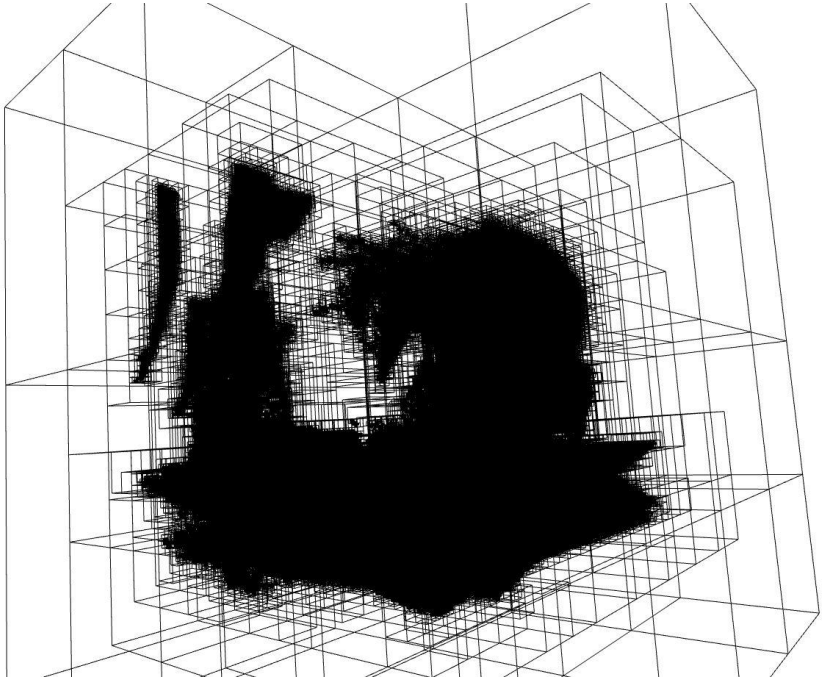
- Testing the existing suite of tools available in MATLAB to try and benchmark the existing solution as well as explore whether the tools available in MATLAB are sufficient to solve the problem.



Octree Generation



Octree - Multiway Registration



Octree - Matlab ICP

Future Work

- Fine tuning the Multiway Method
- Test out another method - octree embedding
- Try Deep Learning based scene flow method(*)
- Testing and documenting

Deliverables

- Minimum:
  - Data collection at various stages
  - Working pipeline for one algorithmic approach
  - Accuracy Evaluation for both static and moving models:
    - < 2 mm average surface distance, ± 2.5% CI/CVAI
- Expected:
  - All of minimum deliverables
  - Testing and documentation of 2 algorithms
  - Accuracy Evaluation for both static and moving models:
    - < 2 mm average surface distance, ± 1.5% CI/CVAI
- Maximum:
  - All of expected deliverables
  - Building a simulation environment and pipeline for data extraction for testing using CoppelliaSim software

Updated Dependencies

Dependency	Status	Fallback	Needed by(latest)	Impact
Sensor + ipad + phantom head	Acquired	Simulation	02/14/22	Delay in testing and implementation
Access to AWS bucket	Acquired	Asking mentors to get data	03/10/22	Delay in testing and implementation
Motion Generator	Not acquired	JHU access - mass testing Bought by company and test Simulation	04/10/22	Delay in evaluation of the algorithms and overall solution

Milestones

Milestone	Planned Date	Delayed Date	Alternative	Status
Debug existing pipeline	02/27/22	03/07/22	-	Complete
Add at least 1 algorithm + debug	03/14/22	03/17/22	-	Complete
Testing + Refining new algorithm	03/21/22	03/24/22	Simulation based testing	On-going
Add another algorithm	04/01/22	04/04/22	Further testing and refinement of 1 algorithm	On-going
Simulation Setup (For Fallback scenario)	04/10/22	04/13/22	-	Not Started
Testing + Refining new algorithm	04/11/22	04/14/22	Simulation based testing	Not Started
Add another algorithm	04/20/22	04/23/22	Further testing and refinement of added algorithms	Not Started
Testing + Refining new algorithm	04/27/2	05/01/22	Simulation based testing	Not Started
Final Documentation	05/03/22	05/05/22	-	Not Started

Management Plan

- Weekly meeting with Mentors on Tuesday
- Slack channel for communication and sharing files
- Code and version management through GitHub
- All documents shared via Google Drive

References

[1] http://www.open3d.org/docs/latest/tutorial/Advanced/multiway_registration.html

[2] Choi, Q.-Y. Zhou, and V. Koltun, Robust Reconstruction of Indoor Scenes, CVPR, 2015.

[3] D. Thanou, P. A. Chou and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 3235-3239, doi: 10.1109/ICIP.2015.7351401.

[4] A. Makadia, A. Patterson and K. Daniilidis, "Fully Automatic Registration of 3D Point Clouds," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2006, pp. 1297-1304, doi: 10.1109/CVPR.2006.122.

Thank you !